Air Traffic Complexity Model Based on Air Traffic Controller Tasks

Antulov-Fantulin, Bruno

University of Zagreb

Faculty of Transport and Traffic Sciences

Bruno Antulov-Fantulin

AIR TRAFFIC COMPLEXITY MODEL BASED ON AIR TRAFFIC CONTROLLER TASKS

DOCTORAL THESIS

Supervisor:

Biljana Juričić, PhD, Associate Professor

Zagreb, 2020

Sveučilište u Zagrebu

Fakultet prometnih znanosti

Bruno Antulov-Fantulin

MODEL KOMPLEKSNOSTI ZRAČNOG PROMETA TEMELJEN NA RADNIM ZADAĆAMA KONTROLORA ZRAČNOGA PROMETA

DOKTORSKI RAD

Mentorica:

izv. prof. dr. sc. Biljana Juričić

Zagreb, 2020

I. Biography of Supervisor

Associate professor Biljana Juričić, PhD, holds position of Head of the Department of Aeronautics at the Faculty of Transport and Traffic Sciences, University of Zagreb, Croatia. The major interest of her scientific work is modelling and simulation in air traffic management, ATCO training and airspace design. She worked as a team leader and researcher on several European scientific projects in the field of air traffic management. She was a leading member of a team in establishing and development of Croatian Air Traffic Control Training Centre (HUSK) and Laboratory for Air Traffic Control. As an author or co-author, she published a number of scientific papers in the field of transport engineering, branch air traffic. She holds several courses on the undergraduate and graduate Study of Aeronautics and as well as on the doctoral Study of Traffic.

II. Acknowledgment

"Patientia Comes Est Sapientiae", applies perfectly to my mentor, prof. Biljana Juričić. The wisdom and the patience she had for me from the beginning of my education is immeasurable. I thank her for believing in me and for always pushing me forward, especially in times when I thought I would not succeed. As often in life, so is the same with this doctoral thesis, *"Ex Nihilo Nihil Fit"*. A lot of work was put into it and now that it is finished, I would not have want it done in any other way. I would not have been able to finish it in time without the great support of my friend, Tomislav Radišić, who was always there, welcoming me when I needed to brainstorm new ideas. Also, I would like to thank Matija Piškorec, who was a great pillar of support in development of machine learning algorithms. I thank my brother Nino, who always had the time and patience to listen to my ideas and to explain and lecture me when I lacked the adequate knowledge. The greatest thanks goes to my dear friend Alen Lančić, without whom this thesis would have been significantly delayed. He provided all the right answers at the right time, taught me in the field of mathematics, and corrected my work when I wandered. Also, I would like to give thanks to my loving parents, who were always there for me and never gave up on my education. I thank my sister Dunja, who is always there to lift my spirit. And finally, I thank my wife, who provided immense support and understanding during my doctoral study period.

I dedicate this work to my daughter Ema, and may she find peace and love in this world.

Post scriptum: this work was proofread by Ivana Francetić and at the time of writing is protected by the patent pending law, ownership of the Faculty of Transport and Traffic Sciences, University of Zagreb.

III. Abstract

Existing models for determining air traffic complexity that are based on air traffic controllers' subjective assessment are not consistent due to possible deviations in complexity assessment. The aim of this research is to create a mathematical model for air traffic complexity which is based on the air traffic controller tasks. The model will use the data on area radar air traffic controller tasks that are defined according to the traffic situation. Certain air traffic controller tasks, such as a conflict resolution, are perceived as one task, but they actually represent a set of multidimensional tasks that need to be defined precisely in order to be used later in mathematical model. Moreover, the existing models for determining air traffic complexity which use the subjective air traffic controller assessments also include the problem of subjectivity resulting from the learned mode of operation in a given airspace. For the purpose of this research new generic airspace will be created. This research introduces a new approach to design a model for determining air traffic complexity which is based on defining area radar air traffic controller tasks for the given traffic situations. Area radar air traffic controllers will be asked to decide which of the two traffic situations is more complex by using the comparison method. In this way, any inconsistency in subjective assessments will be avoided, since air traffic controllers tend to give the same complexity score for different levels of air traffic complexity. Using machine learning, inputs such as defined air traffic controller tasks and data gained through comparison method, will be used to develop a new mathematical model for determining air traffic complexity. The validation of the model will be carried out by the same comparison method using the traffic situation data on a different airspace.

Key words: air traffic complexity, air traffic controller, assessment, workload, tasks.

IV. Prošireni sažetak

Rast potražnje u prometu pokretač je razvoja zračnog prometa. Ipak, to bi moglo dovesti i do negativnih posljedica poput zagušenja zračnog prostora, kašnjenja letova, velike gustoće prometa, neučinkovitost leta zbog pretjerano dugih ruta, povećane potrošnje goriva, a samim time i povećanih troškova leta i utjecaja na okoliš. Ti će problemi postati još izraženiji u narednim godinama, zbog povećane potražnje u prometu.

Trend rasta zračnog prometa u zoni EUROCONTROL od 2013. godine nastavljen je do 2018. godine, nakon nekoliko godina stagnacije uzrokovane globalnom gospodarskom krizom. Broj letova temeljen na pravilima instrumentalnog leta (IFR) u prosjeku je porastao za 3,8% u odnosu na promet u 2017. Rast zračnog prometa veći je u pogledu broja putnika nego u odnosu na letove (6,1% u odnosu na 2017.), što je također slučaj u prethodnim godinama [1]. Taj se rast nastavio u prvoj polovici 2019. godine, a broj kontroliranih letova u zoni EUROCONTROL u prosjeku je porastao za 1,6% u odnosu na 2018. godinu [2]. Prema srednjoročnoj prognozi EUROCONTROL-a, procjenjuje se da će rast prometa IFR-a nastaviti u sljedećim godinama do 2025. godine, s prosječnim godišnjim rastom od 2,0% [3].

U takvim se uvjetima događaju kompleksnije situacije u zračnom prometu, koje mogu otežati pružanje usluge kontrole zračnog prometa, a posebno za specifične zadatke kontrolora zračnog prometa. To može rezultirati povećanim radnim opterećenjem kontrolora zračnog prometa koje predstavlja potencijalni sigurnosni rizik. Kako bi udovoljili prometnoj potražnji, pružatelji usluga u zračnoj plovidbi moraju osigurati odgovarajući sektorski kapacitet koji će omogućiti siguran i učinkovit zračni promet. Budući da kapacitet sektora ovisi o radnom opterećenju kontrolora zračnog prometa, kompleksnost zračnog prometa postaje jedan od ključnih čimbenika koji se razmatra pri istraživanju ovih pokazatelja i sustavu upravljanja zračnim prometom. Kompleksnost zračnog prometa definira se kao poteškoća u praćenju i upravljanju određenom situacijom u zračnom prometu [4].

Jedinstveno europsko nebo (SES), projekt modernizacije i poboljšanja europskog upravljanja zračnim prometom, ima za cilj povećanje sigurnosti prometa, kapaciteta i učinkovitosti, kao i smanjenje negativnih posljedica povećane potražnje zračnog prometa. Nekoliko novih tehnologija (rješenja) razvijeno je putem SESAR-ovog programa upravljanja zračnim prometom (SESAR) kako bi se zadovoljile velike prometne potražnje i osigurala sigurnost u prometu. Kompleksnost zračnog prometa istražuje se i unutar SESAR-a, što je rezultiralo SESAR-ovim rješenjem br. 19 Automatizirana podrška za otkrivanje i rješavanje kompleksnosti (iz SESAR-a 1). Jedna od potfunkcionalnosti koja će se razviti u okviru SESAR2020 je automatizirana podrška za procjenu kompleksnosti prometa koja je propisana Provedbenom uredbom Komisije (EU) br. 716/2014 od 27. lipnja 2014. o uspostavljanju zajedničkog pilot projekta koji podržava provedbu europskog glavnog plana (*Master plan*) upravljanja zračnim prometom.

Ovaj rad daje istraživački pregled modela i metoda za utvrđivanje i ocjenu kompleksnosti zračnog prometa. Na temelju prethodnih istraživanja identificirani su nedostatci postojećih modela, predstavljena je nova metoda za određivanje kompleksnosti zračnog prometa i predložen je novi model koji bi trebao nadmašiti nedostatke koji su i dalje prisutni u ovom polju istraživanja.

Cilj istraživanja: Izraditi model kompleksnosti zračnog prometa temeljen na radnim zadaćama kontrolora zračnog prometa.

Hipoteza: Kompleksnost zračnog prometa moguće je odrediti na temelju radnih zadaća kontrolora zračnog prometa.

Argumenti koji potkrepljuju hipotezu:

- Subjektivne procjene kontrolora zračnog prometa u postojećim modelima određivanja kompleksnosti zračnog prometa nisu konzistentne.
- Modeli za određivanje kompleksnosti zračnog prometa temeljeni na subjektivnim procjenama kontrolora zračnog prometa definirani su u ovisnosti o karakteristikama određenog zračnog prostora te ne daju valjane rezultate u primjeni na druge zračne prostore.
- Radne zadaće aktiviraju se na temelju karakteristika prometnih situacija te su neovisne o kontroloru zračnog prometa koji ih provodi.
- Povećanje kompleksnosti zračnog prometa za posljedicu ima povećanje radnog opterećenja kontrolora zračnog prometa, a radno opterećenje se može izraziti kao skup radnih zadaća kontrolora zračnog prometa.

S ciljem potvrđivanja postavljene znanstvene hipoteze, istraživanje će biti provedeno kroz šest temeljnih faza.

U prvoj fazi istraživanja definirat će se radne zadaće oblasnog radarskog kontrolora zračnog prometa koristeći metodu intervjuiranja kontrolora, analizom postojeće literature te priručnika koji objašnjavaju radne zadaće kontrolora te metodom promatranja rada kontrolora na radnom mjestu. Kontrolori zračnog prometa izvršavaju radne zadaće ovisno o prometnoj situaciji stoga je iznimno važno pravilno definirati sve radne zadaće koje se provode te kreirati veliki broj različitih prometnih situacija. S obzirom na to da se određene radne zadaće, poput razrješavanja konflikta, broje kao jedan problem, a u stvarnosti su višedimenzionalni problem, potrebno je definirati takve, višedimenzionalne radne zadaće koje će se moći koristiti u matematičkom modelu.

U drugoj fazi istraživanja potrebno je kreirati prometne situacije iz kojih se mogu iščitati sve potrebne informacije koje će omogućiti oblasnom radarskom kontroloru zračnog prometa da procijeni kompleksnost prometne situacije. Prometne situacije kreirat će se kao statične slike koje će sadržavati sve potrebne informacije za procjenu kompleksnosti situacije, poput brzine zrakoplova, smjera letenja zrakoplova, destinacije zrakoplova, ulazne i izlazne točke u sektoru, namjere pilota, definirane granice zračnog prostora, trenutne visine zrakoplova, izlazne visine itd. Prometne situacije bit će definirane za generički zračni prostor da se izbjegne subjektivnost ocjenjivanja na poznatim zračnim prostorima i prometnim situacijama. Također, razlog korištenja generičkog zračnog prostora je mogućnost kasnije primjene modela na različite zračne prostore. Prometne situacije imat će varijabilan broj zrakoplova, različite međusobne interakcije ovisno o poziciji, visini, smjeru i brzini kretanja, različit položaj u prostoru, udaljenost od granice prostora, itd. Također u ovoj fazi istraživanja kreirat će se prometne situacije za drugi zračni prostor koji će se kasnije koristiti u zadnjoj fazi istraživanja za validaciju modela na različite zračne prostore.

U trećoj fazi istraživanja bit će potrebno odrediti radne zadaće na temelju prometnih situacija. Definirane radne zadaće iz prve faze istraživanja dodjeljivat će se prometnim situacijama iz druge faze uz pomoć automatiziranog sustava. Radne zadaće definirane su ovisno o prometnim situacijama gdje za svaku radnu zadaću postoje jasno definirana pravila kada se aktiviraju i kada se trebaju provesti. Primjeri radnih zadaća koje se provode su: monitoriranje zračnog prometa, izvršavanje zahtijeva pilota, koordinacija sa susjednim zračnim prostorom, razrješavanje konflikta, itd. Na taj način postojat će jasno definirane radne zadaće za svaku prometnu situaciju.

U četvrtoj fazi istraživanja testirat će se oblasni radarski kontrolori zračnog prometa. Primijenit će se metoda komparacije kojom će oblasni radarski kontrolori zračnog prometa između dvije ponuđene prometne situacije morati odrediti koja je kompleksnija. Primijenit će se 120 prometnih situacija koje će omogućiti aktivaciju svih mogućih radnih zadaća. Po završetku usporedbi prometnih situacija, kontrolori će imati jasan poredak od najmanje do najviše kompleksne prometne situacije koju su sami prethodno poredali metodom komparacije te ih grupirati u ocijene kompleksnosti prometa od 1 do 5. Na osnovu tih ocjena, te prethodnih usporedbi prometnih situacija dodijelit će se linearno interpolirane ocijene ostalim prometnim situacijama. Istom metodom prikupit će se podaci za drugi zračni prostor za potrebe validacije modela.

U petoj fazi istraživanja uz pomoć strojnog učenja trenirat će se linearni model koristeći Bayesian Ridge regresije da se radnim zadaćama (istraživačkim varijablama) dodjele težinske vrijednosti na osnovu linearno interpoliranih ocjena iz prethodne faze (ciljane varijable). Na taj način izraditi će se model za određivanje kompleksnosti zračnog prometa na temelju radnih zadaća kontrolora zračnog prometa.

U šestoj fazi istraživanja radit će se validacija matematičkog modela temeljem subjektivnih procjene kontrolora zračnog prometa dobivenim na drugom zračnom prostoru, te će se vidjeti je li moguća primjena modela na različite zračne prostore.

V. Table of Content

VI. List of Abbreviations

ANN – Artificial Neural Networks

ANSP – Air Navigation Service Provider

ATC – Air Traffic Control

ATCEM – Air Traffic Complexity Evaluation Model

ATCO – Air Traffic Controller

ATM – Air Traffic Management

ATWIT – Air Traffic Workload Input Technique

BPNN – Back Propagation Neural Network

CAL – Conflict Activity Level

CLFL – Cleared Flight Level

CUFL – Current Flight Level

EDA – Exploratory Data Analysis

EEC – EUROCONTROL Experimental Centre

EFL – Exit Flight Level

FL – Flight Level

HITL – Human-In-The-Loop

HMI – Human Machine Interaction

IDA – Initial Data Analysis

LOAA – Learn Once Apply Anywhere

LOO – Leave One Out

NM – Nautical Mile

PCA – Principal Component Analysis

SES – Single European Sky

SESAR – Single European Sky ATM Research

STAM – Short Term ATM Measures

SVM – Support Vector Machines

TBO – Trajectory Based Operations

TBX – Trajectory Based Complexity

VII. List of Figures and Tables

1. Introduction

The growth in traffic demand is a driver of air traffic development but it can also lead to negative consequences such as airspace congestion, flight delays, high traffic density, flight inefficiency due to excessively long routes, increased fuel consumption, and therefore, increased flight costs and environmental impact. These problems will become even more pronounced in the coming years, due to the increased traffic demand.

The trend of air traffic growth in the EUROCONTROL zone, after a few years of stagnation caused by the global economic crisis, continued from 2013 until 2018. The number of flights based on instrument flight rules (IFR) grew by 3.8 % on average compared to the traffic in 2017. Air traffic growth is larger in terms of passenger numbers than in terms of flights (6.1 % compared to 2017), which was also the case in the preceding years [1]. This growth continued in the first half of 2019, with the number of controlled flights in the EUROCONTROL zone increasing by 1.6% on average, compared to 2018 [2]. According to the EUROCONTROL medium-term forecast, it is estimated that the growth of IFR traffic will continue in the following years to year 2025 with the average annual growth of 2.0% [3].

In such conditions, more complex air traffic situations occur which may impede the provision of air traffic control service, in particular, specific air traffic controller tasks. This can result in increased air traffic controller workload that poses a potential safety hazard. To meet the traffic demand, air navigation service providers must ensure adequate sector capacity that will allow safe and efficient air traffic. Since sector capacity depends on air traffic controller workload, air traffic complexity becomes one of the crucial factors that is considered when investigating these indicators and air traffic management system. Air traffic complexity is defined as the difficulty of monitoring and managing a specific air traffic situation [4].

Single European Sky (SES), which is the project of modernizing and improvement of European air traffic management, aims to increase traffic safety, capacity and efficiency as well as reduce the negative consequences of increased air traffic demand. Several new technologies (solutions) have been developed through SES air traffic management research (SESAR) program to meet the high traffic demand and to ensure traffic safety. Air traffic complexity is investigated and researched within the SESAR which resulted in the SESAR Solution#19 Automated Support for Complexity Detection and Resolution (from SESAR 1). One of the sub-functionalities to be developed within SESAR2020 is Automated Support for Traffic Complexity Assessment which is prescribed in the Commission Implementing Regulation (EU) No 716/2014 of 27 June 2014 on the establishment of the Pilot Common Project supporting the implementation of the European Air Traffic Management Master Plan.

This paper gives the research overview of models and methods for determining and assessing air traffic complexity. Based on previous research findings, shortcomings on the existing models were identified, a novel method for determining air traffic complexity is presented and a new model that surpasses the flaws that are still present in this field of the research is given.

1.1. Motivation and Aims

This research was motivated by the fact that almost after two decades of research, the problem of determining adequate complexity score is still an issue in air traffic control, because it is considered subjectively, from the air traffic controllers' perspective. The air traffic controllers observe and analyze the traffic data and decide whether a traffic situation is complex or not. All other methods are just attempts to approximate the level of complexity according to air traffic controllers' subjective assessment.

Therefore, the main objective of this research was to measure the effect of air traffic controller tasks on air traffic complexity in en-route operations. This was achieved by developing and defining a set of air traffic controller tasks and traffic situations that the air traffic controllers assessed and ranked from lowest to highest complexity. Ranked traffic situations were then graded by the air traffic controllers according to the complexity scores from 1 to 5 where 1 was the lowest complexity score and 5 was the highest. These scores were used as target variables to train the model to calculate the complexity of a specific traffic situation based on the air traffic controllers' tasks.

1.2. Methods

One of the most used methods for the complexity calculation of air traffic patterns is through air traffic controllers' subjective assessment. The main goal of air traffic controllers' subjective assessment is to embed the controller's complexity metric into the model calculation. To do this, firstly, a clear definition of the air traffic controller tasks was defined along with the new airspace and traffic situations. Secondly, a group of licensed air traffic controllers were taken to assess the defined traffic situations. They were assessing the situations by comparing the given pair of traffic situations and sorting them with the merge sort algorithm. By the end of the experiment a clear rank from lowest to highest complexity traffic situations was sorted by the air traffic controller. After the ranking, controllers were giving the ranked traffic situations a complexity score from 1 to 5 where 1 was the lowest complexity score and 5 was the highest.

Based on the controllers' complexity ranking and scoring, linearly interpolated grades were assigned to each situation for each controller. Later on, all complexity scores were used as target variables to train the model on how specific air traffic controller tasks (exploratory variables) contribute to air traffic complexity. At the end, a real airspace was used to verify the model. Air traffic controllers have assessed (using the same methodology) air traffic complexity of the traffic situation within new, real airspace, thus allowing us to make a correlation of the newly assessed complexity and one determined by the new model.

1.3. Research Objective and Hypothesis

Based on the previously defined motivation and methods, a hypothesis with the arguments that support it are defined here.

Research objective: Create air traffic complexity model based on air traffic controller tasks.

Hypothesis: The air traffic complexity can be determined on the basis of air traffic controller tasks.

Arguments that support the hypothesis:

- Subjective assessment given by air traffic controllers for the existing models in determining the air traffic complexity are not consistent.
- Models for determining the air traffic complexity based on subjective air traffic controllers' assessments are defined depending on the characteristics of the specific airspace and do not provide valid results if applied to another airspace.
- The air traffic controller's tasks are defined on the basis of the characteristics of the air traffic situation and do not depend on the person controlling the air traffic.
- Increase in air traffic complexity results in increase of air traffic controller workload, and the workload can be expressed as a set of air traffic controller tasks.

1.4. Expected Scientific Contribution

Following scientific contributions are expected in the field of Traffic and Transport Technology:

- Definition of air traffic controller tasks depending on the characteristics of traffic situations.
- Development of a mathematical model for determining the air traffic complexity based on the air traffic controller tasks.

- Determining the weight value for the individual air traffic controllers' tasks or their combination in the overall complexity of air traffic.

1.5. Outline

In the **introductory chapter**, the motivation for the research, hypothesis, and the research objective were presented. Additionally, the overview of the methods used and the expected scientific contribution were given.

In the second chapter, titled **Air Traffic Complexity**, the term air traffic complexity and air traffic controller workload are defined. Also, in this chapter an overview of air traffic complexity models and methods most relevant to this field of research are given.

The third chapter, titled **Experiment Methodology**, contains the detailed description of methodology process which was used to set up the experiments. Detailed definition of the air traffic controller tasks along with the automatization process that will serve later on for the training of the model. Furthermore, airspace and traffic along with the detailed data gathering process is described.

Fourth chapter, titled **Model development**, covers the description of model development methodology which was used to train the air traffic complexity model from the previously obtained target and exploratory variables from the third chapter. A detailed exploratory data analysis along with the feature construction is presented. A final formula for the air traffic complexity is presented.

In the fifth chapter, titled **Result analysis**, the results from the air traffic complexity model are analyzed and compared with the complexity scores that the air traffic controllers gave. Also, the models are validated on a new airspace and the practical application is given.

Conclusions and proposals for future research are elaborated in the **final chapter**. In this chapter, all relevant research objectives are reviewed and the effect of air traffic controller tasks on air traffic complexity is elaborated.

2. Air Traffic Complexity

2.1. Air Traffic Complexity and Air Traffic Controller Workload

Air traffic complexity has been a common research topic since the early days of modern air traffic control (ATC) operations. At the beginning, most of the research was dealing with the air traffic controller (ATCO) workload instead of air traffic complexity to express how difficult some ATCO tasks were. Because of that, it is important to explain the relation between these two indicators. The first papers that deal with complexity were written in the early 1960s [5]. Since then, numerous papers and reports have been written on the topic of complexity – excellent reviews of those papers were written by Mogford [6] and Hilburn [7]. Their conclusion was that the air traffic complexity is a fundamental driver of workload but that the connection between complexity and workload is not straightforward; it is mediated by other factors, such as equipment quality, individual differences, and controller cognitive strategies (Figure 1) [6]. It can be noticed that most of the early research has been conducted in order to better define factors that affect air traffic controller workload. From today's point of view and with present understanding and definitions, the majority of these factors would probably be classified as complexity factors.

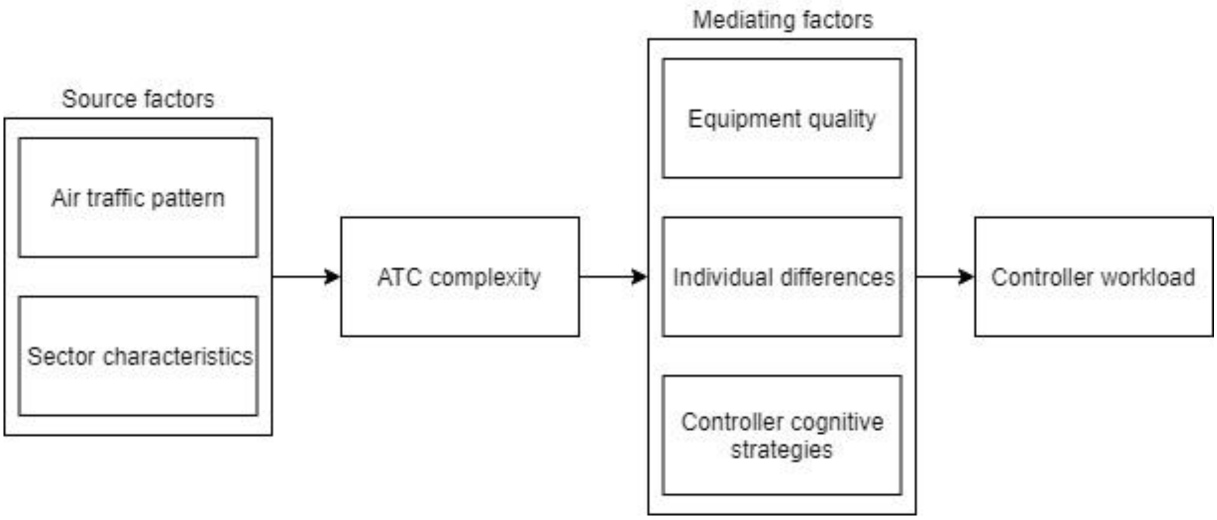


Figure 1: Relationship between ATC complexity and workload [6]

Schmidt [8] approached the problem of modelling controller workload from the angle of observable controller actions. He created the control difficulty index, which can be calculated as a weighted sum of the expected frequency of occurrence of events that affects controller workload. Each event is given different weight according to the time needed to execute a particular task. Although the author conducted extensive surveys to determine appropriate weights and frequencies for various events, this approach can only handle observable controller actions, which makes this approach very limiting.

Even though Hurst and Rose [9], were not the first to realize the importance of traffic density, they were the first to measure the correlation of expert workload ratings and traffic density. They concluded that only 53% of the variance in reported workload ratings can be explained by density.

Stein [10] used the Air Traffic Workload Input Technique (ATWIT), in which controllers report workload levels during simulation, to determine which of the workload factors influenced workload the most. Regression analysis proved that out of the five starting factors, four factors (localized traffic density, number of handoffs outbound, total amount of traffic, number of handoffs inbound) could explain 67% of variance in ATWIT scores. This study showed the importance of localized traffic density which is a measure of traffic clustering. A technique similar to ATWIT will be used throughout the next three decades.

2.2. Overview of Air Traffic Complexity Models and Methods

Today, air navigation service providers still use air traffic controllers' subjective assessment as the most important method for determining air traffic complexity, even though there are many studies that have dealt with the development of new, more objective methods for determination of air traffic complexity. The most important scientific papers dealing with the methods and models for determining air traffic complexity are based on the subjective assessment by the air traffic controllers.

Laudeman et al. [11] expanded on the notion of the traffic density by introducing Dynamic Density which they defined as a combination of 'both traffic density (a count of aircraft in a volume of airspace) and traffic complexity (a measure of the complexity of the air traffic in a volume of airspace)'. Authors used informal interviews with controllers to obtain a list of eight complexity factors to be used in the dynamic density equation. The only criterion was that the factors could be calculated from the radar tracks or their extrapolations. The intention was to obtain an objective measure of controller workload based on the actual traffic. Their results showed that the dynamic density was able to account for 55% of controller activity variation. Three other teams [12–14] working under the Dynamic Density program developed additional 35 complexity indicators (factors), which were later successfully validated as a group by Kopardekar et al. [15]. Unfortunately, it was later shown that the complexity indicator weights were not universal to all airspace sectors, i.e. they had to be adjusted on a sector by sector basis [16]. This shortcoming, while making Dynamic Density technique difficult to implement for operational purposes, has no influence if one wishes to compare two concepts of operations under similar conditions (similar sector configuration). Furthermore, same authors [15] suggested that, due to possibly non-linear interactions between complexity factors, the Dynamic Density performance could be improved by using non-linear techniques such as non-linear regression, genetic algorithms, and neural networks.

Almost the same group of authors will use multiple linear regression method five years later to determine which subset of complexity indicators will correlate well with the controller's subjective complexity ratings [17]. After extensive simulator validation, results of this study showed that there are 17 complexity indicators that are statistically significant. Top five complexity indicators were: sector count, sector volume, number of aircraft under 8 NM from each other, convergence angle, and standard deviation of ground speed/mean ground speed. Similar work was done by Masalonis et al. [18] who selected a subset of 12 indicators, and Klein et al. [19] who selected a subset of only seven complexity indicators, though with less extensive experimental validation.

In a similar vein, Bloem et al. [20] tried to determine which of the complexity indicators had the greatest predictive power in terms of future complexity. The authors concluded that there is a significant difference in the predictive power of different complexity indicators. To complicate matter further, they concluded that the subset of the complexity indicators that had the best predictive power changed depending on the prediction horizon.

To calculate the potential impact of air traffic complexity on workload and costs, in 2000 the EUROCONTROL has given the same set of traffic data to UK National Air Traffic Services (NATS) and EUROCONTROL Experimental Centre (EEC) with a task of independently devising a method of measuring the level of service [21]. While NATS has estimated ATS output (the service provided), the EEC has estimated the ATS workload needed to deliver the service. Both 'were found to produce reasonably consistent results', with additional note that further analysis should be done before the final parameters for determining ATS provider costs are established. By 2006 EUROCONTROL's Performance Review Commission finalized the complexity indicators to be used for ANSP benchmarking [22]. For this method the European airspace is divided into 20 NM X 20 NM X 3000 ft cells, and for each cell the duration of potential interactions is calculated. Aircraft are 'interacting' if they are in the same cell at the same one hour time frame window. The ratio of the hours of interactions and flight hours is so called ‘Adjusted Density’. In addition, the ‘Structural Index’ is calculated as a sum of potential vertical, horizontal and speed interactions. The final complexity score is calculated as a product of adjusted density and structural index. All in all, only 4 complexity indicators are used for this analysis and no validation of any sort was presented in the report. It was noted, however, that shifting the starting position of the grid by 7 NM caused the ANSP ranking to change dramatically (up to 16 places in an extreme case). Nonetheless, this method is still used for ANSP benchmarking.

The first to consider measuring complexity during trajectory-based operations (TBO) were Prevot and Lee in 2011 [23]. They coined the term Trajectory-based Complexity (TBX) which is a measure of complexity in TBO. The basis of the TBX calculation is a set of nominal conditions – nominal sector size, nominal number of transitioning aircraft, and a nominal equipage mix. Any difference to nominal operations causes a modification to the TBX value. Authors do not explain the method to determine the nominal conditions except that they can ‘be defined through knowledge elicitation sessions on a sector by sector basis or based upon more generic attributes’. The TBX value is then a number of aircraft that would produce the same workload under the nominal conditions as do aircraft under real conditions (e.g. the TBX of 20 means that the workload is equal to the aircraft count of 20 under nominal conditions even though there are actually only 16 aircraft in the sector). The advantage of this method is that it gives a single complexity value that can be easily related to aircraft count and is thus very user-friendly and self-explanatory (unlike many other complexity metrics). However, this study included only six complexity indicators with weights that were determined in an ad-hoc manner and hardly any validation with actual subjective complexity. Only one of those complexity indicators was indirectly related to TBO (number of aircraft with data-link). Many Human-In-The-Loop (HITL) simulation runs were performed in which the controllers had to give workload scores which were then compared with TBX value and simple aircraft count. While the authors claim that the subjective workload score correlated better with the TBX value, there was no objective correlation assessment presented. Finally, the authors have not compared the effect of fraction of TBO aircraft on air traffic complexity.

In a subsequent paper by the same authors, the relationship between workload and data-link equipage levels was explored [24]. It was concluded that the workload ratings correlated much better with the TBX score than with the aircraft count for varying data-link equipage levels.

Another study of the complexity of TBO was made by Radišić et al. [25]. The authors investigated how transitioning from conventional to trajectory-based operations affects the air traffic complexity for the area radar air traffic control. They developed a series of scenarios and simulated the conditions of different traffic loads. They used licensed air traffic controllers to implement HITL simulations. During the simulation, the controllers assessed the complexity of the traffic situation on a scale from 1 to 7 (modified ATWIT grading scale). The authors proved that the subjective air traffic complexity has significantly decreased in trajectory-based operations. It has been experimentally demonstrated that the decrease in air traffic complexity was significant only in traffic situations with a larger number of aircraft and with a larger share of aircraft flying in accordance with TBO.

Prandini et al. have developed a new method of mapping complexity based exclusively on traffic density [26]. This method is applicable only to the future concept of aircraft self-separation and does not take into account the human factors at all.

Gianazza [27–29] proposed a method for prediction of air traffic complexity using tree search methods and neural networks. This method is based on the assumption that the air traffic complexity in historic flight data increased prior to the splitting of the collapsed sector into two smaller ones and decreased prior to collapsing the sectors into larger one. The neural network was trained using this historical data and then it could predict future increase in air traffic complexity. Tree search method was then used to determine the airspace configuration which yields lowest workload and complexity for the given air traffic pattern.

Lee et al. [30] have proposed that airspace complexity can be described in terms of how the airspace (together with the traffic inside it and the traffic control method) responds to disturbances. The effect of disturbances on control activity needed to accommodate that disturbance is what defines complexity in their opinion. The more control activity needed the more complex the airspace is. They propose a tool, airspace complexity map, which should help to plan the airspace configuration and the future development of ATM.

Wee et al. [31] developed a dynamic tactical complexity model, known as Conflict Activity Level (CAL) that evaluate the likely aircraft flight shape profile based on its current and projected position and trajectory. From the flight shape profile, CAL values are computed and overall complexity score is given. Authors state that the proposed complexity approach shows good agreement with other methods in terms of ranking the order of complexity of various air traffic scenarios.

Dervic and Rank [32] used comparison method while interviewing the ATCO’s to develop a formula that is capable of calculating traffic complexity in the terminal area. First group of answers, taken from questioning the ATCO's, studied the complexity of the scenarios individually and ranked the scenarios in reference to each other. Those answers were used to make the formula by linear regression models and the second group of ATCO's were used to validate the same formula. Despite the small amount of data samples, authors were able to prove a genuine relation between variables and the traffic complexity.

Wang et al. [33] constructed a dynamic weighted network by considering aircraft, waypoints, and airways as nodes, and the complexity relationships among those nodes as edges. Complexity is defined as the sum of the weights of all edges in the network and the results indicate that the new complexity index is more accurate than traffic count. Thus, complexity-based management is more efficient than the traffic count-based management.

Xiao et al. [34] developed ATCEM – an air traffic complexity evaluation model that consists of three elements: selected complexity factors as an input data, air traffic complexity level as an output data and classifier for mapping relationship between complexity factors and complexity level. In this model, 7 critical complexity factors are selected from the complexity factor pool by genetic algorithm. Model was trained according to aviation domain knowledge and using back propagation neural network (BPNN) and large sample data. Although ATCEM was positively empirically evaluated, authors suggest further research and model improvement by building more effective integration method that would increase the classification performance of air traffic complexity.

Similar work was done by Xi ZHU et al. [35]. Authors proposed a new model to measure air traffic complexity based on small samples. Authors generated multiple small-size complexity factor subsets from complexity factor pool and used improved machine learning model – random subspace to train the model. Basic complexity evaluator was built according to each factor subset. Final complexity measure is obtained by integrating all results from the basic complexity evaluators. Although model's performance was experimentally evaluated and showed advantages comparing to other small sample complexity models, authors propose model improvement by optimizing some parameters and using semi-supervised machine learning techniques.

Andraši et al. [36] proved that artificial neural networks (ANNs) can be used to determine air traffic complexity with accuracy similar to linear models. They conducted the human in the loop experiments with licensed air traffic controllers and concluded that the remaining variance in subjective complexity scores cannot be explained by traffic characteristics. Also they stated that one of the problems for the errors were inconsistent ATCOs grading of traffic situations. ATCOs could not score (rate) traffic situations with perfect consistency among themselves or between different traffic situations.

3. Experiment Methodology

In this chapter, the development of a new methodology and a novel complexity model based on air traffic controller tasks will be presented. The air traffic controller's tasks are defined on the basis of characteristics of the air traffic situation and do not depend on the person controlling the air traffic. Because of this, by defining a set of air traffic controller tasks based on the pre-conflict resolution parameters, the model could calculate adequate complexity score. The tasks so defined could be applicable to other airspaces and would not be tied to the specific air traffic controller. Additionally, to ensure the possibility that the model could be used on different airspaces, traffic situations were defined on a generic airspace to avoid air traffic controller subjective assessment for the already known traffic situations and airspaces. In the experiments, air traffic controllers only evaluated air traffic complexity by comparing two presented traffic situations at the same time. Upon selecting which one of the two presented traffic situations was more complex, they received a set of two new traffic situations for complexity assessment. Traffic situations were presented and given to air traffic controllers by paper static images. Images are similar to the radar image of the real air traffic controller's working position. All air traffic situations are designed to contain a specific number of aircraft in different positions that influence the activation of the appropriate combination of air traffic controller tasks. There are 120 unique traffic situations developed and divided into six groups. Each group consists of 30 traffic situations out of which 18 are unique to that group, while 12 traffic situations are the same and repeated in all six groups. Repeated traffic situations are created intentionally in order to better evaluate the air traffic controller assessment between themselves. There were three air traffic controllers per group, so in total, 18 ATCOs were tested.

Before the assessment, the air traffic controllers were briefed on the definitions of complexity, workload and airspace capacity to ensure that all of them had the same level of understanding of the air traffic complexity and that they understood what they were expected to do during the assessment. The ranking of the traffic situations was done by the merge sort algorithm. The air traffic controller input was taken as a sort list rule. By ranking them in this manner, it was impossible for two traffic situations to have the same complexity score. Using that approach, possible inconsistency in the assessment was eliminated, so that by the end of the validations, there was a clear rank from the lowest to the highest complexity. Furthermore, to establish a clear grading system, controllers were asked to assess the ranked traffic situations and give a complexity score, from 1 to 5, where 1 is the lowest and 5 is the highest complexity score. Based on the controllers' complexity ranking and scoring, linearly interpolated grades were assigned to each situation for each controller. Later on, all complexity scores were used as target variables to train the model on how specific air traffic controller tasks (exploratory variables) contribute to air traffic complexity. Finally, a real airspace was used to verify the model. Air traffic controllers assessed (using the same methodology) air traffic complexity of the traffic situation within a new, real airspace, thus allowing a correlation comparison of the newly assessed complexity and one determined by the new model.

The example of the above described methodology is depicted in Figure 2 below.

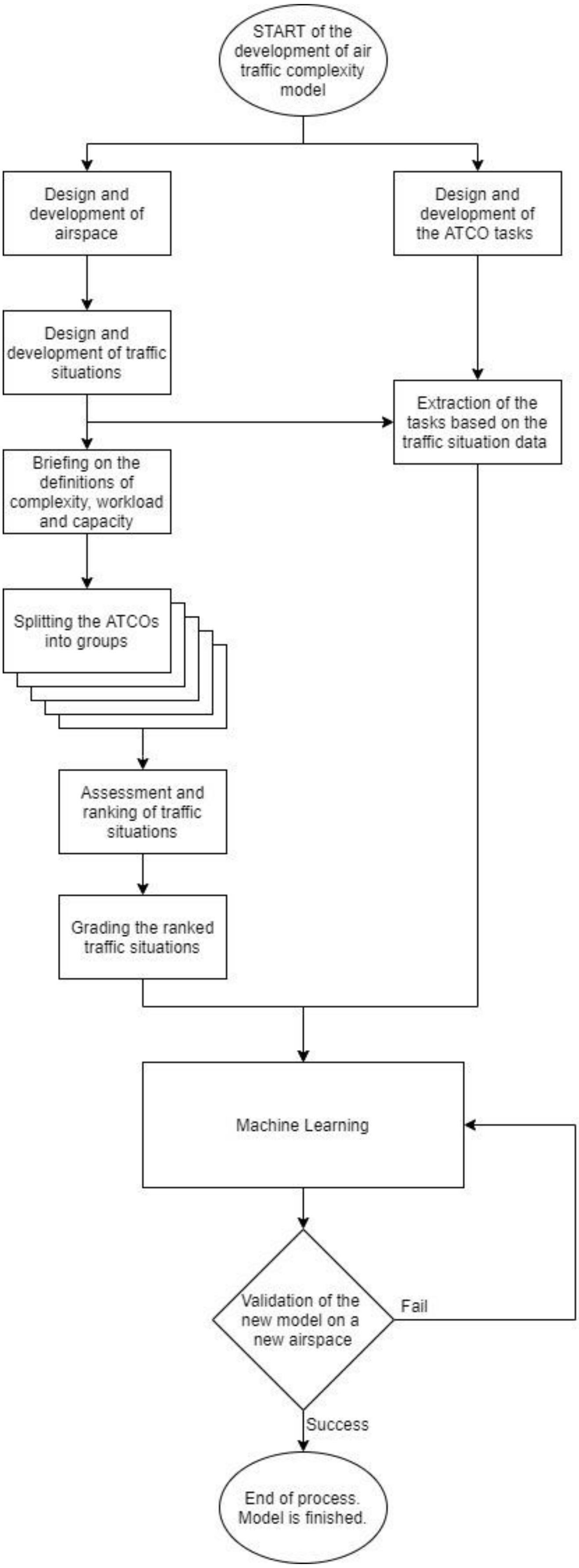


Figure 2: Methodology and plan of research

3.1. Airspace and Traffic

After defining the experiment methodology, in order for the experiment to proceed, a careful definition of airspace and air traffic is needed because the traffic will trigger the exact air traffic controller tasks in the experiments.

3.1.1. Airspace

Traffic situations are defined on a generic airspace (Figure 3) to ensure the possibility of using the model on different airspaces and to avoid air traffic controller subjective assessment for the already known traffic situations and airspaces. Black-shaded area is the controlled airspace S_0, while S_1, gray-shaded area, is the airspace controlled by other controllers. The S_0 airspace is hexagon shaped, diagonally 100 NM in distance and vertically 10000 ft. The extended airspace S_1 is formed by making a homothetic transformation from the centroid of the airspace S_0 for a factor D for each point from the airspace S_0. In this research, the preferred embodiment of the parameter D has a value of 1.5. Airspace S_0 and all aircraft inside it (white) are under the responsibility of the executive controller. Other aircraft (green), although they are outside the jurisdiction of the executive controller, are generating certain coordination tasks. Airspace is designed to simulate free route operations.

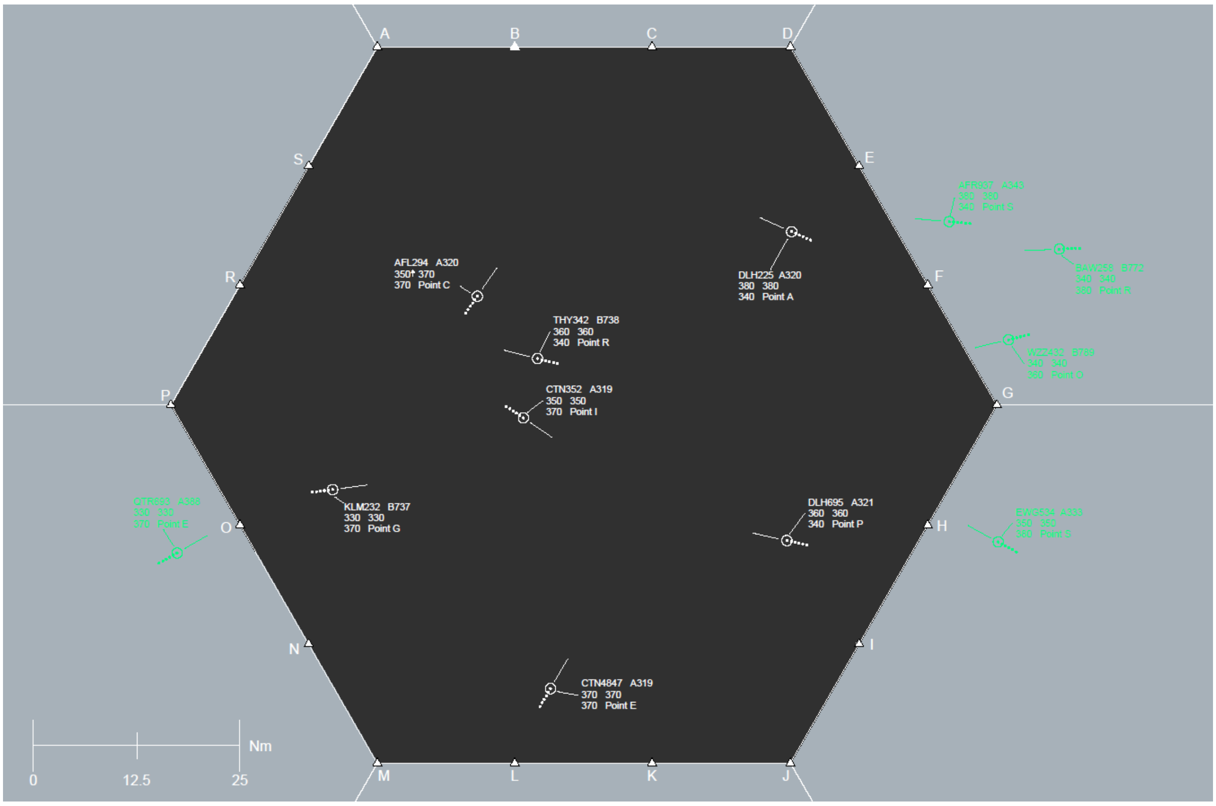


Figure 3: Example of the generic airspace

3.1.2. Traffic

Air traffic situations w_l ; $l \in \{A1, \dots, A40, B1, \dots, B40, C1, \dots, C40\}$ are carefully created so that the air traffic is distributed into three main categories: Low complexity (A in Table 1), Medium complexity (B in Table 1) and High complexity traffic (C in Table 1). There are 40 traffic situations in each category, and each has the exact number of aircraft that triggers a specific ATCO task. There are 120 unique traffic situations developed and divided into six groups (G1-G6 in Table 1). Each group consist of 30 traffic situations from which 18 are unique to that group and 12 traffic situations are the same and repeated in all six groups (Table 1). Repeated traffic situations are created intentionally in order to better evaluate the air traffic controller assessment between themselves. There were three air traffic controllers per group, so in total 18 ATCOs were tested. In Table 1 the number before the slash represents the aircraft number in the controlled airspace and the number after the slash represents the number of aircraft that are about to enter the airspace (green aircraft in Figure 3). Abbreviation G1-G6 was used for the control group of the air traffic controllers.

Table 1: Shows a detail overview of each traffic situation

No. A/C	**6/4**	**6/4**	**7/5**	**7/5**	**8/5**	**8/5**	**9/5**	**9/5**	**10/6**	**10/6**
G1	A1	A5	A6	A2	A7	A8	A3	A9	A10	A4
G2	A1	A11	A12	A2	A13	A14	A3	A15	A16	A4
G3	A1	A17	A18	A2	A19	A20	A3	A21	A22	A4
G4	A1	A23	A24	A2	A25	A26	A3	A27	A28	A4
G5	A1	A29	A30	A2	A31	A32	A3	A33	A34	A4
G6	A1	A35	A36	A2	A37	A38	A3	A39	A40	A4
No. A/C	**8/5**	**8/5**	**9/6**	**9/6**	**10/6**	**10/7**	**11/7**	**11/7**	**12/8**	**12/8**
G1	B1	B5	B6	B2	B7	B8	B3	B9	B10	B4
G2	B1	B11	B12	B2	B13	B14	B3	B15	B16	B4
G3	B1	B17	B18	B2	B19	B20	B3	B21	B22	B4
G4	B1	B23	B24	B2	B25	B26	B3	B27	B28	B4
G5	B1	B29	B30	B2	B31	B32	B3	B33	B34	B4
G6	B1	B35	B36	B2	B37	B38	B3	B39	B40	B4

No. A/C	10/6	10/6	11/7	11/7	12/8	12/8	13/9	13/9	14/10	14/10
G1	C1	C5	C6	C2	C7	C8	C3	C9	C10	C4
G2	C1	C11	C12	C2	C13	C14	C3	C15	C16	C4
G3	C1	C17	C18	C2	C19	C20	C3	C21	C22	C4
G4	C1	C23	C24	C2	C25	C26	C3	C27	C28	C4
G5	C1	C29	C30	C2	C31	C32	C3	C33	C34	C4
G6	C1	C35	C36	C2	C37	C38	C3	C39	C40	C4

As mentioned earlier in the experiment methodology chapter, new validation traffic scenarios will be used to validate the trained model. For that scenario, the Top-High-North airspace sector was taken from the Croatian airspace. This chosen airspace largely differs from the original hexogen shape, by its volume and border design, thus making it an excellent choice to test the model on a new airspace. A detailed overview of each new validation traffic situation is shown in Table 2.

Table 2: Shows a detail overview of new validation traffic scenarios

No. A/C	7/5	10/6	9/6	12/8	11/7	14/10
G1	V1	V2	V3	V4	V5	V6
G2	V7	V8	V9	V10	V5	V6
G3	V7	V8	V9	V10	V5	V6
G4	V11	V12	V13	V14	V15	V16
G5	V17	V18	V19	V20	V21	V22
G6	V23	V24	V25	V26	V27	V28

In Table 2 it can be seen that some traffic situations were repeated for the same effect as in the original traffic scenarios in order to better evaluate the air traffic controller assessment between themselves. Based on the prescribed data in this chapter, air traffic was created in the program AutoCAD 2017. All the 120 initial traffic situations plus 28 new validation scenarios can be seen and examined in the Appendix 1 and the Appendix 2.

3.2. Defining conflict

In order to explain more clearly the tasks defined at this stage of the research, firstly, it is required to explain what is considered as a conflict between two aircraft and what the minimum separation cylindrical zone of the aircraft is. In air traffic, for the en-route phase of the flight, minimum vertical distance H_{min} that aircraft need to maintain is 1000 ft and minimum horizontal distance S_{min} is 5 NM. If any other aircraft flight path violates the cylinder boundary, it is called a conflict, and any potential flight path that could violate the minimum norm of the cylindrical boundary of the aircraft is called a potential conflict.

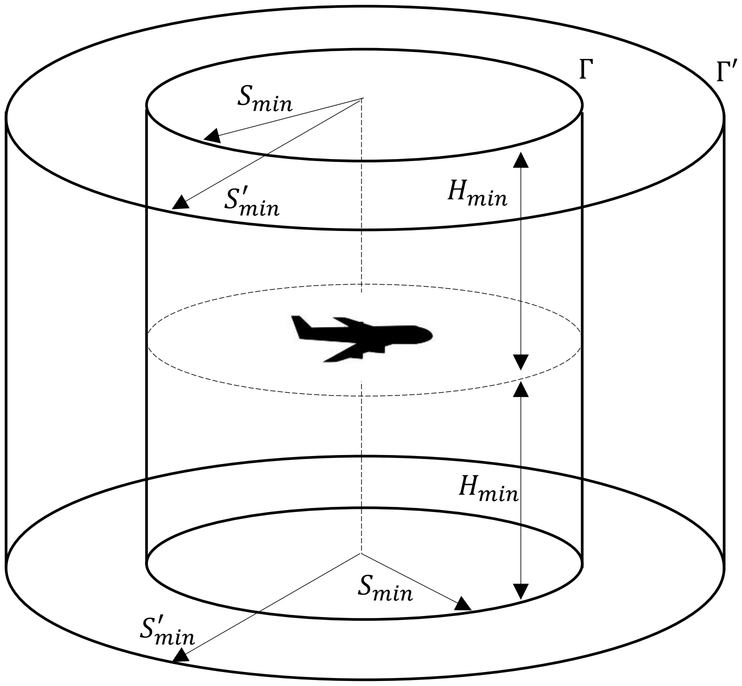


Figure 4: A representation of the minimum separation cylindrical system for the aircraft

Cylinders Γ and Γ' are defined around the selected aircraft A_i, see Figure 4, by which the position of the aircraft A_i is located in the center of the cylinders. The first cylinder Γ is defined by the radius S_{min} and the height $2H_{min}$ if $A_i \in S_0$. The second cylinder Γ' is defined by the radius S'_{min} and the height $2H_{min}$ if $A_i \in S_1$.

For the ATCO tasks defined at this stage of the experiment, a radius S_{min} of 10 NM was taken for the aircraft that are in the controlled airspace and a S'_{min} of 15 NM for the aircraft that are about to enter the controlled airspace. There were two reasons behind this decision, firstly, the air traffic controllers were assessing the air traffic complexity through paper static images and thus did not have the adequate tools for measuring the minimum norm from which they could detect the conflict. Secondly, through ATCOs expert knowledge and the recommended working practices of the Air Traffic Services Operations Manual document, the spotting of the conflict area is prescribed as such.

Before explaining the ATCO tasks, an example of the conflict point T_c is presented in Figure 5, where aircraft positions are represented as points A_i and A_j and distance from conflict point for each aircraft would be lengths d_1 and d_2. Points A'_i and A'_j are points where in future, horizontal separation loss will occur for the first time for the given pair of aircraft. Conflict point T_c is at half length $\frac{S_{min}}{2}$ of the distance between two aircraft when the conflict occurs for the first time. Distance from conflict point are represented as d_1 and d_2 for each aircraft A_i and A_j. Where the distance d_1 is always shorter than the distance of A_i and A_j to the conflict point T_c and the distance d_2 is always greater than the distance of A_i and A_j to the conflict point T_c. Because, A_i aircraft is defined as the closest to the conflict point T_c, while the A_j aircraft is further to the conflict point T_c.

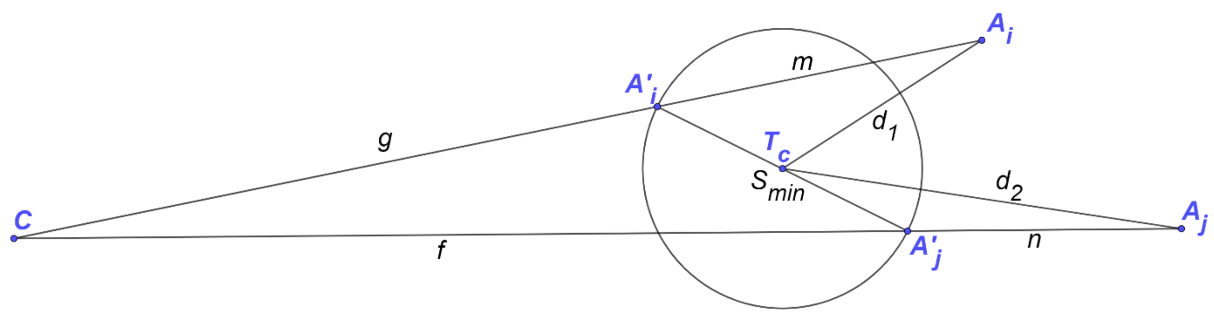


Figure 5: Representation of the conflict point T_c

If the aircraft coordinates, speed and the angle of the aircraft flight vector projected into a plane parallel to the ground are known, it is possible to calculate d_1 and d_2:

$$\measuredangle CA'_iA'_j = arccos\frac{g^2+4\left(\frac{S_{min}}{2}\right)^2-f^2}{4g\left(\frac{S_{min}}{2}\right)} \qquad \measuredangle A_iA'_iT_c = \pi - arccos\frac{g^2+4\left(\frac{S_{min}}{2}\right)^2-f^2}{4g\left(\frac{S_{min}}{2}\right)}$$

$$d_1{}^2 = \left(\frac{S_{min}}{2}\right)^2 + m^2 - 2\left(\frac{S_{min}}{2}\right)m\cos\left(\pi - arccos\frac{g^2+4\left(\frac{S_{min}}{2}\right)^2-f^2}{4g\left(\frac{S_{min}}{2}\right)}\right) = \left(\frac{S_{min}}{2}\right)^2 + m^2 + 2\left(\frac{S_{min}}{2}\right)m\frac{g^2+4\left(\frac{S_{min}}{2}\right)^2-f^2}{4g\left(\frac{S_{min}}{2}\right)} = \left(\frac{S_{min}}{2}\right)^2 + m^2 + m\frac{g^2+4\left(\frac{S_{min}}{2}\right)^2-f^2}{2g}$$

$$\Rightarrow \boxed{d_1 = \sqrt{\left(\frac{S_{min}}{2}\right)^2 + m^2 + m\frac{g^2+4\left(\frac{S_{min}}{2}\right)^2-f^2}{2g}}} \qquad \boxed{d_2 = \sqrt{\left(\frac{S_{min}}{2}\right)^2 + n^2 + n\frac{f^2+4\left(\frac{S_{min}}{2}\right)^2-g^2}{2f}}}$$

Conflict point T_c and distance from conflict point d_1 and d_2 are always observed from the horizontal plane. Since earlier, the forbidden cylinder was defined as a 3D object, it is also needed to describe how the vertical profile for the conflict is defined.

Aircraft are in a state of conflict if their flight trajectories (flight level) intersect, i.e. if their defined volume Γ or Γ' is violated on the trajectories from the current flight level (CUFL) directly to the cleared flight level (CLFL), and after the cleared flight level the aircraft flies in a straight and level flight at the cleared flight level until the last moment when it must start ascending or descending in order to reach the exit flight level (EFL) before exiting the airspace S_0. The trajectories thus defined are called the conflicting trajectories of the aircraft A_i and A_j and are denoted as CoP_i and CoP_j. Potential conflict states are defined if their defined volume Γ or Γ' is violated on flight trajectories going from the current flight level (CUFL) directly to the exit flight level (EFL) and continuing at the level flight directly until exiting airspace S_0, provided that in order for the potential conflicts to occur, the condition must be met that there is no violation of the conflict trajectory. Previously described trajectories are called the potential trajectories of the aircraft A_i and A_j and are denoted as PoP_i and PoP_j.

To determine the correct conflict state, first, there needs to be a horizontal violation S_{min} or S'_{min} for the cylinder Γ or Γ'. For the selected violation, the times *tt1* and *tt2* are defined as the start and end times of the horizontal separation violation. If horizontal separation violation time does not exist, then it means that there will never be a conflict and the situation does not have to be looked any further. If times *tt1* and *tt2* exist, then the described procedure in the previous paragraph is performed, but only between the times *tt1* and *tt2*, provided that the check for conflict trajectories is performed first, and only then potential conflict trajectories.

Figure 6 shows the condition of the two aircraft with respect to vertical separation. Labels are characteristic of the profession; FL is the flight level mark in hundreds of feet measured at a pressure of 1013.25 hPa, e.g. FL 330 is 33000 feet (1000 ft = 304.8 m) shown on the ordinate, at time *t* written on the abscissa. The times *tt1* and *tt2* are the times between which the horizontal separation of the aircraft from S_{min} or S'_{min} is violated, depending on whether $A_i \in S_0$ or $A_i \in S_1$. For altitude changes in trajectory, the notations used for aircraft A_i are > for CoP_i and > ' for PoP_i, while for aircraft A_j the notations used are >> for CoP_j and >> ' for PoP_j.

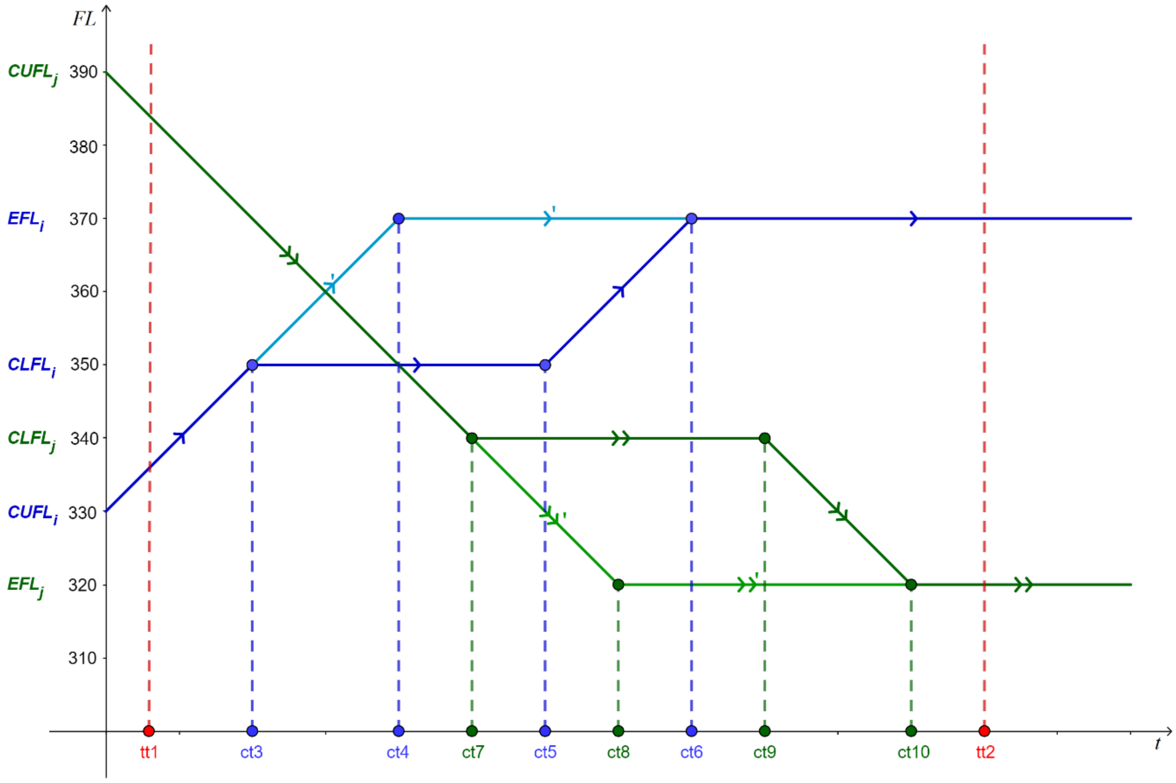


Figure 6: Example of critical times depending on the altitude for a given pair of aircraft

The times *ct3-ct6* are called the critical times of the aircraft A_i, and *ct7-ct10* are called the critical times of the aircraft A_j. The critical times are the critical times of the aircraft A_i together with the critical times A_j. The critical times *ct3* and *ct7* are the times when the aircraft has completed the ascent or descent from the current flight level (CUFL) directly to the cleared flight level (CLFL) of the flight. The times *ct4* and *ct8* are the times when the aircraft completed the ascent or descent from the current flight level (CUFL) directly to the exit flight level (EFL). The times *ct5* and *ct9* are the last times when the aircraft must start ascending or descending in order to reach the exit flight level (EFL) before exiting the S_0 airspace. The times *ct6* and *ct10* are the times when the aircraft reaches the exit altitude exactly upon exiting the S_0 airspace. Time *tt1* is the time when the violation of the horizontal separation between the observed aircraft A_i and A_j has started, and *tt2* is the time when the violation of the horizontal separation stops for the same observed aircraft. The times *tt1* and *tt2* are calculated in advance based on the data of flight speeds, positions and flight directions of the observed aircraft.

An example of determining the exact time of onset of separation loss for a given example from Figure 6 is explained in more detail in the text below. The trajectories marked with > and >> are checked first, and only if there is no separation loss time found, then the trajectories marked with > ' and >> ' are checked.

The first check is done on *tt1* where it checks which aircraft is higher by subtracting the altitudes of the first from the second and determines, if, for example, the value obtained is negative, that the first aircraft is below the second, and in the case of a positive amount - vice versa. Once the altitude positions of the aircraft are determined, it is stored for further verification and it is automatically monitored whether the altitude separation of 1000 ft is impaired.

The next check would be at time *ct3* where the altitude is subtracted from the initially higher aircraft with the initially lower and it is checked if the difference is less than 1000 ft. If altitude separation is permitted and appropriate, the mentioned check continues, which is the time of *ct4* in the shown case. At time *ct4*, it can be seen that, after subtracting the altitude of the initially higher aircraft from the altitude of the initially lower aircraft, the result is less than 1000 ft, i.e. negative, meaning that the separation is violated or it has been violated.

To determine the exact time of the beginning of the separation loss, the last critical time when the vertical separation was all right needs to be looked, which is the time *ct3*, and the critical moment when the separation is disturbed, which is *ct4*. In this interval from *ct3* to *ct4*, it needs to be looked at where the difference in altitude of the initially higher and lower aircraft is equal to 1000 ft and this is the exact time of the beginning of the violation of the vertical separation between the aircraft A_i and A_j. Since the violation of the horizontal separation is determined between *tt1* and *tt2*, and they contain the interval between *ct3* and *ct4*, this is also the exact time of the beginning of the separation violation.

The general procedure for determining the exact time of onset of separation loss is done as follows. The trajectories of aircraft A_i and A_j are selected and the first critical time between *tt1* and *tt2* is sought in which the difference in height of the initially higher and initially lower aircraft is less than 1000 ft for the selected trajectories. Once that time is determined, the interval between the first previous critical time and that critical time is analyzed. In this interval, it needs to be looked at where the difference in height of the initially higher and lower aircraft is equal to 1000 ft and this is the exact time of the beginning of the separation violation between the aircraft A_i and A_j.

The priority list of function execution is as follows. First, it is looked at whether there are times *tt1* and *tt2*. If they do not exist, then it means that there will never be a conflict and the situation does not have to be observed any further. If times *tt1* and *tt2* exist, then the previously described procedure is performed, but only between times *tt1* and *tt2*, with the first check for the trajectories for CoP_i and CoP_j, and only if there is no separation loss time found for the CoP_i and CoP_j trajectories, then the trajectories for PoP_i and PoP_j.

3.3. Air Traffic Controller Tasks

In this chapter a detailed comprehensive definition of the new air traffic controller tasks will be defined. It is important to define these tasks in a way that they are not affected by the air traffic controllers' decisions and resolutions of a specific conflict. This is one of the main factors that will contribute to a good overall feature selection for the model training, because if it is possible to define the tasks not to take into account the unpredictability of the human behavior, for example, air traffic controllers tasks resolutions, they could, overall better evaluate the air traffic complexity.

ATCO tasks are classified as:

1. Conflict resolution (Code: C)
2. Potential conflict resolution (Code: P)
3. Coordination of conflict resolution (Code: CC)
4. Coordination of potential conflict resolution (Code: CP)
5. Interactive conflict screening (Code: SI)
6. Potential interactive conflict screening (Code: SP)
7. Non-interactive conflict screening (Code: SN)
8. Initial call (Code: IC)
9. Frequency transfer (Code: FT)
10. Execution of requests (Code: ER)

The first two tasks have already been explained earlier, and for the third and fourth tasks the same rules apply, with the exception that S'_{min} is applied instead of S_{min} and only for the aircraft that will arrive in controlled airspace S_0. The fifth, sixth and seventh tasks are directly related to the first four. Interactive conflict screening is performed only if there are first and third tasks, potential conflict screening is performed only if there are second and fourth tasks, and non-interactive conflict screening is performed if there is none of the first four tasks The fifth, sixth and seventh tasks give the information about the task that the controller is doing, which is the conflict detection. Three categories are classified depending on the status of the conflict (first four tasks). Although first four tasks and the fifth, sixth and seventh tasks appear to contain the same information, in practice, the controller performs two different tasks for the same thing. They must first spot a conflict as a fifth task and then resolve it as the first or third task to maintain air traffic safety. Additional classification in this way gives a more detailed information for calculating the air traffic complexity. The eight task, the initial call, is performed if the aircraft that is about to enter the controlled airspace S_0 is 20 NM or less from the controlled airspace boundary. The ninth task, frequency transfer, is performed if the aircraft that is in the controlled airspace S_0 is 15 NM or less until exiting that airspace boundary. Execution of request is performed only if the cleared altitude is not equal to the exit altitude of the flight.

First four tasks (C, P, CC and CP) can be subclassified into three categories by their geometrical parameters (Figure 7):

- Same track (Code: S)
- Crossing track (Code: C)
- Opposite track (Code: O)

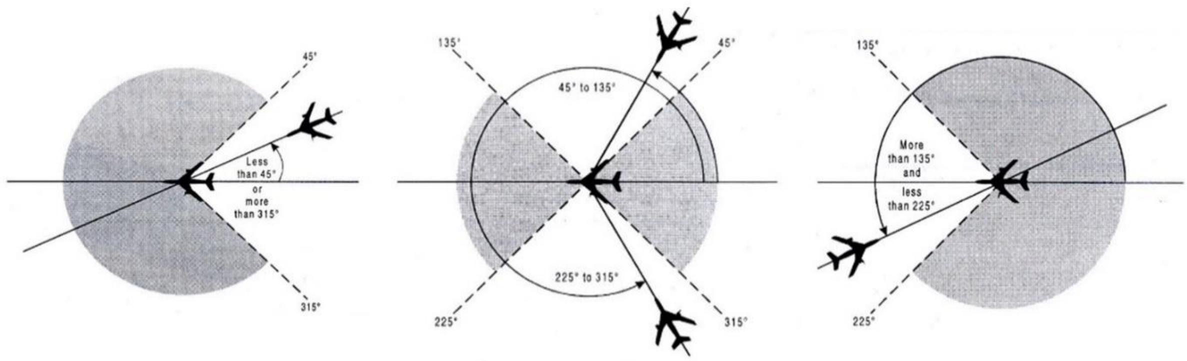


Figure 7: Subclassification of the first four tasks [37]

The Same track is on the left in Figure 7, in the middle is the Crossing track and on the right is the Opposite track. The Same track is executed if the flight angle of two aircraft has the angular difference which is less than 45 degrees or more than 315 degrees. The Crossing track is executed if the flight direction angle of two aircraft has the angular difference which is less than 225 degrees or more than 135 degrees. The Opposite track is executed if the flight direction angle of two aircraft has the angular difference between 45 and 135 degrees and 225 and 315 degrees (Figure 7).

Furthermore, all three previous subclassifications (S, C and O) can be further classified according to their geometric and physical parameters (Table 3):

Table 3: Detailed classification of the main three subclassifications of the tasks

	Code: 1	Code: 2	Code: 3	Code: 4	Code: 5	Code: 6
Distance between aircraft	0-10 NM	11-20 NM	21-30 NM	31-50 NM	51-80 NM	81 or more NM
Speed of aircraft based on the distance to the conflict point	Faster	Same	Slower	/	/	/
Converging to the same point	0^o-20^o	21^o-44^o	45^o-90^o	91^o-135^o	136^o-159^o	160^o-180^o
Distance from the 1st aircraft to the conflict point	0-10 NM	11-20 NM	21-30 NM	31-50 NM	51-80 NM	81 or more NM
Distance from the 2nd aircraft to the conflict point	0-10 NM	11-20 NM	21-30 NM	31-50 NM	51-80 NM	81 or more NM
1st aircraft is free of traffic to its LEFT, 5^o-45^o from the current track	Yes	No	/	/	/	/
1st aircraft is free of traffic to its RIGHT, 5^o-45^o from the current track	Yes	No	/	/	/	/
2nd aircraft is free of traffic to its LEFT, 5^o-45^o from the current track	Yes	No	/	/	/	/
2nd aircraft is free of traffic to its RIGHT, 5^o-45^o from the current track	Yes	No	/	/	/	/
1st aircraft is free of traffic ABOVE	Yes	No	/	/	/	/
1st aircraft is free of traffic BELOW	Yes	No	/	/	/	/
2nd aircraft is free of traffic ABOVE	Yes	No	/	/	/	/
2nd aircraft is free of traffic BELOW	Yes	No	/	/	/	/
Distance to exit for the 1st aircraft	0-15 NM	16-30 NM	31-45 NM	46 or more NM	/	/
Distance to exit for the 2nd aircraft	0-15 NM	16-30 NM	31-45 NM	46 or more NM	/	/

Some of the categories in Table 3 will be explained in more detail. The wake turbulence category between first and second aircraft can receive 3 values; Faster, Same and Slower. Faster if the 1st aircraft belongs to higher turbulence category than the 2nd aircraft. The same if both the first and second aircraft are of the same wake turbulence category and slower if the 1st aircraft belongs to the lower turbulence category than the 2nd aircraft. Converging to the same point has a total of 6 categories, the first two can be classified only under the same track, the third and fourth under the crossing track, while the fifth and sixth categories are classified as the opposite track. This category all together may be excluded, because there can be a conflict between two aircraft that are on a parallel track, not converging to the same point.

Codes for the numerical subclassification (Table 3) are respectively named by numbers starting from 1 for each of their categories. For example, for the category *"Distance between aircraft; 11-20 NM"*, code is number 2, since it is the second value in that category. Furthermore, 1st aircraft is defined as the closest to the conflict point, while 2nd aircraft is further to the conflict point. When looked at all the possible tasks permutations, there are possible $1.927544120276803 \times 10^{27}$ (Octillion) individual tasks.

To make the defined tasks clear, an example of one ATCO task (Figure 9) for a random traffic situation (Figure 8) is presented and explained below.

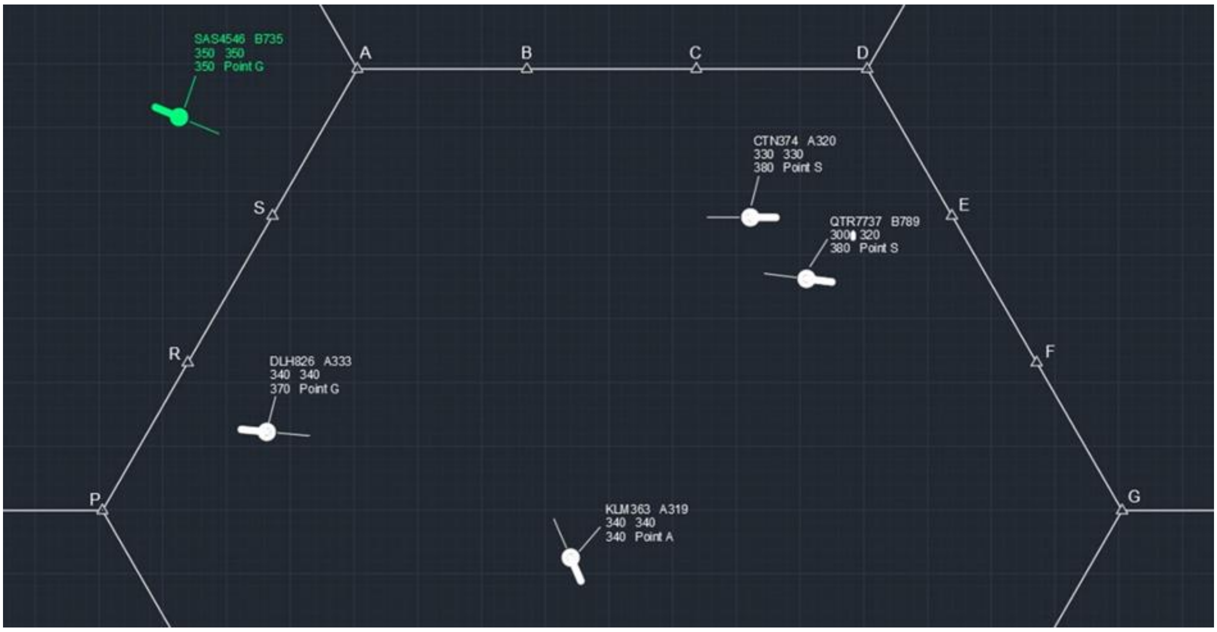


Figure 8: Example of a random traffic situation

Figure 9 represents manually measured, displayed and explained only one task out of a possible 20 tasks defined for the given traffic situation in Figure 8. Later on, tasks will be automatically calculated based on the given coordinates of aircraft and airspace, aircraft speed, aircraft direction of movement, current, cleared and exit flight levels and their intentions.

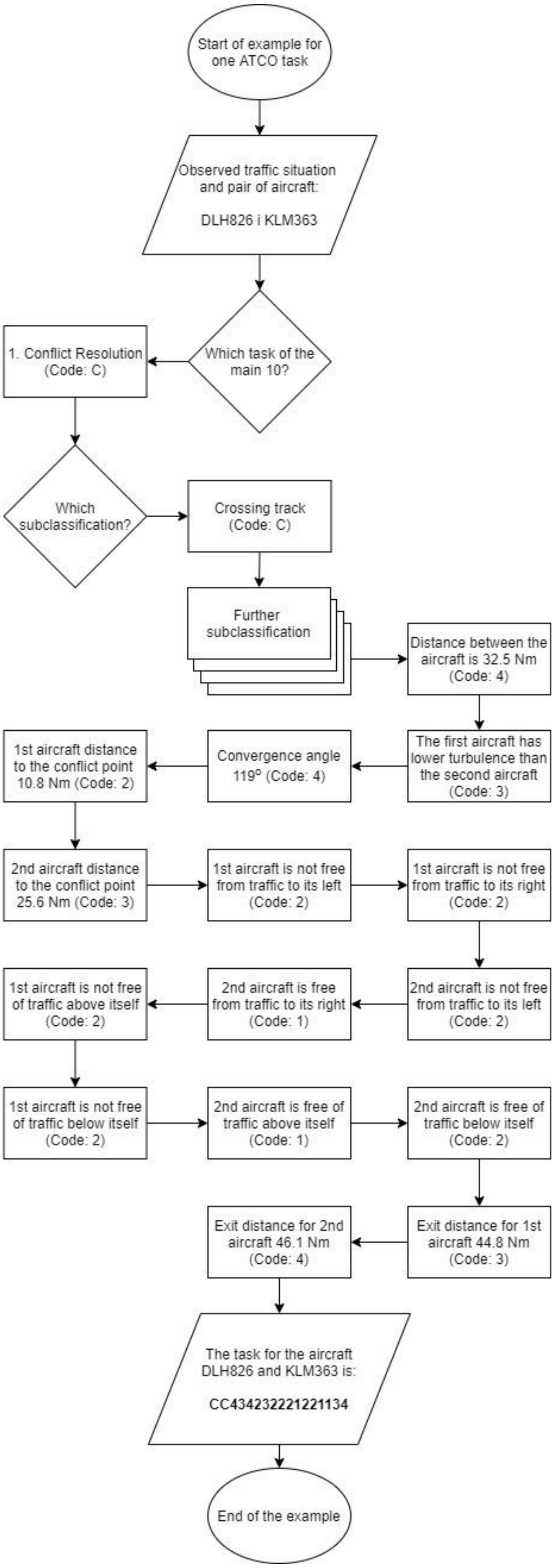


Figure 9: Example of one ATCO task for a given traffic situation from Figure 8

3.4. Automation of ATCO Tasks

Now that the first three phases have been completed, ATCO tasks have to be automated to automatically calculate task codes for any air traffic and any airspace. All that is needed as input variables are: airspace coordinates, aircraft coordinates, aircraft flight direction, aircraft speed, current aircraft altitude, default altitude and output altitude.

The process of task automatization that classifies the status of the selected A_i aircraft in the expanded S_1 airspace given the $N-1$ of other A_j aircraft in the same airspace S_1, expressed as a series of state vectors Δ_{ij}, $i, j \in [1, \ldots, N]$, $i \neq j$, where the expanded air space S_1 is formed by making a homothetic transformation from the centroid of the airspace S_0 for a factor *D* for each point from the airspace S_0. In a preferred embodiment, the parameter *D* has a value of 1.5, and the whole process consists of the following steps:

A. data loading for the selected aircraft A_i and data for each aircraft A_j, $i \neq j$; where the data for an aircraft consist of: aircraft call sign, position (coordinates), speed, angles of aircraft flight direction, current altitude, cleared altitude and exit altitude and loading the observed airspace boundaries S_0 and calculating the airspace boundary of expanded airspace S_1 for the previously specified parameter *D*
B. conducting the classification of the condition of the selected aircraft A_i with respect to the selected aircraft A_j, $i \neq j$, by testing a series of selected binary states $B_z \in \zeta_z = \{0,1\}$ that make up the set Ω where $\psi = \prod_{z=1}^{53} \zeta_z$, $\mathrm{K} \subseteq \{1,2,\ldots,53\}$, $\Omega = \prod_{z \in \mathrm{K}} \zeta_z$ where the states $(B_1, B_2, \ldots, B_{53}) \in \psi$ are defined as follows:
    a. that cylinders Γ and Γ' (Figure 4) are defined around the selected aircraft A_i which places the aircraft A_i at the center of the cylinder, where cylinder Γ is defined by the radius S_{min} and height $2H_{min}$ if $A_i \in S_0$, and Γ' is defined by the radius S'_{min} and height $2H_{min}$ if $A_i \in S_1$ for the next states (Table 4):

Table 4: Binary states for the first four ATCO tasks

B_1 = 1: Conflict resolution (C): $\forall A_j$ having a path that intersects the volume Γ and where $A_j \in S_0$ B_1 = 0: for any other state
B_2 = 1: Potential conflict resolution (P): $\forall A_j$ having a potential path that intersects the volume Γ and where $A_j \in S_0$

B_2 = 0: for any other state
B_3 = 1: Coordination of conflict resolution (CC): $\forall A_j$ having a path that intersects the volume Γ' and where $A_j \in S_1$ B_3 = 0: for any other state
B_4 = 1: Coordination of potential conflict resolution (CP): $\forall A_j$ having a potential path that intersects the volume Γ' and where $A_j \in S_1$ B_4 = 0: for any other state

b. further status classification of the aircraft A_i with respect to other aircraft A_j, $i \neq j$, given the angle ϕ_{ij} between the flight vector of aircraft A_i and aircraft A_j projected onto a plane parallel to the ground (Table 5):

Table 5: Binary states for the main four subclassification tasks

B_5 = 1: same track (S): $-45° < \phi_{ij} < 45°$ B_5 = 0: for any other state
B_6 = 1: opposite track (O): $135° < \phi_{ij} < 225°$ B_6 = 0: for any other state
B_7 = 1: crossing track (C): if $B_5 = 0$ and $B_6 = 0$ B_7 = 0: if $B_5 = 1$ or $B_6 = 1$

c. Further status classification of the aircraft A_i with respect to the other aircraft A_j, $i \neq j$, and the ratio of these aircraft to the observed airspace S_0 and S_1, subject to one or more of the conditions prescribed in below (Tables 6-12):

Table 6: Binary states for the main three subclassification tasks; aircraft distance $d(A_i, A_j)$ in regard to each other

B_8 = 1: aircraft distance $d(A_i, A_j)$ [0-10 NM] B_8 = 0: for any other state
B_9 = 1: aircraft distance $d(A_i, A_j)$ [11-20 NM]

B_9 = 0: for any other state
B_{10} = 1: aircraft distance $d(A_i, A_j)$ [21-30 NM] B_{10} = 0: for any other state
B_{11} = 1: aircraft distance $d(A_i, A_j)$ [31-50 NM] B_{11} = 0: for any other state
B_{12} = 1: aircraft distance $d(A_i, A_j)$ [51-80 NM] B_{12} = 0: for any other state
B_{13} = 1: aircraft distance $d(A_i, A_j)$ [>81 NM] B_{13} = 0: for any other state

Table 7: Binary states for the aircraft speed category v_1 and v_2 for the corresponding aircraft A_i and A_j where v_1 is the speed of the aircraft that is at a shorter distance to the conflict point T_c, and v_2 is the speed of the aircraft that is at a greater distance to the conflict point T_c

B_{14} = 1: $v_1 > \mathrm{v}_2$ B_{14} = 0: for any other state
B_{15} = 1: $\mathrm{v}_1 = \mathrm{v}_2$ B_{15} = 0: for any other state
B_{16} = 1: $\mathrm{v}_1 < \mathrm{v}_2$ B_{16} = 0: for any other state

Table 8: Binary states for the traffic situation with respect to the angle of convergence of the flight path Θ between the flight vectors of aircraft A_i and A_j between aircraft A_i and A_j observed from the point of flight path convergence

B_{17} = 1: angle Θ from interval [0° - 20°] B_{17} = 0: for any other state
B_{18} = 1: angle Θ from interval [21°- 44°]

B_{18} = 0: for any other state
B_{19} = 1: angle Θ from interval [45°- 90°] B_{19} = 0: for any other state
B_{20} = 1: angle Θ from interval [91°-135°] B_{20} = 0: for any other state
B_{21} = 1: angle Θ from interval [136°-159°] B_{21} = 0: for any other state
B_{22} = 1: angle Θ from interval [160°-180°] B_{22} = 0: for any other state

Table 9: Binary states for the main three subclassification tasks; the distance d_1 is shorter than the distance A_i and A_j to the conflict point T_c

B_{23} = 1: distance d_1 is [0-10 NM] B_{23} = 0: for any other state
B_{24} = 1: distance d_1 is [11-20 NM] B_{24} = 0: for any other state
B_{25} = 1: distance d_1 is [21-30 NM] B_{25} = 0: for any other state
B_{26} = 1: distance d_1 is [31-50 NM] B_{26} = 0: for any other state
B_{27} = 1: distance d_1 is [51-80 NM] B_{27} = 0: for any other state
B_{28} = 1: distance d_1 is [>81 NM] B_{28} = 0: for any other state

Table 10: Binary states for the main three subclassification tasks; the distance d_2 is greater than the distance A_i and A_j to the conflict point T_c

B_{29} = 1: distance d_2 is [0-10 NM] B_{29} = 0: for any other state
B_{30} = 1: distance d_2 is [11-20 NM] B_{30} = 0: for any other state
B_{31} = 1: distance d_2 is [21-30 NM] B_{31} = 0: for any other state
B_{32} = 1: distance d_2 is [31-50 NM] B_{32} = 0: for any other state
B_{33} = 1: distance d_2 is [51-80 NM] B_{33} = 0: for any other state
B_{34} = 1: distance d_2 is [>81 NM] B_{34} = 0: for any other state

Table 11: Binary states for the main three subclassification tasks; freedom of movement for the aircraft A_i and A_j

B_{35} = 1: A_i is free from traffic to the left, 5°-45° from the current trajectory B_{35} = 0: A_i is not free from traffic to the left, 5°-45° from the current trajectory
B_{36} = 1: A_i is free from traffic to the right, 5°-45° from the current trajectory B_{36} = 0: A_i is not free from traffic to the right, 5°-45° from the current trajectory
B_{37} = 1: A_j is free from traffic to the left, 5°-45° from the current trajectory B_{37} = 0: A_j is not free from traffic to the left, 5°-45° from the current trajectory
B_{38} = 1: A_j is free from traffic to the right, 5°-45° from the current trajectory B_{38} = 0: A_j is not free from traffic to the right, 5°-45° from the current trajectory

B_{39} = 1: A_i is free from traffic above itself B_{39} = 0: A_i is not free from traffic above itself
B_{40} = 1: A_i is free from traffic below itself B_{40} = 0: A_i is not free from traffic below itself
B_{41} = 1: A_j is free from traffic above itself B_{41} = 0: A_j is not free from traffic above itself
B_{42} = 1: A_j is free from traffic below itself B_{42} = 0: A_j is not free from traffic below itself

Table 12: Binary states for the main three subclassification tasks; the distance d_{ext} of the aircraft A_i or A_j with respect to the exit from the airspace S_0

B_{43} = 1: distance $d_{ext}(A_i, S_0)$ is [0-15 NM] B_{43} = 0: for any other state
B_{44} = 1: distance $d_{ext}(A_i, S_0)$ is [16-30 NM] B_{44} = 0: for any other state
B_{45} = 1: distance $d_{ext}(A_i, S_0)$ is [31-45 NM] B_{45} = 0: for any other state
B_{46} = 1: distance $d_{ext}(A_i, S_0)$ is [> 46 NM] B_{46} = 0: for any other state
B_{47} = 1: distance $d_{ext}(A_j, S_0)$ is [0-15 NM] B_{47} = 0: for any other state
B_{48} = 1: distance $d_{ext}(A_j, S_0)$ is [16-30 NM] B_{48} = 0: for any other state
B_{49} = 1: distance $d_{ext}(A_j, S_0)$ is [31-45 NM] B_{49} = 0: for any other state
B_{50} = 1: distance $d_{ext}(A_j, S_0)$ is [> 46 NM] B_{50} = 0: for any other state

d. where, given the classification in a., Table 4, the states are defined as Table 13:

Table 13: Binary states for the tasks of screening the traffic

B_{51} = 1: interactive conflict screening (SI) if classified B_1 = 1 or B_3 = 1 B_{51} = 0: for every other combination of states B_1 and B_3
B_{52} = 1: Potentially interactive conflict screening (SP) if classified B_2 = 1 or B_4 = 1 B_{52} = 0: for every other combination of states B_2 and B_4
B_{53} = 1: non-interactive conflict screening (SN) when $B_1 = B_2 = B_3 = B_4 = 0$ B_{53} = 0: for every other combination B_1, B_2, B_3, B_4

Thereby obtaining a status record Δ_{ij} from step B. for one selected aircraft A_i with respect to an aircraft A_j and the selected set of states Ω from the overall set of binary states ψ; in the shape of:

$$\Delta_{ij} = \prod_{z \in \mathrm{K}} B_z \tag{1}$$

C. for the states when $i = j$, which refer to the observed aircraft A_i, they are further classified by a series of binary states (C_1, C_2, C_3) as shown in Table 14:

Table 14: Binary states for the tasks of initial call, frequency transfer and execution of request

C_1 = 1: Initial call (IC) for $A_i \in S_1$, where A_i is to enter S_0 and is distant by the amount of R_{in} or less than the S_0 border C_1 = 0: for any other situation
C_2 = 1: Frequency transfer (FT) for $A_i \in S_0$, where A_i is to leave S_0 and is distant by the amount of R_{out} or less than the S_0 border C_2 = 0: for every other situation
C_3 = 1: Execute of request (ER) if the cleared flight altitude of the aircraft A_i is not equal to the exit flight altitude

C_3 = 0: for every other situation

Providing a status record of ordered triple Δ_{ii} in the step C. for one selected aircraft A_i in the form:

$$\Delta_{ii} \;=\; (C_1, C_2, C_3)_{ii} \tag{2}$$

D. Repetitions of the step B. for all values of j to complete the series of state vectors $\Delta_{ij}\forall j$ with respect to the default value i.

The values used for the variables are: S_{min} = 10 NM, S'_{min} = 15 NM, H_{min} = 1000 ft, R_{out} = 15 NM, R_{in} = 20 NM, and $D = 1.5$.

Now that the binary states are carefully defined separately for each traffic situation, w_l a $N^l\ x\ N^l$ matrix with states Δ_{ij} for $i, j \in \{1, \dots, N^l\}$ is calculated, where N^l is the number of aircraft participating in each of the traffic situations w_l. Since the example of a binary states would not give much information to the common reader, an example of a matrix with ATCO tasks will be presented using the codes earlier defined in Chapter 3.3. How one random matrix looks like with a random traffic situation with only three aircraft is presented below:

$$\begin{bmatrix} 0 & CTN4847 & QTR7737 & KLM363 \\ CTN4847 & \{ER, Null, IC\} & SN & SN \\ QTR7737 & SN & \{Null, Null, Null\} & CC433332222212244, SI \\ KLM363 & SN & CC433332222222144, SI & \{ER, FT, Null\} \end{bmatrix}$$

With clear definition of the automation process, the Wolfram Mathematica 11 program will be used to code the functions that will calculate all the necessary tasks for the given traffic situation and print it in a matrix form (Appendix 3). A side note to the reader, indexing in the Wolfram code is different from the variables described earlier. This is done to simplify the code.

Program for the automation of the tasks is separated into several smaller functions that calculate specific parameters for certain aircraft related to the defined tasks form 3.1 of this chapter. The automation program has a total of 16 subfunctions (*f1, f2, f3, f4, typeofconflict, fdistance, fintersectionangle, fintersectionpoint, finsidetest, finsidetest2, fconflictpoint, fdistancetoboundary, fdistanceclosertoboundary, fwtc, fthorizontanefree, ftverticalfree*) that calculate the default task parameters before being called to the main program. Now, a brief overview of the functions will be explained.

The main automation program (Appendix 4) invokes the four predefined functions that are labeled as *f1*, *f2*, *f3*, and *f4*. It pulls the information from those four predefined functions and fills the matrix with tasks for a specific pair of aircraft and diagonally assigning the task to the specific aircraft.

The *f1* function (Appendix 4) calculates the distance of the aircraft to the boundary by invoking the predefined *fdistancetoboundary* function, the execution of requests (*ER*), frequency transfer (*FT*), and the aircraft initial call (*IC*) tasks.

The *f2* function (Appendix 4) only calculates if there is a conflict within the expanded airspace of the following parameters; distance between aircraft (*d0*), converging angle (*convergingangle*), distance from 1st aircraft to conflict point (*dcp1*), distance from 2nd aircraft to conflict point (*dcp2*), category of wake turbulence between two aircraft (*wtc*), conflict track, that is a subclassification of the first four tasks (*conflicttrack*) and a type of conflict (*typeofconflict*) to be able to classify the tasks of detecting conflict screening (*cs*).

The *f3* function (Appendix 4) calculates whether the aircraft is free horizontally (*tcwfree, tccwfree*) and vertically, (*tupfree, tdownfree*) by invoking the functions *fthorizontalfree* and *ftverticalfree*.

The *f4* function (Appendix 4) fills the task codes into the matrix for each pair of aircraft.

The *typeofconflict* function (Appendix 4) calculates the conflict time (*conflicttime*), the exact coordinates of the conflict with respect to horizontal and vertical distance (*trueconflictpoint*), and the type of conflict for the first four tasks (*conflictype*).

The *fdistance* function (Appendix 4) calculates the horizontal distance between a pair of aircraft.

The *fintersectionangle* function (Appendix 4) calculates the convergence angle between two intersecting aircraft paths.

The *fintersectionpoint* function (Appendix 4) calculates the point where the directions of the aircraft flight path intersect, but only horizontally.

The *finsidetest* function (Appendix 4) tests whether the observed aircraft is within controlled airspace.

The *finsidetest2* function (Appendix 4) tests whether the observed aircraft is within the expanded airspace.

The *fconflictpoint* function (Appendix 4) calculates the point where the separation of aircraft will be violated, but only horizontally.

The *fdistancetoboundary* function (Appendix 4) calculates the distance of the aircraft from the further airspace boundary.

The *fdistanceclosertoboundary* function (Appendix 4) calculates the distance of the aircraft to the closer airspace boundary.

The *fwtc* function (Appendix 4) calculates the wake turbulence category, which aircraft is faster, and which one is slower in relation to the distance from the conflict point.

The *fthorizontalfree* function (Appendix 4) calculates whether the aircraft is free of traffic if it turns right (clockwise) for 5^o – 45^o from its current flight path, in 5-degree steps (*thorizontalfreecw*) and left (anticlockwise) with the same turn-degree rule (*thorizontalfreeccw*).

The *ftverticalfree* function (Appendix 4) calculates whether there is a vertical and horizontal separation loss for a pair of aircraft.

The appendices explain the functions of task automation clearly and in more detail.

3.5. Data gathering

It is vital to state the process of gathering the data for the experiment, since it is one of the factors that contributed to the good results in the model. First, it is important that the air traffic controllers assess the air traffic complexity on the paper static images since this gives them more time to think about the actual traffic complexity instead of measuring the workload by mistake. Secondly, it is important to always give the tested air traffic controller a set of two traffic situations for complexity assessment. In this way, air traffic controller is forced to choose which of the two presented traffic situations is more complex. During the experiment, Merge sort algorithm is used for ranking the traffic. Merge sort is a divide and conquer algorithm that was invented by John von Neumann in 1945. In computer science, merge sort is an efficient, general-purpose, comparison-based sorting algorithm. Merge sort repeatedly splits a list of data into several sublists until each sublist consists of a single component (Figure 10). Afterwards, it joins these sublists into a sorted list. Here, the input from the air traffic controller was taken as a sort list rule (Appendix 5).

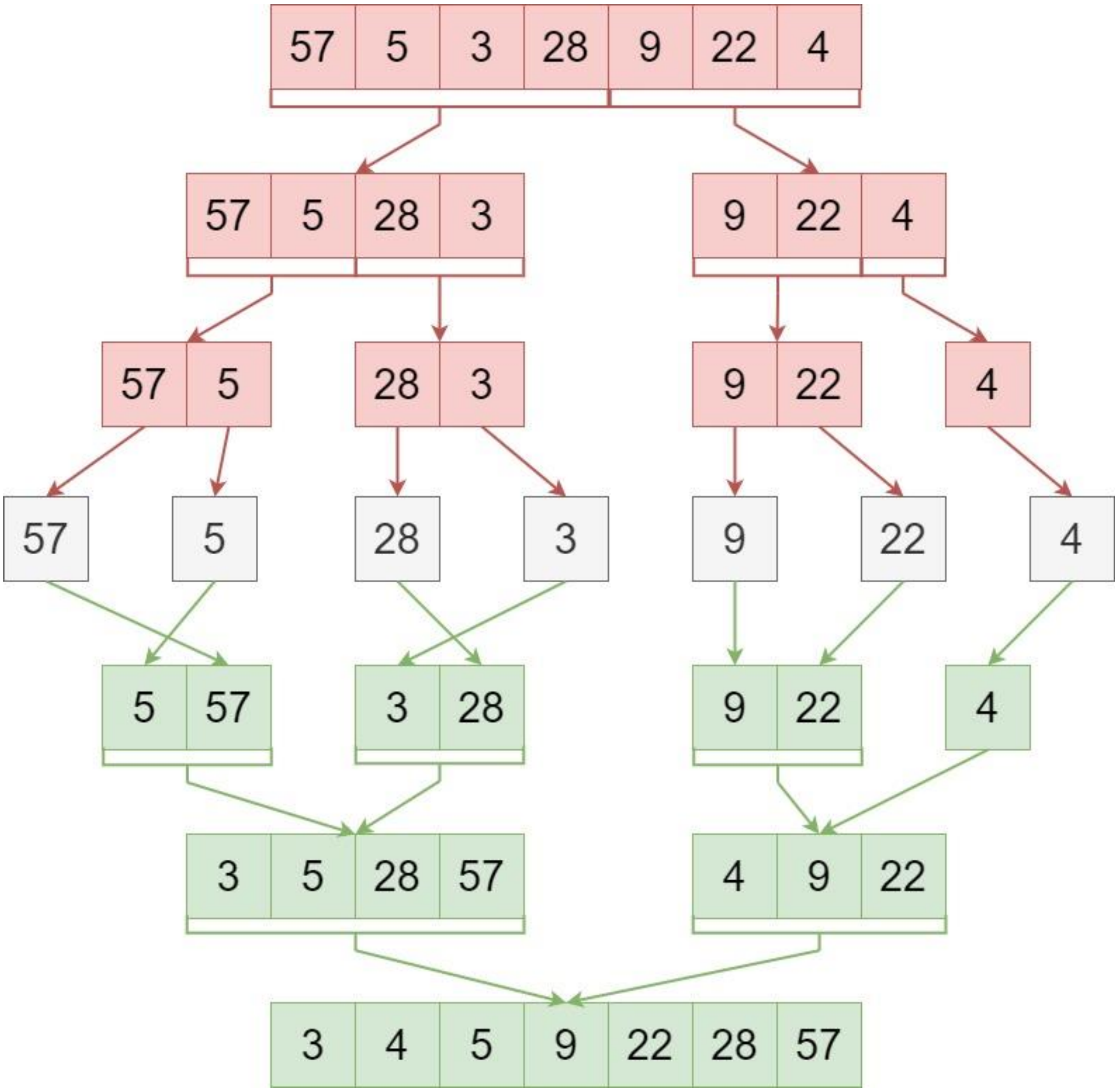


Figure 10: Example of Top-down Merge sort

By ranking them in this manner, it is impossible that two traffic situations have the same complexity score. Using this approach, any inconsistent assessment by the air traffic controllers is eliminated, so that at the end of the validations, there is a clear rank from the lowest w^1 to the highest w^M air traffic complexity $\{w^1, w^2, w^3, \dots, w^M\}$. Furthermore, to establish a clear grading system, controllers need to grade the ranked traffic situations into classes of complexity scores, from 1 to 5, where 1 is the lowest and 5 is the highest complexity score. Based on the controllers' complexity ranking and scoring, linearly interpolated grades η^l are assigned to each ranked traffic situation w^l for each controller (Table 15). As a final step, controllers were asked to determine where the point would be when they believed that the sector had to be divided to reduce the air traffic complexity.

Table 15: Example of the data gathering from the experiment with one of the air traffic controllers

<table>
<tr><td colspan="2">Date and time of the experiments:
10.04.2019. / 10:00 h - 12:00 h</td><td colspan="2">Candidate:
Example no. 1</td><td colspan="2">Years of experience:
23</td></tr>
<tr><td colspan="3">Time required to rank the traffic:
02h00m (10-min break)</td><td colspan="3">Traffic sample taken and group:
A1-C10 / G1</td></tr>
<tr><td colspan="6">Merge sort candidate's answers:</td></tr>
</table>

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A5 =>A4
4. What is more complex: A6 or A7 =>A7
5. What is more complex: A5 or A6 =>A5
6. What is more complex: A5 or A7 =>A7
7. What is more complex: A4 or A7 =>A4
8. What is more complex: A1 or A6 =>A6
9. What is more complex: A2 or A6 =>A6
10. What is more complex: A3 or A6 =>A6
11. What is more complex: A8 or A9 =>A9
12. What is more complex: A10 or B1 =>A10

26. What is more complex: A9 or B4 =>A9
27. What is more complex: A9 or B2 =>A9
28. What is more complex: A1 or B5 =>A1
29. What is more complex: A1 or B3 =>A1
30. What is more complex: A1 or B1 =>B1
31. What is more complex: A2 or B1 =>A2
32. What is more complex: A2 or A8 =>A2
33. What is more complex: A2 or A10 =>A2
34. What is more complex: A2 or B4 =>B4
35. What is more complex: A3 or B4 =>B4
36. What is more complex: A6 or B4 =>A6
37. What is more complex: A6 or B2 =>A6

51. What is more complex: B6 or C1 =>C1
52. What is more complex: B8 or C1 =>C1
53. What is more complex: C3 or C4 =>C4
54. What is more complex: C5 or C6 =>C6
55. What is more complex: C3 or C5 =>C3
56. What is more complex: C3 or C6 =>C3
57. What is more complex: C7 or C8 =>C7
58. What is more complex: C9 or C10 =>C10
59. What is more complex: C8 or C9 =>C9
60. What is more complex: C7 or C9 =>C9
61. What is more complex: C5 or C8 =>C5
62. What is more complex: C5 or C7 =>C5

76. What is more complex: C2 or C5 =>C2
77. What is more complex: C2 or C6 =>C2
78. What is more complex: C2 or C3 =>C3
79. What is more complex: B10 or C3 =>B10
80. What is more complex: B10 or C10 =>C10
81. What is more complex: B5 or B7 =>B7
82. What is more complex: B3 or B7 =>B7
83. What is more complex: A1 or B7 =>B7
84. What is more complex: B1 or B7 =>B7
85. What is more complex: A8 or B7 =>B7
86. What is more complex: A10 or B7 =>B7
87. What is more complex: A2 or B7 =>A2

13. What is more complex: A8 or B1 =>A8

14. What is more complex: A8 or A10 =>A10

15. What is more complex: A9 or A10 =>A9

16. What is more complex: B2 or B3 =>B2

17. What is more complex: B4 or B5 =>B4

18. What is more complex: B3 or B5 =>B3

19. What is more complex: B3 or B4 =>B4

20. What is more complex: B2 or B4 =>B2

21. What is more complex: B1 or B5 =>B1

22. What is more complex: B1 or B3 =>B1

23. What is more complex: B1 or B4 =>B4

24. What is more complex: A8 or B4 =>B4

25. What is more complex: A10 or B4 =>B4

38. What is more complex: A6 or A9 =>A9

39. What is more complex: A5 or A9 =>A9

40. What is more complex: A7 or A9 =>A9

41. What is more complex: A4 or A9 =>A4

42. What is more complex: B7 or B8 =>B8

43. What is more complex: B6 or B7 =>B6

44. What is more complex: B6 or B8 =>B8

45. What is more complex: B9 or B10 =>B10

46. What is more complex: C1 or C2 =>C2

47. What is more complex: B9 or C1 =>B9

48. What is more complex: B9 or C2 =>C2

49. What is more complex: B10 or C2 =>B10

50. What is more complex: B7 or C1 =>C1

63. What is more complex: C5 or C9 =>C5

64. What is more complex: C5 or C10 =>C10

65. What is more complex: C6 or C10 =>C10

66. What is more complex: C3 or C10 =>C10

67. What is more complex: C4 or C10 =>C4

68. What is more complex: B7 or C8 =>C8

69. What is more complex: B6 or C8 =>C8

70. What is more complex: B8 or C8 =>C8

71. What is more complex: C1 or C8 =>C8

72. What is more complex: B9 or C8 =>C8

73. What is more complex: C2 or C8 =>C2

74. What is more complex: C2 or C7 =>C2

75. What is more complex: C2 or C9 =>C2

88. What is more complex: A2 or B6 =>B6

89. What is more complex: A3 or B6 =>B6

90. What is more complex: B4 or B6 =>B6

91. What is more complex: B2 or B6 =>B6

92. What is more complex: A6 or B6 =>B6

93. What is more complex: A5 or B6 =>B6

94. What is more complex: A7 or B6 =>B6

95. What is more complex: A9 or B6 =>A9

96. What is more complex: A9 or B8 =>A9

97. What is more complex: A9 or C1 =>C1

98. What is more complex: A4 or C1 =>A4

99. What is more complex: A4 or B9 =>B9

Ranking results:

['B5', 'B3', 'A1', 'B1', **1** 'A8', 'A10', 'B7', 'A2', **2** 'A3', 'B4', 'B2', 'A6', 'A5', 'A7', **3** 'B6', 'B8', 'A9', 'C1', 'A4', 'B9', 'C8', 'C7', 'C9', 'C5', 'C6', **4** 'C2', 'C3', 'B10', 'C10', 'C4' **5**]

Linearly interpolated scores:

1.	B5=0.25	(0+1/4)
2.	B3=0.5	(0+2/4)
3.	A1=0.75	(0+3/4)
4.	B1=1	(0+4/4)
16.	B8=3.181818	(3+2/11)
17.	A9=3.272727	(3+3/11)
18.	C1=3.363636	(3+4/11)
19.	A4=3.454546	(3+5/11)

5.	A8=1.25	(1+1/4)	20.	B9=3.545455	(3+6/11)
6.	A10=1.5	(1+2/4)	21.	C8=3.636364	(3+7/11)
7.	B7=1.75	(1+3/4)	22.	C7=3.727273	(3+8/11)
8.	A2=2	(1+4/4)	23.	C9=3.818182	(3+9/11)
9.	A3=2.166667	(2+1/6)	24.	C5=3.909091	(3+10/11)
10.	B4=2.333333	(2+2/6)	25.	C6=4	(3+11/11)
11.	B2=2.5	(2+3/6)	26.	C2=4.2	(4+1/5)
12.	A6=2.666667	(2+4/6)	27.	C3=4.4	(4+2/5)
13.	A5=2.833333	(2+5/6)	28.	B10=4.6	(4+3/5)
14.	A7=3	(2+6/6)	29.	C10=4.8	(4+4/5)
15.	B6=3.0909091	(3+1/11)	30.	C4=5	(4+5/5)

Comment form the candidate:	**Observations from the moderator:**
Does not need to open a sector, and candidate stated that all the traffic situation seemed easy.	Candidate did not use the ruler.

Validation airspace Merge sort candidates answers:			
1.	What is more complex: V2 or V3 =>V2	5.	What is more complex: V4 or V6 =>V6
2.	What is more complex: V1 or V3 =>V3	6.	What is more complex: V1 or V5 =>V5
3.	What is more complex: V5 or V6 =>V6	7.	What is more complex: V3 or V5 =>V5
4.	What is more complex: V4 or V5 =>V4	8.	What is more complex: V2 or V5 =>V5

Validation Ranking results:	**Validation Linearly interpolated scores:**			
['V1', **1** 'V3', 'V2', **2** 'V5', **3** 'V4', 'V6' **4**]	1. V1=1		4.	V5=3
	2. V3=1.5		5.	V4=3.5
	3. V2=2		6.	V6=4

Here, it can be seen how the controllers graded the ranked traffic situations into classes of complexity scores. Some air traffic controllers chose not to put all five complexity scores, for instance, some chose to skip the first complexity score and start with 2 and others skipped other complexity scores. Detailed data gathering and complexity ranking and scoring can be seen for all the participants in the Appendix 6.

4. Model development

In this stage of research, a statistical model is learned from the set of the chosen exploratory variables Ω and the target variables η^l. The goal of the statistical model is to give, in statistical terms, relationship between exploratory variables which describe an air traffic situation and the complexity of the situation itself. In this way, the statistical model can generalize to unseen air traffic situations.

In principle, any suitable statistical model can be used. In this experiment a regularized version of linear regression - Bayesian Ridge Regression was used, but other, more sophisticated models can be used also, as well as nonlinear models. However, an advantage of using a linear regression model is that there is a direct relationship between regression model's coefficients and the contribution that each individual exploratory variable (ATCO tasks) contributes to the air traffic complexity. Therefore, the coefficients themselves can be used as a measure of ATCO task complexity.

Statistical model, regardless of whether it is a linear or nonlinear, can be used to calculate the contribution of individual aircraft to the air traffic complexity. A procedure of how to do this for one particular aircraft is given. First, statistical model of air traffic complexity is trained in a usual way, by building a model based on sample data, known as "training data". Second, a copy of the traffic situation where this particular aircraft appears is created, with the sole distinction that the copy does not contain that particular aircraft. The statistical model trained in the previous step is then applied on the original air traffic situation and on the copy where the particular aircraft is missing, and the difference in complexity between the two situations is a measure of aircraft's contribution to the overall situation complexity. This procedure is then repeated with every aircraft in the situation, and the aircraft whose removal contributes to the highest reduction in situation's complexity is considered the most complex. The statistical model constructed in the first step can be reused in these subsequent evaluations of each aircraft's complexity. This procedure does not depend on the specific form of statistical model - any suitable statistical model can be used, both linear and nonlinear.

Moreover, once a statistical model of air traffic complexity is finished, other variables can be calculated which are interesting to the air traffic controllers. For example, by using the controller's estimates at which particular air traffic situation the complexity warrants opening the new sector (so that the complexity of the two new sectors is lower than in their combination), a statistical model can be build which predicts this decision automatically. To do this, any suitable statistical classification model can be used, both linear and nonlinear, while using the same set of exploratory variables (ATCO tasks) which allows the statistical model to generalize to the unseen air traffic situations.

4.1. Exploratory feature analysis

In statistics, exploratory data analysis (EDA) is a method of examining data sets to summarize their key features, usually with the visual approach. A statistical model can be used, but primarily, EDA is used for ascertaining the information from the data beyond the formal modeling or hypothesis testing task. EDA is different from initial data analysis (IDA), which emphasizes more on the assumptions required for proving model fitting and hypothesis testing, controlling missing values and creating transformations of variables as needed [38].

4.1.1. Histograms of feature value distributions

In order to see whether there are any inconsistencies in the data collection process, the histograms of feature value distributions for all 120 traffic situations is plotted in Figure 11.

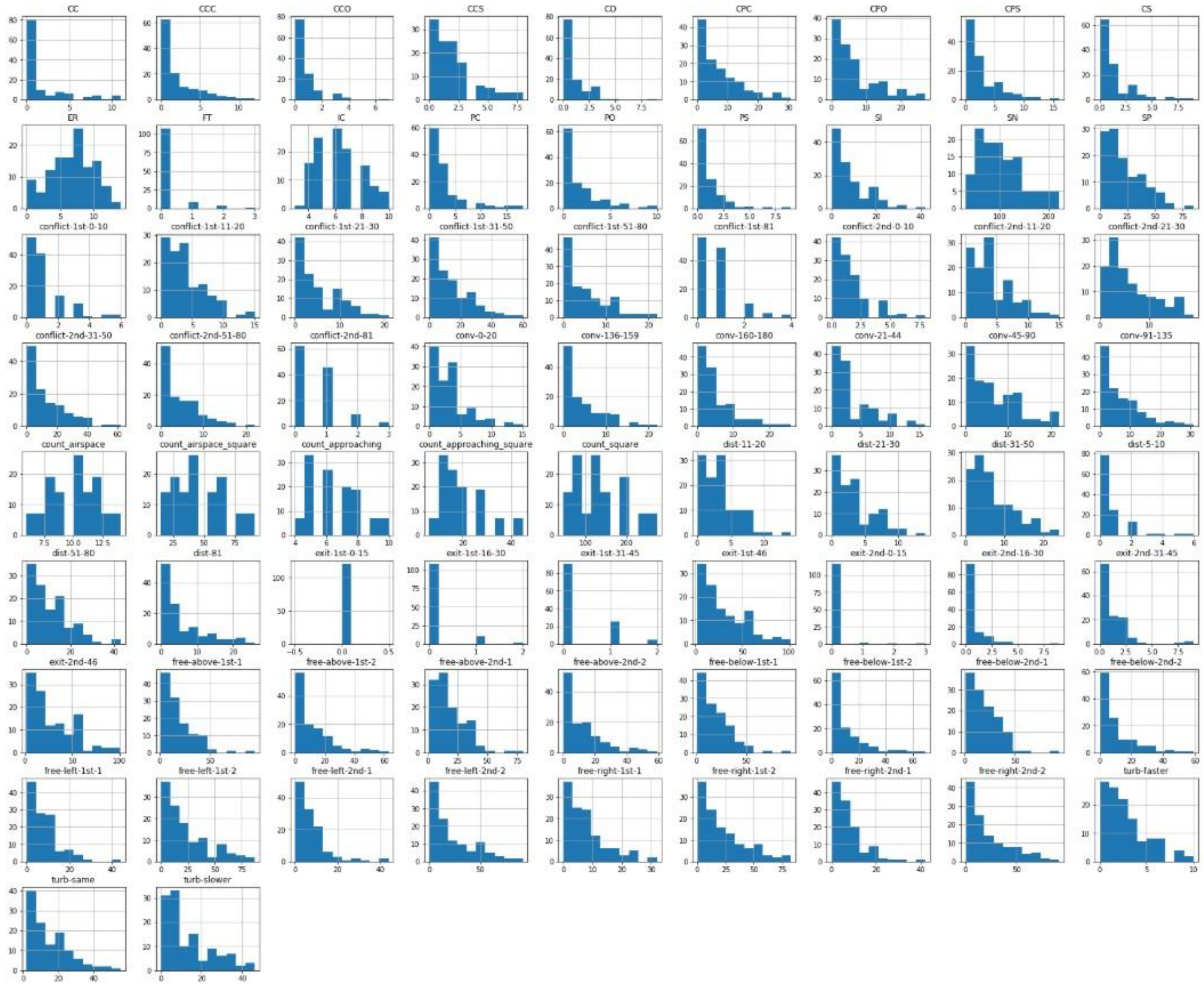


Figure 11: Histograms of feature value distributions

It can be seen from the histograms of feature value distributions (Figure 11) that almost all of the ATCO tasks were represented to the same extent. However, it can be seen that there were some exceptions. For instance, B_{43} had zero occurrences through 120 traffic situations, while some tasks, such as B_{51} – B_{53} were represented in more than half of the original data set. Air traffic situations are presented on the ordinate of the histogram and the number of occurrences is presented on the abscissa.

4.1.2. Correlation matrix between features

It was previously established that this model uses a large number of exploratory variables. In fact, the version six of the model uses 74 independent features. It is always good to check if there are any correlations between the features themselves, as this can give a better insight to what features are good for the overall model and maybe it can lead to new conclusion regarding the feature correlation with the air traffic situations as well.

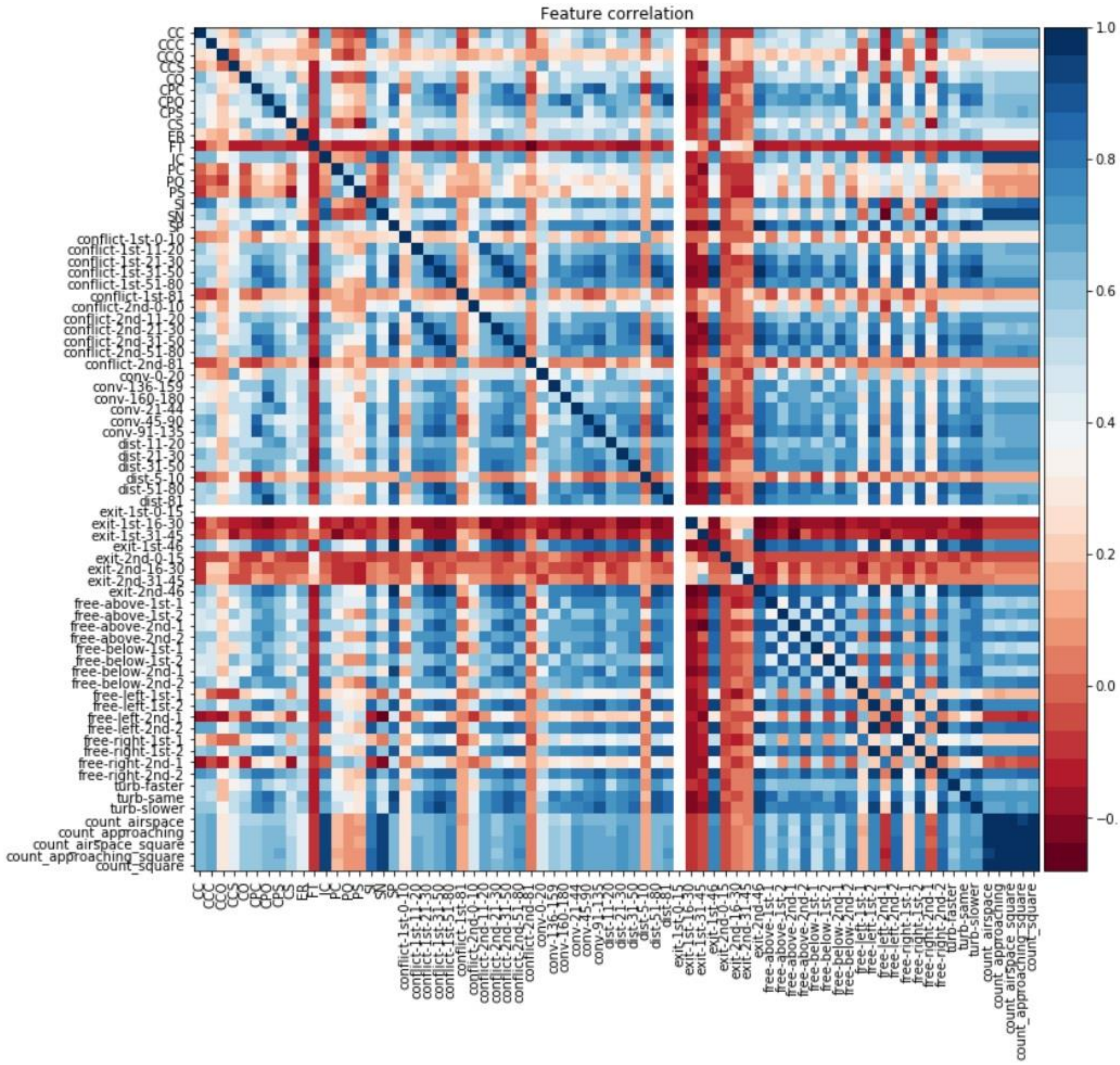


Figure 12: Correlation matrix between features

The correlation matrix between features (Figure 12) shows whether there are any significant correlations between the variables. Some features correlate as high as 0.8, and there are some weak negative correlations as well. There are some obvious high correlations between the screening of the aircraft and the number of aircraft. Nevertheless, that was expected since both contain the same information within themselves, and that is the number of aircraft. However, several interesting ones, like the conflict between first and second aircraft regarding the distance to conflict point, correlate pretty good.

4.1.3. PCA analysis

The principal component analysis (PCA) is performed on the original feature matrix (Figure 13). PCA transforms the data (technically, it performs rotation of the data vectors) so that the complete information is preserved. The resulting data is of the same dimensionality as the original, but the principal components (which correspond to features in the original data) are ordered so that the first principal component carries the largest amount of linear variance. If the features in the original dataset are very correlated, then the majority of this redundant information will concentrate in the first few principal components, with subsequent components carrying less and less information. It is already known, from the correlation matrix, that many features in the dataset are correlated. In fact, first principal component explains around 0.55 of variance, while it takes around first 8 components to explain 0.8 of variance.

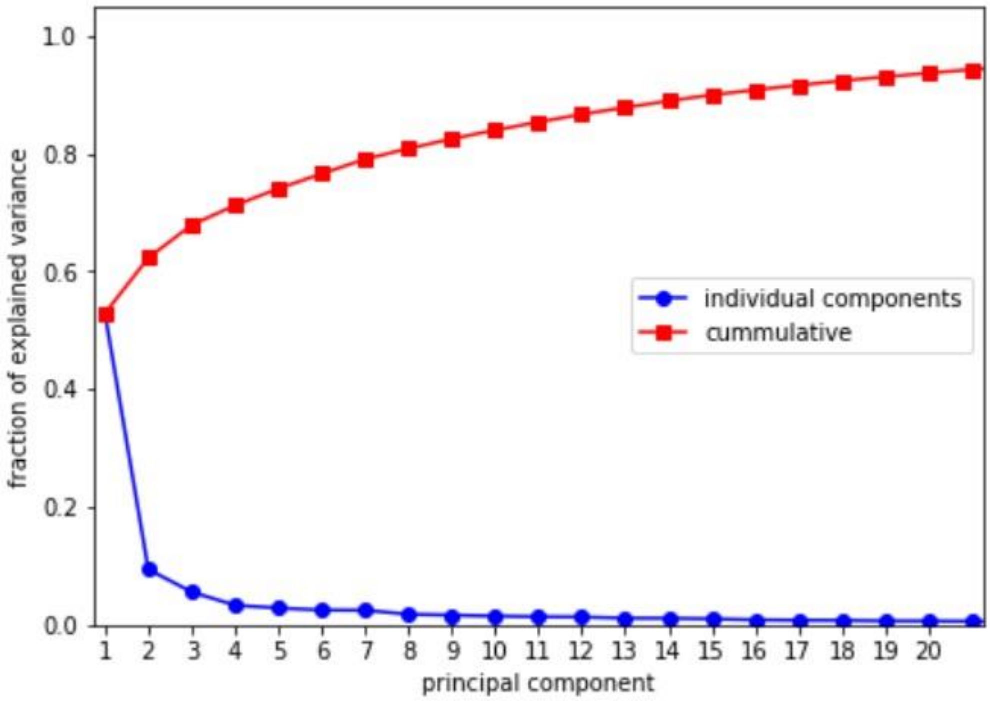


Figure 13: PCA analysis on the original feature matrix

Possible future research is to perform a more detailed PCA or even non-negative matrix factorization (NMF) analysis and see whether loadings for features in the first few components have a semantic interpretation. For example, whether first component captures information related to the aircraft count in direct or indirect way. It is already known that *SN* (non-interactive screening) and aircraft count features are highly correlated, and both are consistently highly correlated with the target variable, so maybe their information is somehow captured in the first principal component.

4.1.4. Univariate feature correlation with the target variable

The best performing features can achieve correlation of around 0.8, which is already significantly high, considering that the statistical model achieves correlation of around 0.85 with the target variable.

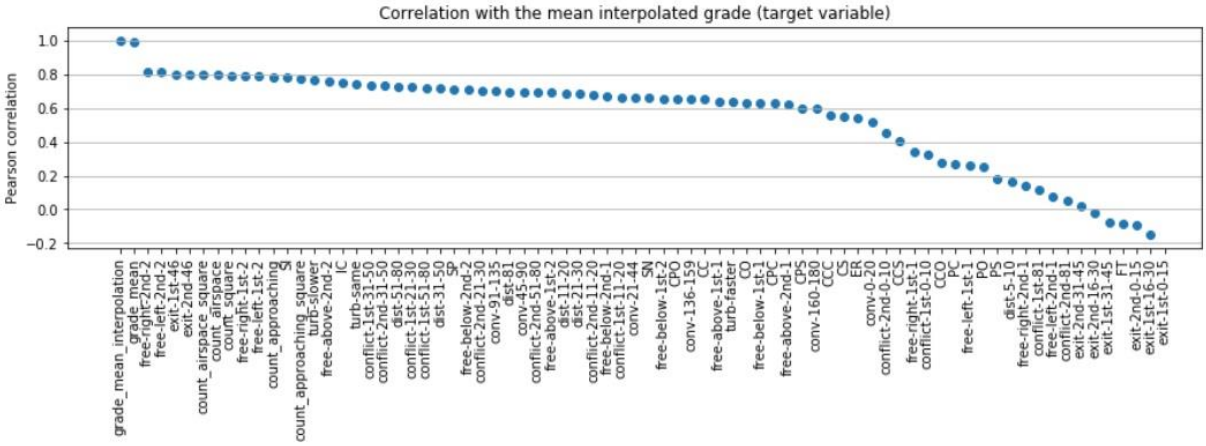


Figure 14: Correlation of the ATCO tasks with the mean interpolated grades

It can be seen from Figure 14 that the freedom of movement for the second aircraft that is in conflict with the first aircraft has a high correlation with the complexity scores that the air traffic controllers gave. That gives insight to the nature of the complexity for the performing task of the air traffic controller. A simple task, such as conflict resolution, can have multiple complexity levels that can be identified.

4.2. Feature construction

The first thing that needs to be done is to determine the feature selection for the model. Each traffic situation consists of several aircraft and their mutual positions. ATCO tasks correspond to the tasks that controller needs to do in order to assess the situation or resolve any conflicts. These are supposed to be independent of individual controllers - all controllers are aware of the same set of tasks, although their way of resolving them might be different. Each feature is effectively a number of times that a particular task appears in a given situation. Following set of Ω features for the model training are taken:

B_1 – B_4 combined with B_5 – B_7 *(CCC, CCO, CCS, CPC, CPO, CPS, CC, CO, CS, PC, PS* and *PO)*, B_{51} – B_{53} *(SI, SN, SP)* and C_1 – C_3 *(ER, FT, IC)* results in the combination of 18 tasks types. Then each individual task from B_8 to B_{50}, makes up to total of 69 individual tasks, because the tasks from the B_{35} to B_{42} are doubled due to the binary output of 0 or 1. Also, the aircraft count is taken into account as one of the features. N^{in} is a number of aircraft that are about to enter S_0 airspace, and N^{out} is a number of aircraft that are about to exit S_0 airspace. Furthermore, number of all the possible pairs of aircraft are calculated to capture the fact that the number of pairwise screenings which have to be performed scales in non-linear way with the number of aircraft. An exact formula for the number of aircraft pairs is:

$$\frac{N(N-1)}{2}$$

Where N is the number of aircraft. So, in total, after adding all the possible aircraft count permutations to the feature selection, there are 74 features for the training of the model.

4.3. Target variables

Two different sets are going to be used for the target variables. First set, for a pairwise comparison modeling, results of pairwise comparisons (merge sort results) will be used. The outcomes can be defined as binary states (Table 16), so this effectively turns into a binary decision problem which can be solved with a classifier. In the preliminary analysis for this approach a logistic regression was used. There are in total, 1757 comparison results and due to a large number of data just a sample of three comparisons are presented in Table 16. The rest of the comparison results can be found in the Appendix 6 in the section Merge sort results.

Table 16: Example of the binary target variables from the pairwise comparison of the traffic situation.

Candidate number	Comparison situation 1	Comparison situation 2	Comparison result
3	A2	A3	0
3	A1	A3	1
3	A4	A5	0

For the second set, a controller's grades were used which come in two versions (Figure 15):

1. Original discrete grades on the scale from 1 to 5
2. Linearly interpolated grades on the scale from 1 to 5, based on how controllers ranked the situations within each grade.

Although there are in total 540 controllers' scores, in order to obtain a single grade for each situation, regardless of how many controllers graded it, only the mean value has been taken. 12 situations (A1-A4, B1-B4, and C1-C4) are graded by all 18 controllers, while all other situations are graded by only 3 controllers. From the analysis of the results, there is not that much difference in predictive performance when using target variables of the mean grades obtained in the first or the second way. But clearly it can be seen from Figure 15 that the interpolated scores contain and carry more information than the flat mean value of the whole score. There is much more information embedded within the interpolated scores which makes them a better choice to be used as target variables for the linear regression models.

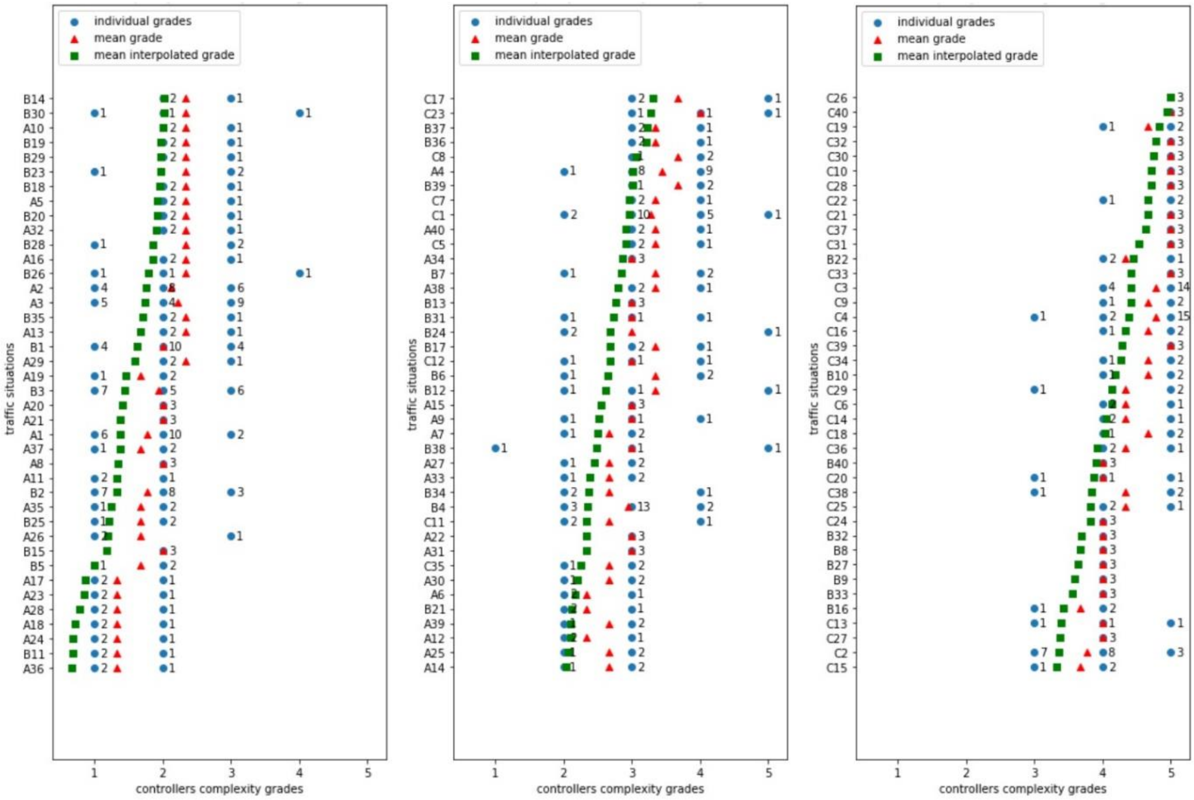


Figure 15: Complexity scores given to the original set of 120 traffic situations by the air traffic controllers

4.4. Statistical modeling

Python 3.7 is used as a programming language within Jupyter notebook environment for statistical modeling, and Python packages *sklearn*, *pandas* for statistical modeling and data manipulation, and *matplotlib* for statistical graphics (Appendix 7).

Two kinds of statistical models were tested:

- logistic regression which works on binary comparison target variable data, and
- linear regression which learns from air traffic controller's scores.

For now, mostly the linear regression modeling approach has been used as it showed more promising results. But, in future work other approaches will be tested.

Logistic regression is used for pairwise comparison modeling. The model was abandoned after few experiments because linear regression models showed better results, but it might be worth revisiting it as future work, especially if the regularized version of logistic regression is used.

For the linear regression, Bayesian Ridge Regression is used. Other nonlinear models for regression will be used as future work. The plan is to start with support vector machines (SVM) and random forest and avoid neural networks as these are harder to train especially with a smaller data sample.

The final formula for the air traffic complexity using the Bayesian Ridge Regression, since it yields best results, will be presented below. Using Bayesian Ridge Regression to join the selected set of binary states Ω from step B. and the binary states from step C from Chapter 3.4 as well as N^{in} and N^{out} for the given traffic situation w^l, for which the values of Δ_{ij} are calculated accordingly and associated to the mean of the interpolated score estimate η^l all for the purpose of generating a linear model η_{real} that is depending on the weight coefficients β_z, $\gamma_{z'}$, $\alpha_{z''}$ where $\delta_{i,j} = 1$ for the $i = j$ and $\delta_{i,j} = 0$ for the $i \neq j$.

$$\boxed{\eta_{real} = \sum_{j \geq i \geq 1}^{N^l} \left[\Delta_{ij}\right]_l + F_l(N^{in}, N^{out})} \tag{3}$$

Where:

$$\left[\Delta_{ij}\right]_l = \sum_{z \in \Omega} \beta_z \, (B_z)_{ij}\left(1 - \delta_{i,j}\right) + \sum_{z'=1}^{3} \gamma_{z'}(C_{z'})_{ii} \, \delta_{i,j} \tag{4}$$

$$F_l\left(N^{in}, N^{out}\right) = \alpha_1 N_l^{in} + \alpha_2 N_l^{out} + \alpha_3 \left(\frac{N_l^{in}\left(N_l^{in} - 1\right)}{2}\right) + \alpha_4 \left(\frac{N_l^{out}(N_l^{out} - 1)}{2}\right) + \alpha_5 \left(\frac{\left(N_l^{in} + N_l^{out}\right)\left(\left(N_l^{in} + N_l^{out}\right) - 1\right)}{2}\right) \tag{5}$$

The process ends by determining all the coefficients of the above model β_z, $\gamma_{z'}$, $\alpha_{z''}$ in a way that they first need to be standardized so that the mean of every feature is zero and with unit variance before the application of the formula (3). This approach was favored instead of giving out the raw coefficient, because the ATCO tasks can be chosen from any given set of Ω and because of that every time the feature set is changed, the weight coefficients are different.

4.4.1. Recursive feature selection

Now that the model is defined, a recursive forward feature selection will be performed to see which combination of features gives the best correlation score. Recursive forward feature selection chooses greedily the best performing features one-by-one. In general, the most informative features are the ones related to aircraft counts in some way.

All that was done with feature selection leads to two conclusions. First, many features are highly correlated with each other, which enables the option to use less features in order to achieve good complexity estimates. Second, the features that perform best in the statistical modeling are those that somehow encode an information on the number of aircraft in the airspace.

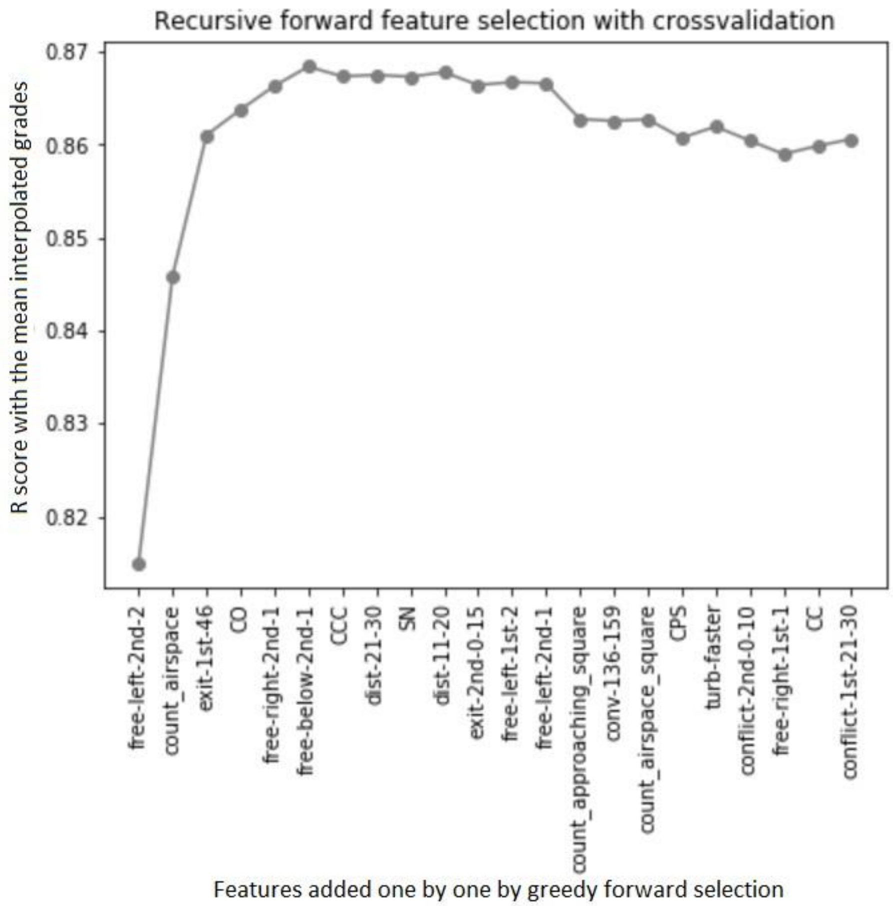


Figure 16: Recursive forward feature selection

Figure 16 shows the correlation between consecutive feature selection and the mean interpolated grade. It can be seen that the model can reach high correlation score after the combination of the first six features presented on the abscissa in Figure 16. Most of the features that contribute to the high correlation are the ones that contain the information about the freedom of movement of the aircraft, number of aircraft in the airspace, and conflict with the opposite track.

4.4.2. Feature sets

Several feature sets are constructed which were then used in the statistical modeling as exploratory variables. First three are used in logistic regression modeling where the comparison data target variables were used for learning:

- *features1* – original set of 69 task types (all features)
- *features2* – all features with aircraft counts
- *features3* – square root of all features, with aircraft counts

The rest are used in linear regression modeling where the controller's grades were used as target variables for learning:

- *features4* – all features
- *features5* – all features with aircraft counts
- *features6* – square root of all features with aircraft counts
- *features7* – only task types ($B_1 – B_4 + B_5 – B_7$) with aircraft counts
- *features8* – only numerical features ($B_8 – B_{50}$) without task types and with aircraft counts
- *features9* – only aircraft count features (*5 features from formula (5) without the* $\alpha_{z''}$)
- *features10* – just pairs within the airspace $\left(\frac{N_l^{in}(N_l^{in}-1)}{2}\right)$ feature
- *features11* – just pairs within the airspace $\left(\frac{N_l^{in}(N_l^{in}-1)}{2}\right)$, *SN, CO, SP* and N^{in} features
- *features12* – just pairs within the airspace $\left(\frac{N_l^{in}(N_l^{in}-1)}{2}\right)$, *CO, PS, PC* and N^{in}features
- *features13* – just *CO, CPS, CPC, CS, CC* features
- *features14* – all possible conflict ($B_1 – B_4 + B_5 – B_7$) and freedom of movement tasks ($B_{35} – B_{50}$)
- *features15* – all possible conflict $B_1 – B_4 + B_5 – B_7$ and $B_{14} – B_{34}$
- *features16* – features like v14 but with aircraft counts
- *features17* – features like v15 but with aircraft counts

4.4.3. Bootstrapping

The bootstrap method is a resampling technique used to estimate statistics on a dataset by sampling a dataset with replacement. Bootstrap aggregating was done on all the 17 feature sets, where the data set (features and target variable) was repeated 300 times and each time it was randomly divided into two sets, one for training (80%) and the other for testing (20%). After training on the 80% of the random sampled data set, it was tested on the remaining 20% and the Pearson correlation coefficient was calculated (Figure 17). The graphs of all 17 models are a distribution of out of sample correlations of the model complexity and the actual complexity from the controllers, where a red line is the mean value of Pearson correlation of the model and controllers' complexity results.

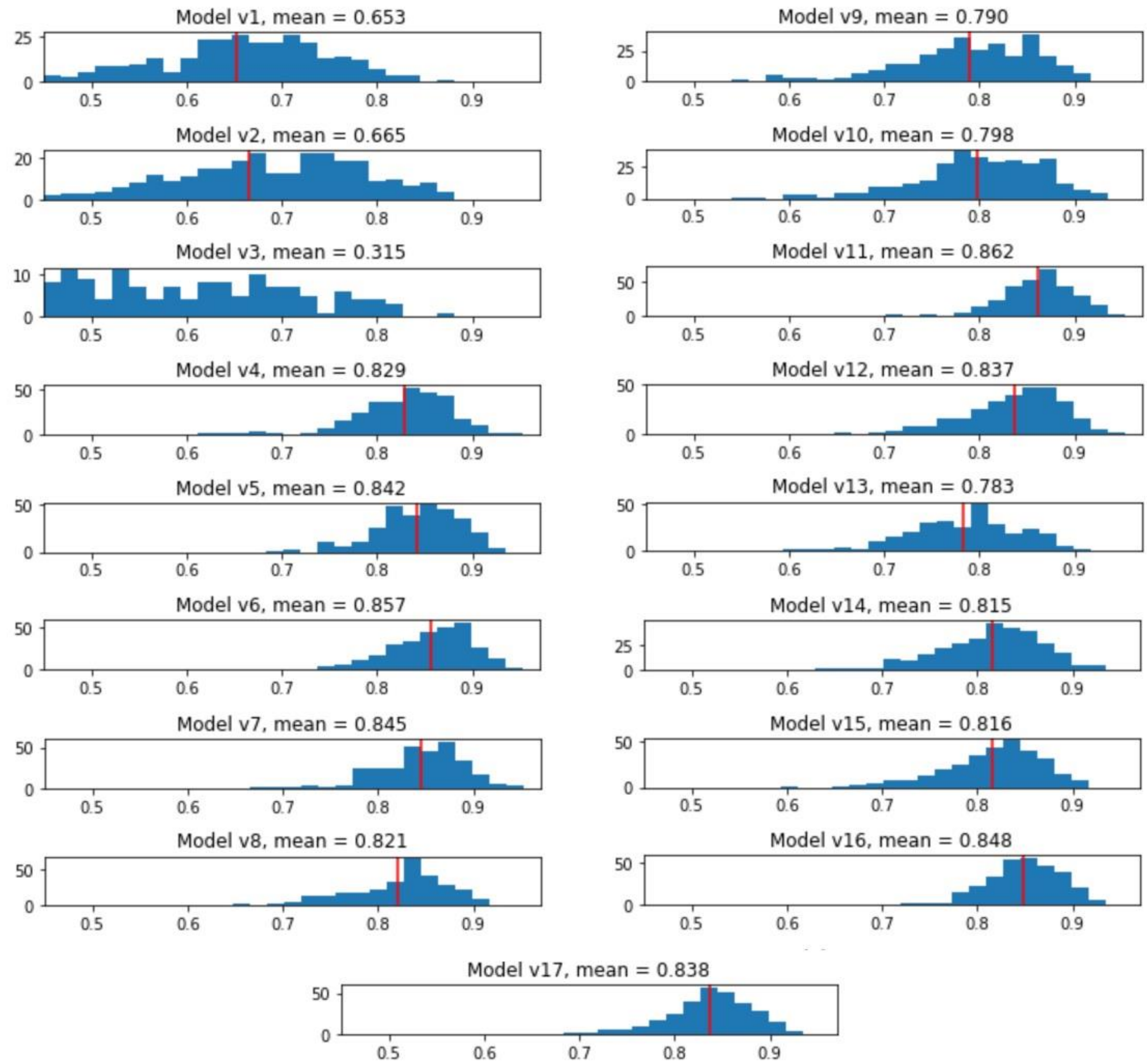


Figure 17 Bootstrap aggregating results for the selected feature sets

Direct comparison of the two models to see how do the results of one model better than the other in terms of 300-fold situations. For example, v6 and v11 are shown to have very similar scores (Figure 18).

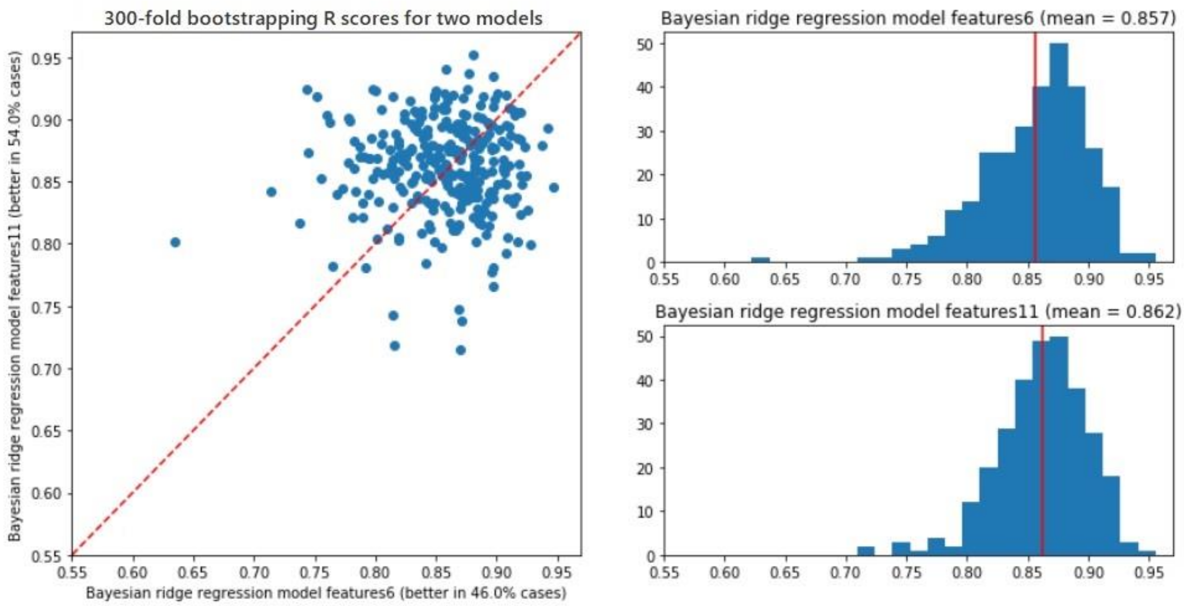


Figure 18 Bootstrap aggregating results comparison for two models

Although feature set 11 has a slightly better correlation result compared to feature set 6, based on the expert knowledge and the data that is imbedded in the exploratory variables from feature set 6, feature set 6 was taken as the final model of this research. Later on, in the validation section of the experiment, it is can be seen that the reason behind this decision was justified, when the comparison of the two models is shown again.

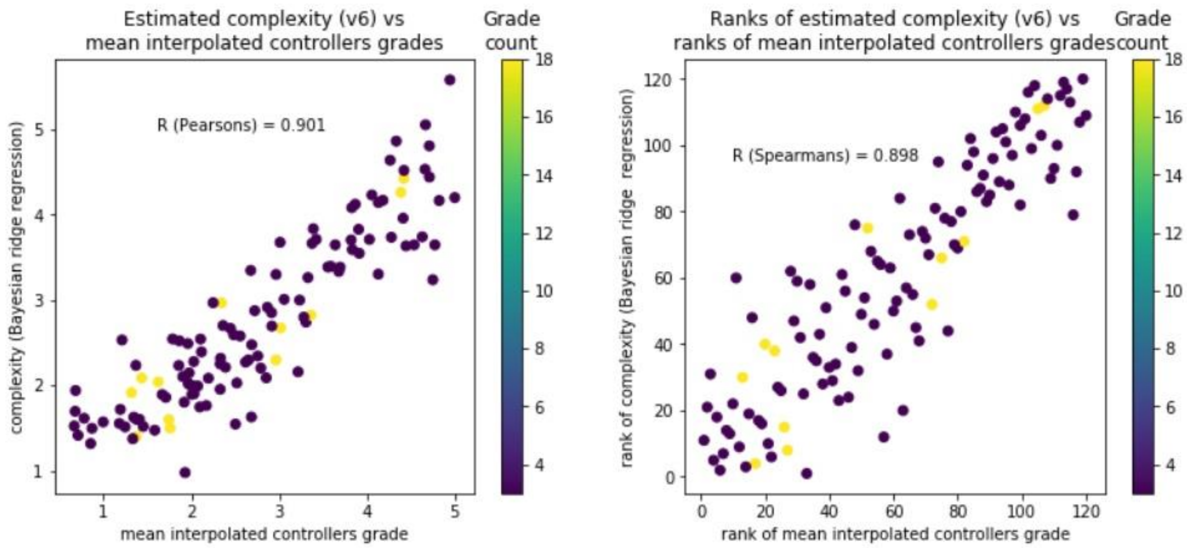


Figure 19: Pearson and Spearman correlation of model complexity compared to the air traffic controllers' complexity estimation

In order to test the maximum correlation strength, the chosen model will be tested and trained on the whole data set. This result will not be taken as a real and final result of the correlation coefficient, but just to see how high the correlation score can reach when training on a whole data set. The process was repeated 600 times and tested on the random 20% of the data set. When the Pearson coefficient correlation of the model complexity estimates is compared to the air traffic controllers' estimates, a result of R = 0.901 is obtained (Figure 19). A good linear correlation can also be seen when the model ranks the traffic situations by complexity and compares it to the air traffic controllers ranking (Figure 19).

5. Result analysis

It was established in the previous chapter that the version 6 of the feature set was taken as a final feature set to train the model. Validation results showed very good results, which is further analyzed in this chapter, alongside with the model complexity score, controllers' ranking consistency, and the real-life application of the air traffic complexity model.

5.1. Score analysis

How the model compares to the air traffic controllers' complexity evaluation (target variables) is presented through the score comparison between the air traffic controllers and the model.

5.1.1. Consistency of controller's grades

In order to evaluate how the air traffic controllers graded each individual traffic situation, a distribution of the controllers' grades and model 90% interval score is plotted in Figure 20.

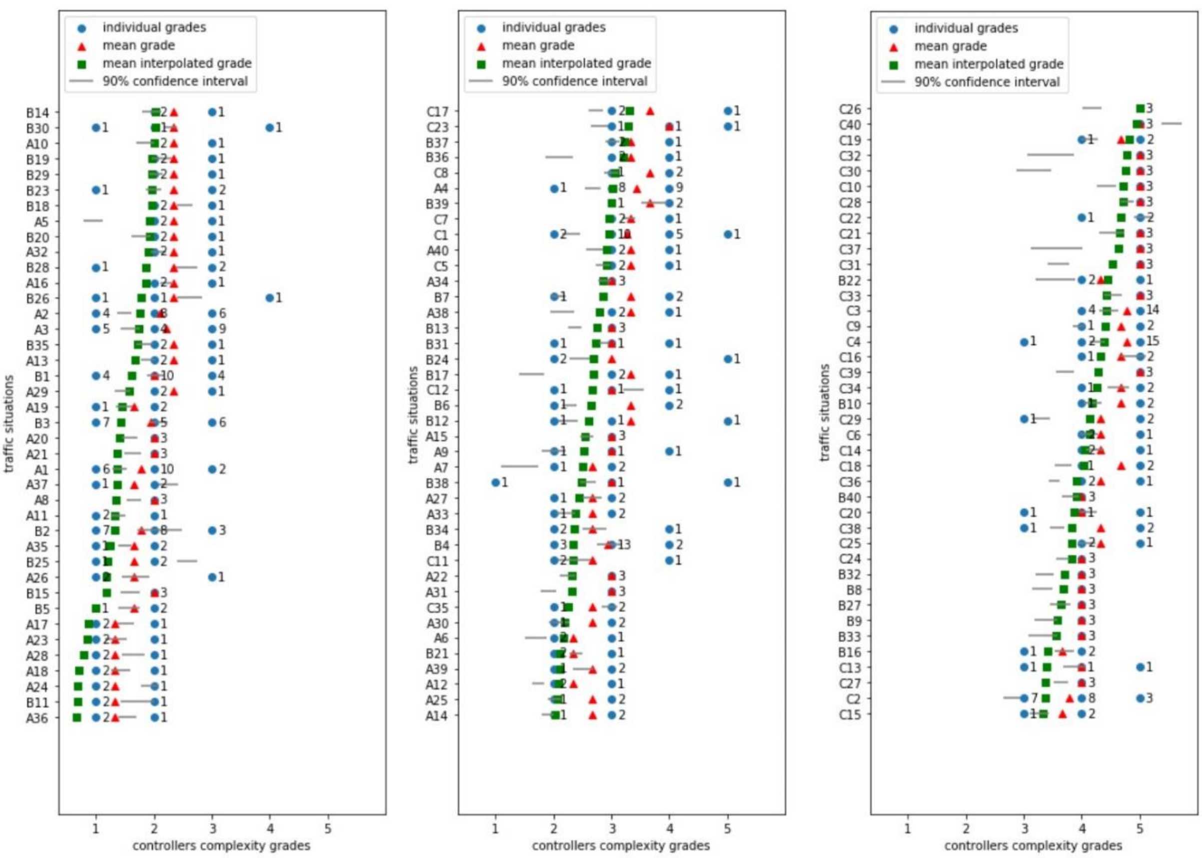


Figure 20: Complexity scores given by the air traffic controllers and the model

In Figure 20 a ranking by mean interpolated grade from the lowest complexity to the highest complexity is compared with the models 90% interval confidence. It can be seen that different air traffic controllers had a difference in opinion for the same traffic situation, which only confirms the theory that the complexity is a subjective construct. Furthermore, models 90% interval confidence is also not 100% accurate. After analyzing it, it is clear that the interval is almost all the time in the range that the air traffic controllers would estimate, but for some situations it can have greater deviation than the air traffic controller. But, it is important to mention when looking at the overall situation, the model will have less deviation from the mean interpolated grade then the air traffic controllers.

A closer look in the consistency of the controllers ranking and grading situation that all the 18 air traffic controllers graded is shown in Figure 21.

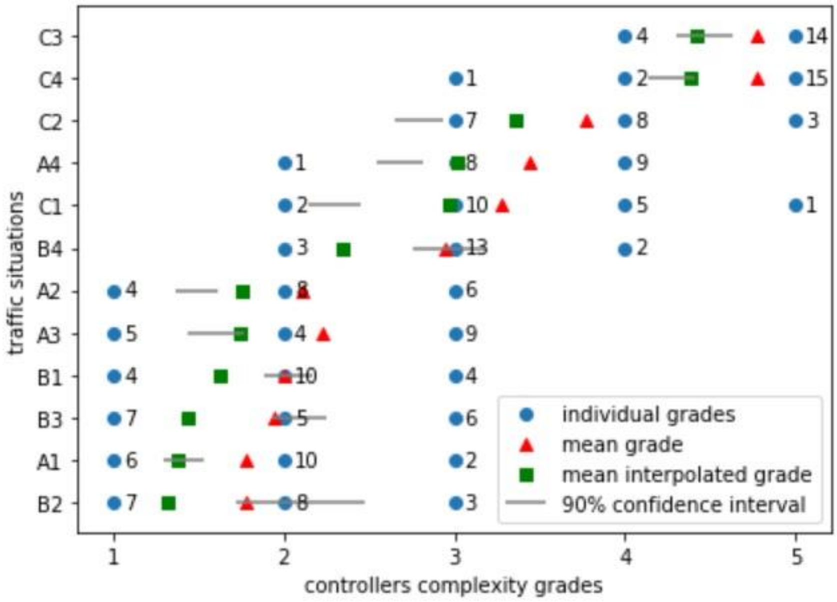


Figure 21: Complexity scores given by all 18 air traffic controllers and the model

Figure 21 perfectly shows and confirms the earlier mentioned statement. It can be seen that the models 90% confidence interval is always less wrong in the complexity estimation then all air traffic controllers were for each traffic situation. This becomes apparent when more air traffic controllers start to give complexity estimates for the same traffic situation.

5.1.2. Similarities between controllers

There are 12 situations which are shared between all controllers (A1-A4, B1-B4, C1-C4). How similar are controllers' rankings of these situations? To analysis this, a Spearman correlation matrix is plotted (Figure 22) for all controllers, sorted by the average similarity, so that the controllers that are most similar on average to everyone else are on the top left. Controller 19 is the statistical model version 6.

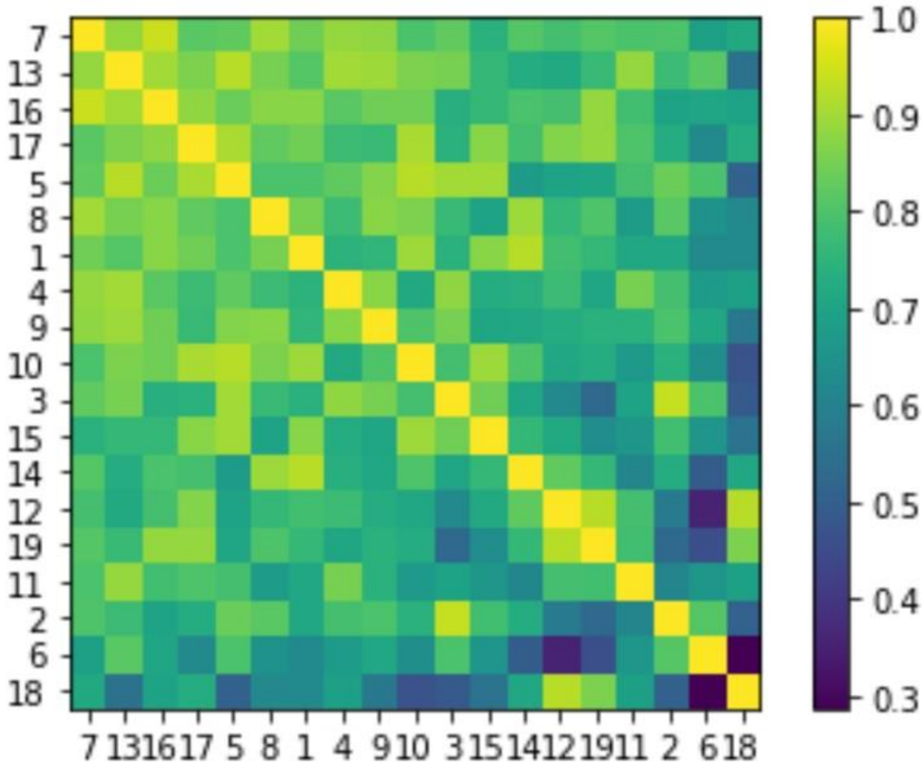


Figure 22: Matrix of ATCO correlation between themselves

Here, the model can be observed as a new air traffic controller (no. 19) and it is clear that each air traffic controller has its own bias. Some corelate well with each other or it can be said that they have the same opinion as their colleagues, and vice versa.

5.1.3. Task complexity

The chosen model can give indirect estimates of complexities of individual tasks and individual aircraft. This can be looked as the interpretability of the model. These estimates (Figure 23) are extracted from the coefficients of the linear regression model. The 100-fold crossvalidation is performed in order to get a distribution of coefficients. It can be seen from Figure 23 that a conflict with the opposite track and a conflict with the same track have high complexity impact on overall traffic situation, while some tasks such as potential conflict with crossing track reduce the overall complexity.

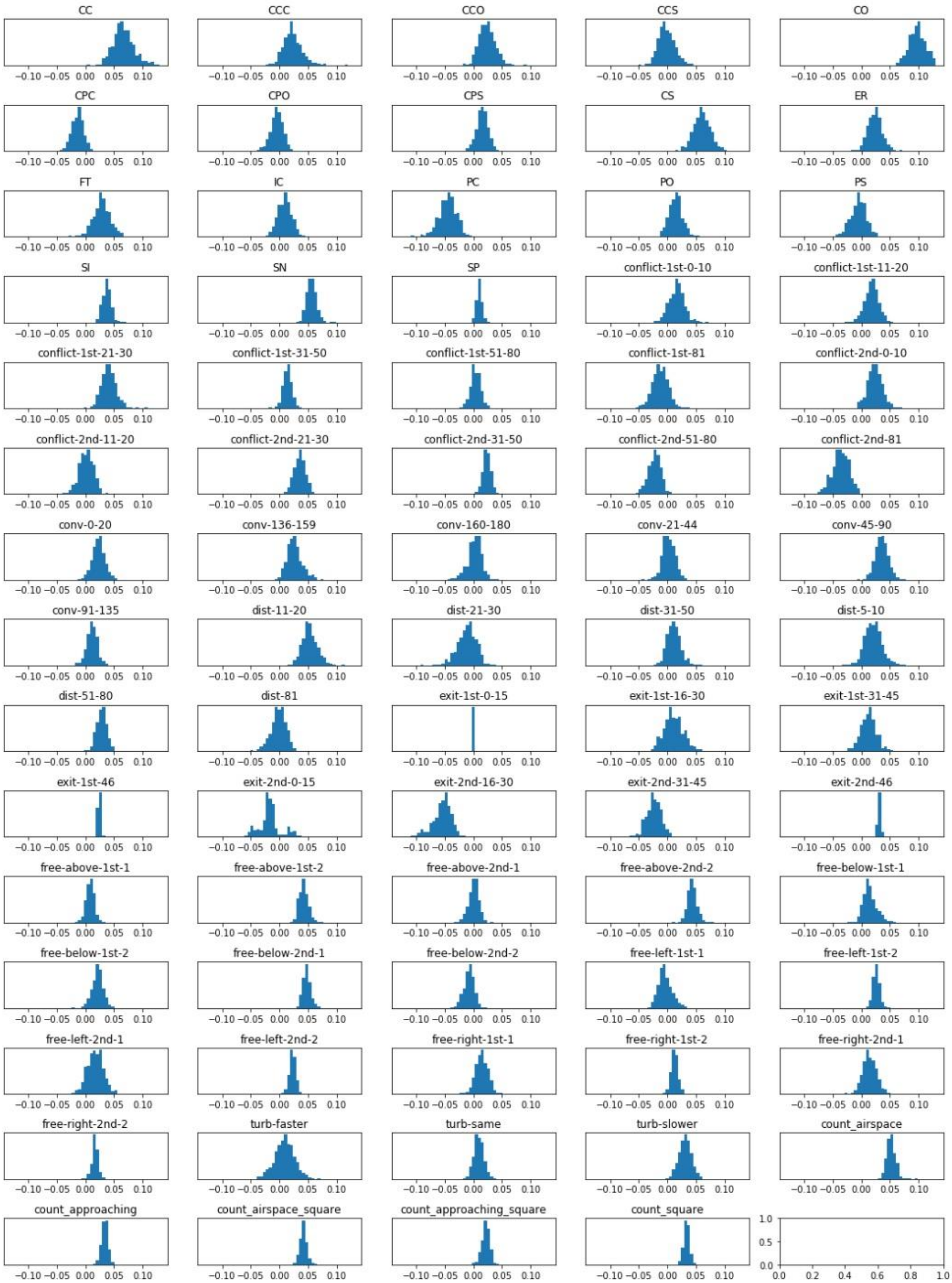


Figure 23: Estimates of complexities of individual tasks

5.2. Validation of the complexity model

Now that the model has been built and it calculates the air traffic complexity, it is going to be validated. There are two ways the model is validated. First, the results of the model are compared to the results of the air traffic controller to see how much they deviate from the mean interpolated score and use that as an error reference. Second, the trained model is tested on a whole new airspace and air traffic, and again, the results of the model are compared to the results of the air traffic controller to see how much they deviate in a new airspace and traffic from the mean interpolated score and use that as an error reference.

5.2.1. Model vs Controller

How much controller's grades differ from the mean grade calculated across all controllers is interesting data to analyze. This difference can be expressed in two ways:

1. Average case, where an average difference is taken from the mean grade across all 30 situations that a particular controller graded, and
2. Extreme case, where the maximum absolute difference (extreme difference) is taken from the mean grade among all 30 situations that a particular controller graded.

For each of these cases one value per controller was obtained, and these are shown by red vertical lines in Figure 24. This statistical model can provide a distribution of estimates for each of these cases, and these distributions are plotted on the same graphs bellow. Distributions are calculated by building 1000 separate statistical models on 1000 random subsets, each containing 30 situations - the same number of situations that controllers had to grade.

The preliminary analysis shows that the model is comparable to the evaluations expected from the controllers. This analysis approach will be favored through the experiment as the main evaluation measure because it compares the performance of the model directly with the controller's in fairly interpretable way, unlike when using correlation coefficient for evaluation.

In Figure 24 the distributions for the underestimates and overestimates are separated, as the preliminary analysis indicated that they are not symmetrical, in fact, the distribution for the extreme differences is bimodal. Separating this distribution into extreme negative differences and extreme positive differences shows that indeed these two distributions have different means. In general, it is important to know whether the model is overconfident or underconfident. The model, ideally, should be as conservative as possible, especially in the extreme case (as opposed to the average case), and prefer overestimation over underestimation.

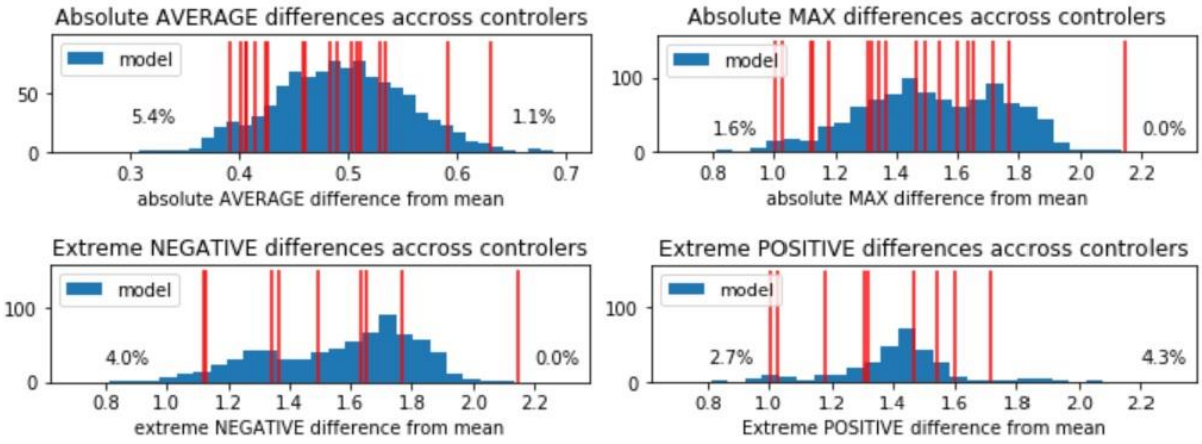


Figure 24: Validation of the model compared to the controllers' deviation to the mean interpolated scores

There are 18 controllers, divided into 6 groups where all controllers within one group are given an identical set of 30 air traffic situations. In Table 17, the average difference between controller's complexity estimates across 30 situations is observed, and the mean of his group which contains three controllers in total. This is compared with the estimates from the chosen statistical model.

Table 17: Average difference between controllers' complexity

Group 1		Group 2		Group 3	
ATC No.	Average diff.	ATC No.	Average diff.	ATC No.	Average diff.
1	0.533259	4	0.458540	7	0.401684
2	0.483319	5	0.528410	8	0.405489
3	0.424635	6	0.508308	9	0.459599
Model=>	0.489050	Model=>	0.549737	Model=>	0.492913
Group 4		Group 5		Group 6	
ATC No.	Average diff.	ATC No.	Average diff.	ATC No.	Average diff.
10	0.502113	13	0.391835	16	0.511063
11	0.489007	14	0.406116	17	0.414378
12	0.590718	15	0.426546	18	0.630797
Model=>	0.594016	Model=>	0.500404	Model=>	0.586074

Table 17 shows that the average difference of the trained model is within the average difference as the air traffic controllers estimated per each individual group.

5.2.2. New airspace

Final evaluation of the statistical model is performed on the validation air traffic situations - a separate set of air traffic situations which are defined on a completely new airspace. These are not used for training of the statistical model.

What if the statistical model performs worse on validation set? There can be two reasons for this:

1. New airspace means the complexity function is different, and the complexity model that was inferred is not representative anymore. This is both due to the shape of the airspace and due to controller's internal representation of complexity, both of which can change with different airspaces.
2. There are less validation situations, so that model estimates will have higher variance, and therefore larger error. To some degree, these two causes can be separate, as change in the underlying complexity function will be reflected in the consistent larger errors - models distribution of estimates will shift, while changes in variance will reflect in the width of the model distribution of estimates. So, the model can be more wrong (larger bias) and the variance of the model estimates will be larger (larger variance), which again leads to larger error.

For the validation of the model on a new airspace, the same testing methodology will be used as it was done when comparing the model with the controllers' scores, building a 1000 separate statistical models on 1000 random subsets, each containing 6 situations - the same number of situations that controllers had to grade for the new airspace (Figure 25).

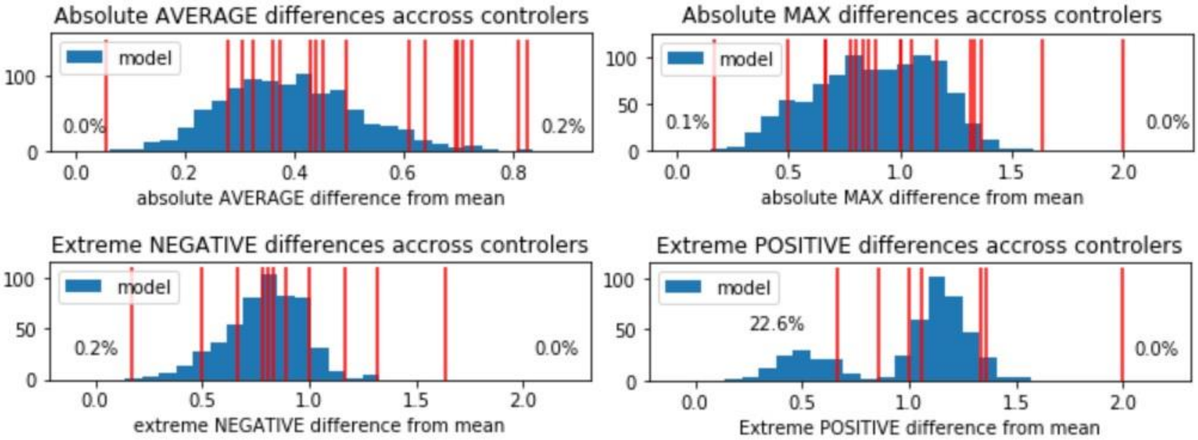


Figure 25: Validation of the chosen model v6 on a new airspace compared to the controllers' deviation to the mean interpolated scores

As mentioned in the model development, similar scores in the bootstrapping testing achieved chosen model 6 and model 11. To see that the feature sets from model 6 was the right choice, the same validation will be performed for model 11 in Figure 26.

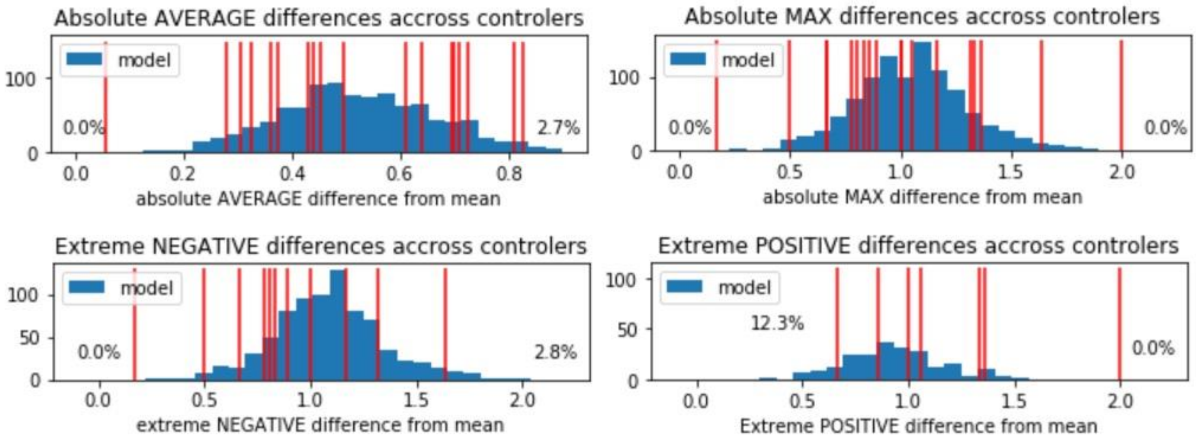


Figure 26: Validation of model v11 on a new airspace compared to the controllers' deviation to the mean interpolated scores

It can be seen that when comparing validation results from chosen model v6 and model v11 model v6 has better overall results although it had lower R score in the bootstrapping testing. The distribution is wider for model version 11 which is not that good compared to version 6. In average difference, version 11 has higher error by 2.7% then the most extreme air traffic controller. By observing the results from these two comparisons, it can be seen that complexity does not only depend on the aircraft number, but the overall ATCO tasks that are produced form the traffic situation.

To take a closer look into the validation of a new airspace and traffic, a mean and extreme differences for each of the 28 validation situations are presented in Figure 27. Random subset of 90 situations will be used for training in order to build multiple instances of the model and apply it to each validation situation. Figure 27 shows plotted distribution of the model estimates and the estimates of controllers that evaluated each particular situation, both discrete and interpolated, where doted red lines represent each controllers maximum deviation from the mean interpolated grade, green line is the mean interpolated grade and red line is the mean grade.

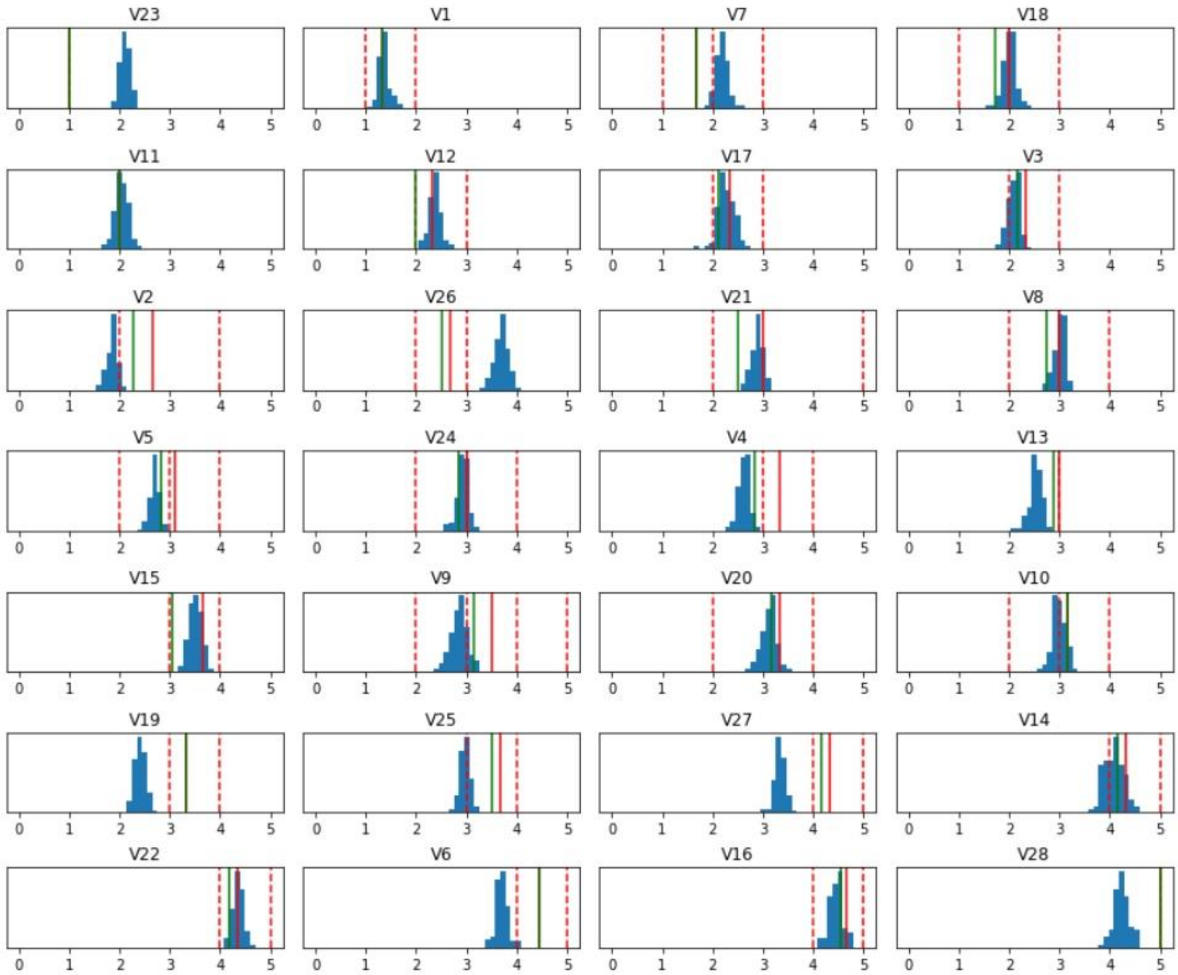


Figure 27 Validation of each individual new traffic situation in new airspace

In order to evaluate how the air traffic controllers graded each individual new traffic situation in new airspace, a distribution of the controllers grades and model 90% interval score is plotted in Figure 28.

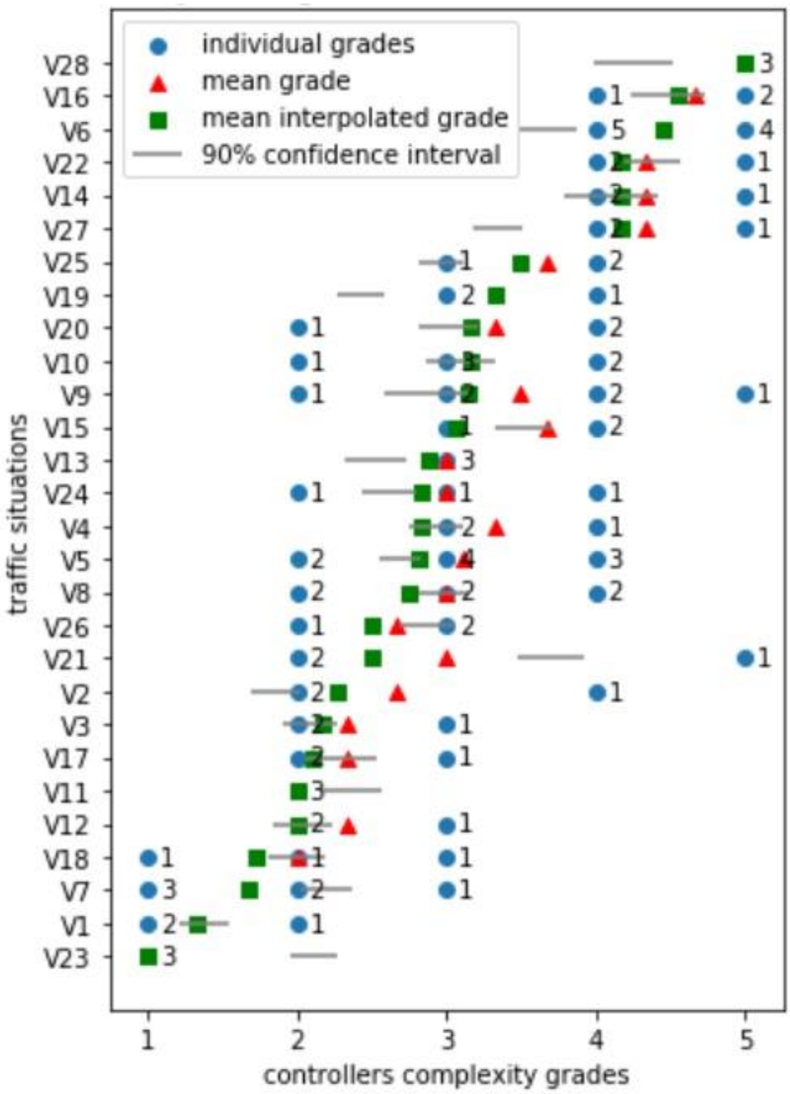


Figure 28: Complexity scores for the validation airspace given by the air traffic controllers and the model

In Figure 28, a ranking by mean interpolated grade from the lowest complexity to the highest complexity is compared with the models 90% interval confidence. It can be seen that different air traffic controllers had a difference in opinion for the same traffic situation, but the models 90% interval confidence is also not 100% accurate. After analyzing Figure 28, it is clear that the interval is almost all the time in the range that the air traffic controllers would estimate, but for some situations it can have greater deviation than the air traffic controller. When V5 is observed and examined in greater detail, it can be seen that the models distribution is extremely on spot to the mean interpolated grade. The reason behind these good results is the fact that V5 was estimated by 9 air traffic controllers, compared to the usual practice of 3 ATCOs per traffic situation. Also, it is important to mention when looking at the overall situation, the model will have less deviation from the mean interpolated grade than the air traffic controllers.

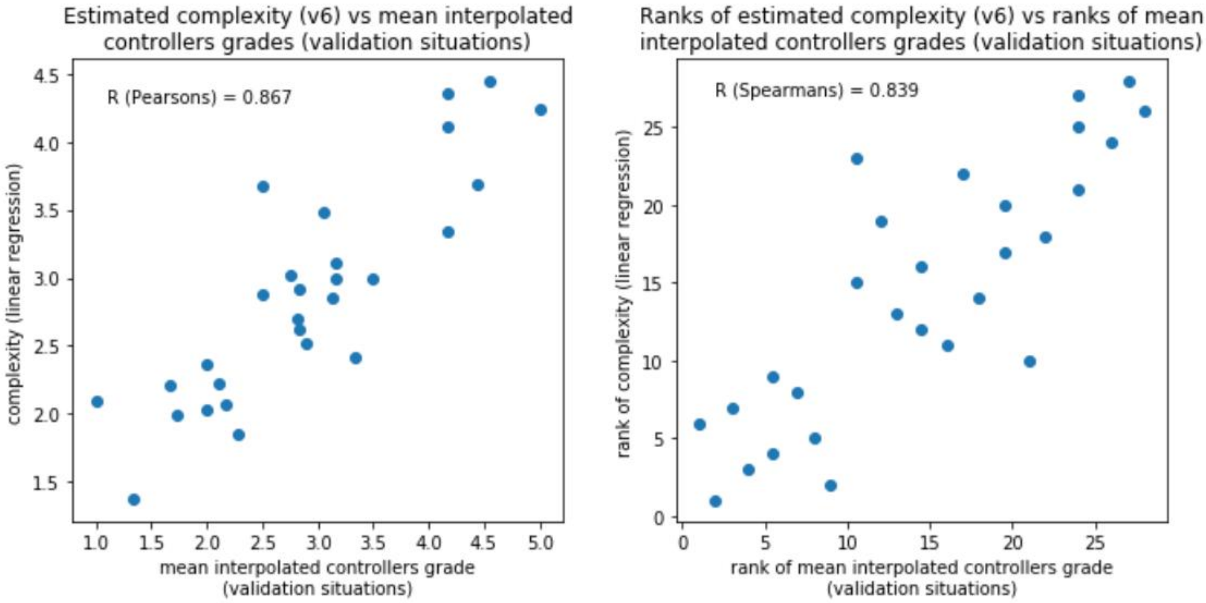


Figure 29: Pearson and Spearman correlation of model complexity compared to the air traffic controllers' complexity estimation for the new, validation traffic situations

Pearson and Spearman correlation of model complexity compared to the air traffic controllers' complexity estimation for the new, validation traffic situations is shown in Figure 29. The model was trained on 100% original data set and it was tested on the random 20% of the new, validation data set, where the process was repeated 600 times. When Pearson coefficient correlation of the model complexity estimates is compared with the air traffic controllers' estimates, a result of R = 0.867 is obtained (Figure 29).

5.3. Practical application

Now that the model is finish and the best one is chosen based on the bootstrap aggregating validation it can be applied on the cherry-picking process for the purpose of best STAM (Short Term ATM Measures) measures and for the adequate sector optimization.

5.3.1. Cherry-picking

Cherry-picking is done by a wrapper approach where, by excluding certain aircraft, it can be seen how much the complexity changes. The aircraft whose removal leads to the highest reduction in complexity are considered the most complex. The advantage of this wrapper method, as compared to extracting complexities directly from the coefficient of the regression, is that this method is model-agnostic and can be performed even with nonlinear models with many non-interpretable or even hidden parameters.

In order to get more stable estimates of aircraft complexities, the leave-one-out (LOO) crossvalidation procedure is applied. So instead of building the full model on all situations and using this to estimate complexities of individual aircraft, the LOO crossvalidation models with *N-2* situation is build, excluding always the situation to which a particular aircraft belongs to, and additionally one more situation which is a part of the excluded crossvalidation set. Now, a distribution of complexities for each aircraft is obtained instead of a single value. The mean value is used to obtain a single value for the calculated complexity distribution, which makes this approach for rankings and estimates more robust.

Following Table 18 shows aircraft complexities for two situations, along with the number of *C* and *P* ATCO tasks associated with each aircraft. These two values can be viewed (number of *C* and *P* type of conflicts) as a crude estimate of aircraft's complexity. The number of conflicts is only looked upon and other features are not, for example, the number of aircraft or *SN* (non-interactive screenings), because features that are related to aircraft counts are identical for all aircraft (always, exactly one aircraft is removed from the situation).

Table 18: Example of cherry-picking aircraft

Aircraft	Complexity	Complexity difference	C tasks	P tasks
EWG343	3.227039	-0.429280	4	1
ADR259	3.275018	-0.381301	3	1
UAE2943	3.279483	-0.376837	5	0
DLH231	3.284907	-0.371413	5	0
AFR227	3.288807	-0.367513	4	3
DLH6372	3.289722	-0.366598	7	0
UAE943	3.293249	-0.363070	5	0
CTN4847	3.297458	-0.358861	3	0
AAL294	3.309715	-0.346604	4	0
MGX916	3.315988	-0.340331	2	2

This can be done on all STAM measures, not just leave-one-out method. The most complex aircraft can be detected and then some of its parameters can be changed, in other words, apply a certain STAM measure and calculate the complexity again to see how the result would behave with a new, applied STAM measure. For the model, this would be a completely new air traffic situation and it would just need to compare the results of the new one and the original one. With this approach, model could also tell the user what the best STAM measure to achieve best complexity reduction would be, something like what-if scenario for the multiple STAM measures.

5.3.2. Sector optimization

After controllers' provided a final complexity ranking of all the 30 situations that they were evaluating, additionally, they were asked to mark a position in the ranking after which they would suggest opening a new sector. All air traffic situations bellow that mark were considered as 0 (do not open a new sector) and all situations above as 1 (open a new sector). As each situation was evaluated by multiple controllers, the graph in Figure 30 shows a total number of 0s and 1s obtained for each individual situation.

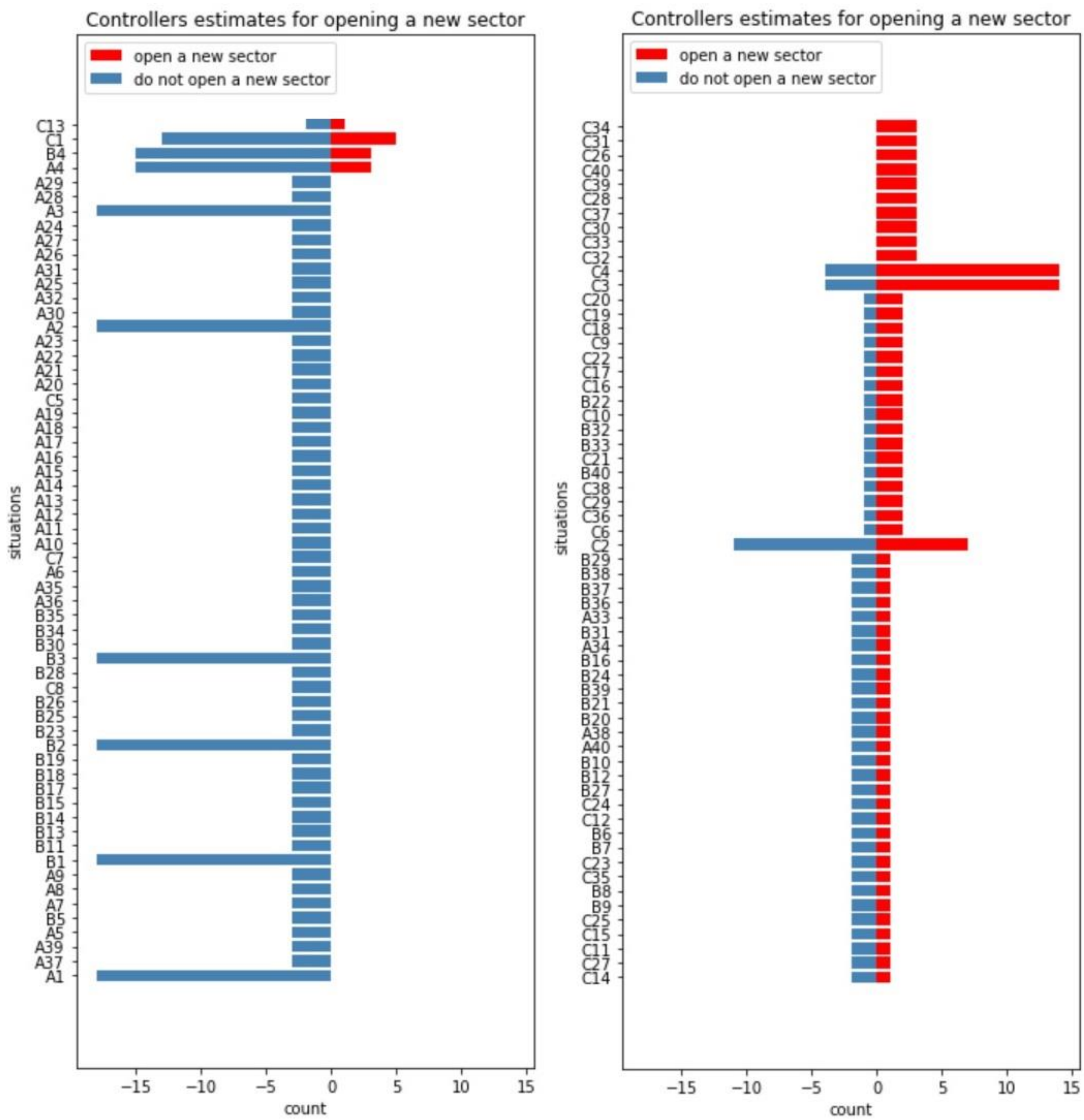


Figure 30: Air traffic controllers estimates for opening a new sector

In the following section, an attempt to explore the possibility of modelling the opening of a new sector in two different ways is explained. As mentioned earlier, opening of a new sector is a special event when the complexity of a situation warrants dividing the current air space sector into two or more separate sections. During the experiment, all controllers were asked to mark the *NS* event, a particular place between the two air traffic situations, where they would suggest opening a new sector.

There are two ways of how this information on opening of a new sector is encoded:

1. Discrete - There is only information on whether a controller classified certain situation as not *NS* (value 0) or *NS* (value 1). Mean value is then calculated for each situation across all controllers that graded a particular situation. Value can range from 0 (no controller who evaluated this situation considers opening a new sector) to 1 (all controllers who evaluated this situation consider opening a new sector).
2. Interpolated - Interpolated distance from the *NS* marking is calculated for each situation, taking into consideration interpolated grades calculated for each controller and each situation. Mean value is then calculated for each situation across all controllers that evaluated this situation, similarly as in interpolated score case. In theory, values can range from -5 to 5, but in practice they range from -3 to 2.

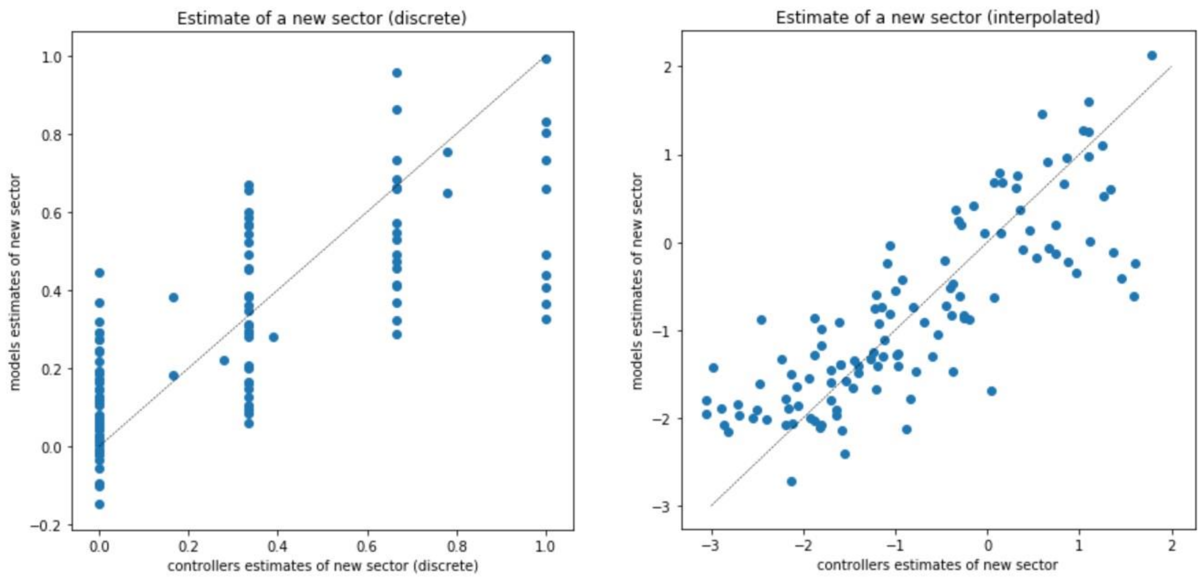


Figure 31: Opening sector discrete versus interpolated approach

From Figure 31 it can be seen that both approaches provide the necessary information needed for opening a new sector. In the left graph, discrete method was used where the sorting of the situations was done by a mean value. Here, it can be seen that the error in estimation can happen between the model and the controllers' prediction. But, when looking at the right graph, which shows the interpolated method, it can be seen that the error in estimation between the controllers and the models estimation is also present, but in less degree, which makes it a better approach when trying to model the sector optimization scheme.

6. Conclusion

This research was motivated by the fact that almost after two decades of research, the problem of determining adequate complexity score was an issue in air traffic control, because it was considered subjectively, from the air traffic controllers' perspective. The air traffic controllers observed and analyzed the traffic data and decided whether a traffic situation is complex or not. This problem was solved within this research and the results will be presented in the text below.

With that said, it can be concluded that in this research, the author successfully managed to design and make a new model that can calculate air traffic complexity based on the air traffic controller tasks, which confirms the set hypothesis at the beginning of the document.

One of the greatest advantages and the strengths of the model is that the model is LOAA, Learn Once Apply Anywhere. This was confirmed through the validation process of the model. Not only is the model fitted to the air traffic controllers opinion regarding the air traffic complexity, but it can also be used on a new, unseen airspace and calculate the air traffic complexity with less error than the air traffic controllers would assess it.

This alone is a huge achievement in the field of air traffic complexity modeling. Finally, the problem of adequate complexity assessment is solved and the wide application of the information that was locked behind the air traffic complexity can now be properly researched and examined further.

With the adequate air traffic complexity metric, flow management position can start to make precise decisions regarding the sector optimization and flow management of aircraft. Correct STAM measures can be taken into account and based on the complexity metric, a precise sector optimization can be put in place. All these claims were proven through practical application chapter. Another interesting possible use of the model can be for detection of the ATCOs situational awareness level by maintaining a specific level of complexity. Thus, out-of-the-loop effect for the air traffic controller can be prevented.

For future work, the author believes that he can make enhancements on the model by observing the influence of ATCOs years of experience and gender have on the complexity assessment. Also, by observing the merge sort results. During the data gathering experiment, the author took extensive notes where he spotted any inconsistencies in the air traffic controller ranking. By applying a new rank order, the author believes that the model can be enhanced further and thus achieve better complexity estimation than the current state.

Literature

[1] Performance Review Commission. Performance Review Report 2017. EUROCONTROL; 2018.

[2] Performance Review Unit. European ANS Performance Data Portal n.d. http://ansperformance.eu/ (accessed November 12, 2018).

[3] STATFOR. EUROCONTROL Seven-Year Forecast February 2018. 2018.

[4] Meckiff C, Chone R, Nicolaon J-P. The Tactical Load Smoother for Multi-Sector Planning, Orlando, USA: 1998.

[5] Davis CG, Danaher JW, Fischl MA. The influence of selected sector characteristics upon ARTCC controller activities. Arlington: The Matrix Corporation 1963.

[6] Mogford RH, Guttman JA, Morrow SL, Kopardekar P. The Complexity Construct in Air Traffic Control: A Review and Synthesis of the Literature. MCKEE CITY NJ: CTA INCORPORATED; 1995.

[7] Hilburn B. COGNITIVE COMPLEXITY IN AIR TRAFFIC CONTROL: A LITERATURE REVIEW. Center for Human Performance Research; 2004.

[8] Schmidt DK. On Modeling ATC Work Load and Sector Capacity. Journal of Aircraft 1976:531–7.

[9] Hurst MW, Rose RM. Objective Job Difficulty, Behavioural Response, and Sector Characteristics in Air Route Traffic Control Centres. Taylor & Francis Ergonomics 1978:697–708.

[10] Stein ES. Air traffic controller workload: An examination of workload probe. FAA; 1985.

[11] Laudeman IV, Shelden SG, Branstrom R, Brasil CL. Dynamic Density: An Air Traffic Management Metric. Moffett Field: Ames Research Center; 1998.

[12] Chatterji G, Sridhar B. Measures for air traffic controller workload prediction, Los Angeles: 2001.

[13] Wyndemere. An Evaluation of Air Traffic Control Complexity. Boulder: 1996.

[14] Kopardekar P. Dynamic density: A review of proposed variables. Federal Aviation Administration; 2000.

[15] Kopardekar P, Magyarits S. Dynamic density: measuring and predicting sector complexity [ATC], IEEE; 2002. https://doi.org/10.1109/DASC.2002.1067920.

[16] Kopardekar P, Magyarits S. Measurement and prediction of dynamic density. Proceedings of the 5th USA/Europe Air Traffic Management R & D Seminar, vol. 139, 2003.

[17] Kopardekar P, Schwartz A, Magyarits S, Rhodes J. Airspace complexity measurement: An air traffic control simulation analysis. International Journal of Industrial Engineering: Theory, Applications and Practice 2009:61–70.

[18] Masalonis A, Callaham M, Wanke C. Dynamic Density and Complexity Metrics for Real-Time Traffic Flow Management, Budapest, Hungary: 2003.

[19] Klein A, Rodgers M, Leiden K. Simplified dynamic density: A metric for dynamic airspace configuration and NextGen analysis, Orlando, USA: IEEE; 2009. https://doi.org/10.1109/DASC.2009.5347539.

[20] Bloem M, Brinton C, Hinkey J, Leiden K, Sheth K. A Robust Approach for Predicting Dynamic Density, Hilton Head, South Carolina: 2009.

[21] Chaboud T, Hunter R, Hustache J, Mahlich S, Tullett P. Investigating the Air Traffic Complexity: Potential Impacts on Workload and Costs. Belgium: Eurocontrol; 2000.

[22] Performance Review Commission. Complexity Metrics for ANSP Benchmarking Analysis. Eurocontrol; 2006.

[23] Prevot T, Lee P. Trajectory-Based Complexity (TBX): A modified aircraft count to predict sector complexity during trajectory-based operations, Seattle, USA: IEEE; 2011. https://doi.org/10.1109/DASC.2011.6096045.

[24] Lee P, Prevot T. Prediction of Traffic Complexity and Controller Workload in Mixed Equipage NextGen Environments. Proceedings of the Human Factors and Ergonomics Society Annual Meeting 2012;56:100–4.

[25] Radišić T, Novak D, Juričić B. Reduction of Air Traffic Complexity Using Trajectory-Based Operations and Validation of Novel Complexity Indicators. TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS 2017:PP: 1-11.

[26] Prandini M, Putta V, Hu J. Air traffic complexity in future Air Traffic Management systems. Journal of Aerospace Operations 2012:281–99.

[27] Gianazza D, Guittet K. Selection and evaluation of air traffic complexity metrics, Portland, United States: IEEE; 2006, p. 1–12. https://doi.org/10.1109/DASC.2006.313710.

[28] Gianazza D. Forecasting workload and airspace configuration with neural networks and tree search methods. Artificial Intelligence, Elsevier 2010:530–49. https://doi.org/10.1016/j.artint.2010.03.001.

[29] Gianazza D. Smoothed traffic complexity metrics for airspace configuration schedules, Fairfax, United States: 2008.

[30] Lee K, Feron E, Pritchett A. Describing Airspace Complexity: Airspace Response to Disturbances. Journal of Guidance, Control, and Dynamics 2009;32:210–22.

[31] Wee HJ, Lye SW, Pinheiro J-P. A Spatial, Temporal Complexity Metric for Tactical Air Traffic Control. THE JOURNAL OF NAVIGATION 2018:1040–54. https://doi.org/10.1017/S0373463318000255.

[32] Rank A, Dervic A. ATC complexity measures: Formulas measuring workload and complexity at Stockholm TMA. Linköping University, 2015.

[33] Wang H, Song Z, Wen R. Modeling Air Traffic Situation Complexity with a Dynamic Weighted Network Approach. Journal of Advanced Transportation 2018:15. https://doi.org/10.1155/2018/5254289.

[34] Xiao M, Zhang J, Cai K, Cao X. ATCEM: a synthetic model for evaluating air traffic complexity. Journal of Advanced Transportation 2015. https://doi.org/10.1002/atr.1321.

[35] ZHU X, CAO X, CAI K. Measuring air traffic complexity based on small samples. Chinese Journal of Aeronautics 2017;30:1493–505.

[36] Andraši P, Radišić T, Novak D, Juričić B. Subjective Air Traffic Complexity Estimation Using Artificial Neural Networks. Promet – Traffic & Transportation 2019;31:377–86.

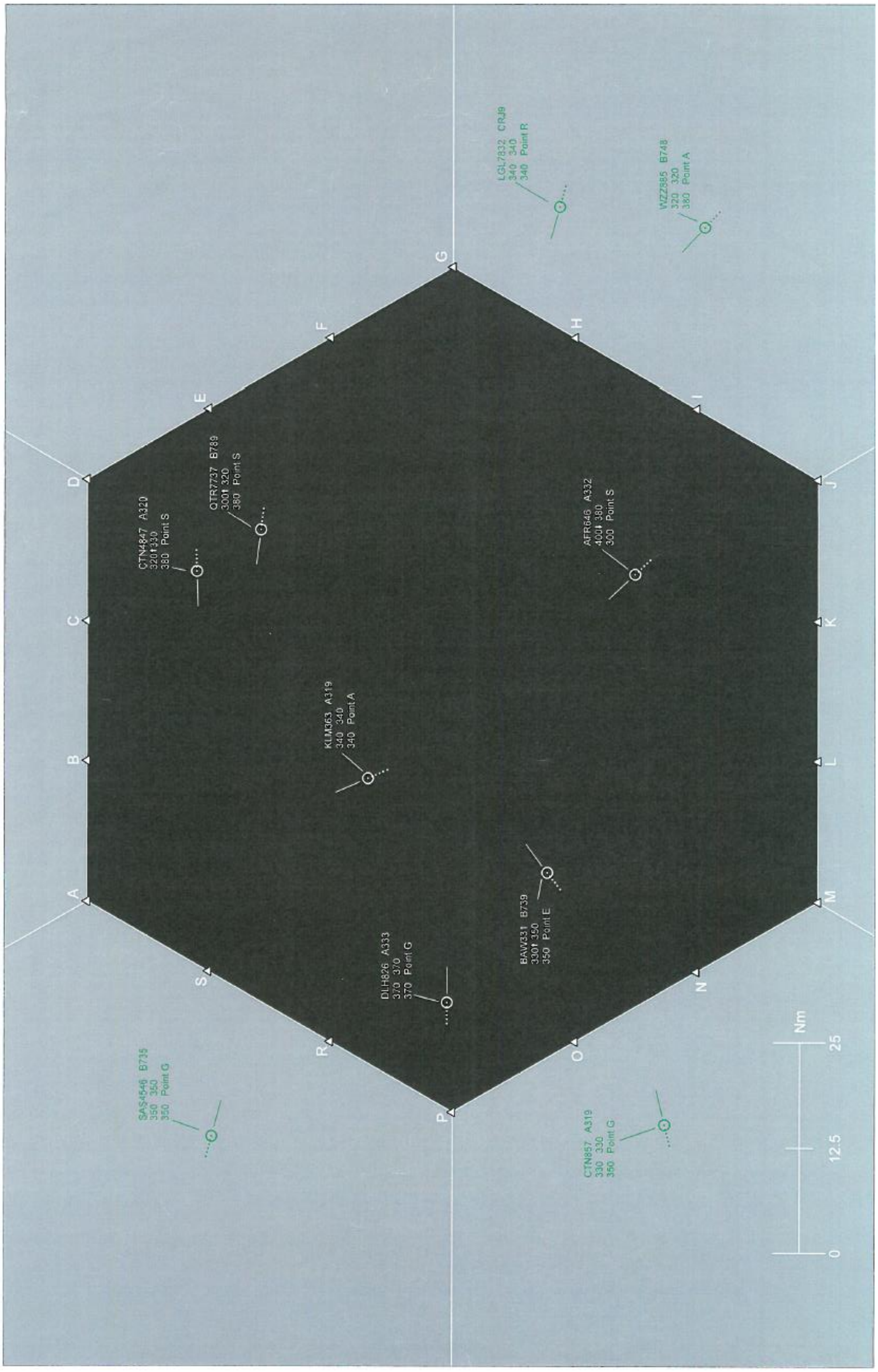
[37] International Civil Aviation Organization Doc 4444 Procedures for Air Navigation Services; Air Traffic Management. Quebec, Sixteenth edition 2016; ISBN 978-92-9258-081-0.

[38] Chatfield, C. (1995). Problem Solving: A Statistician's Guide (2nd ed.). Chapman and Hall. ISBN 978-0412606304.

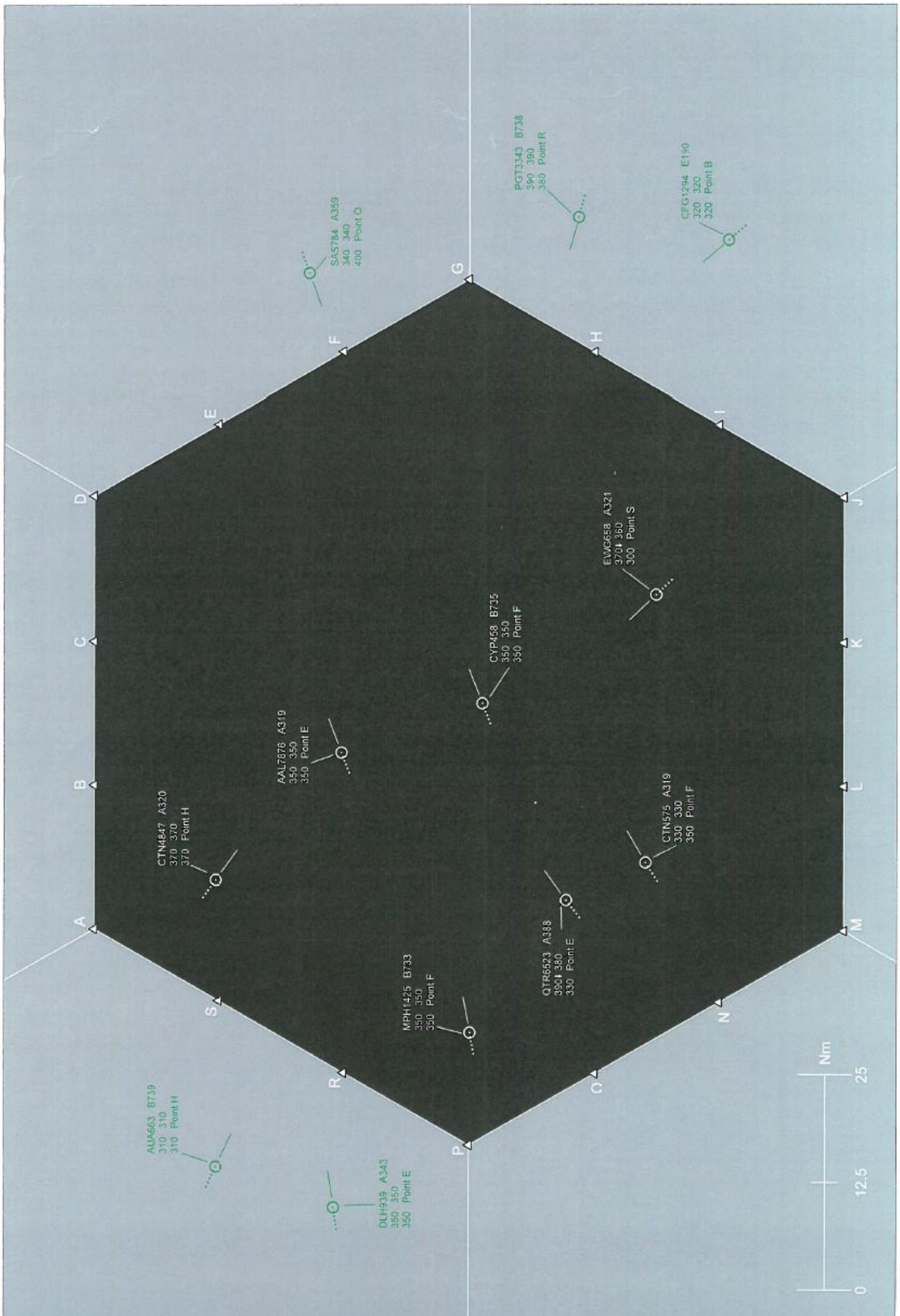
Appendices

Appendix 1 – Original 120 traffic situations

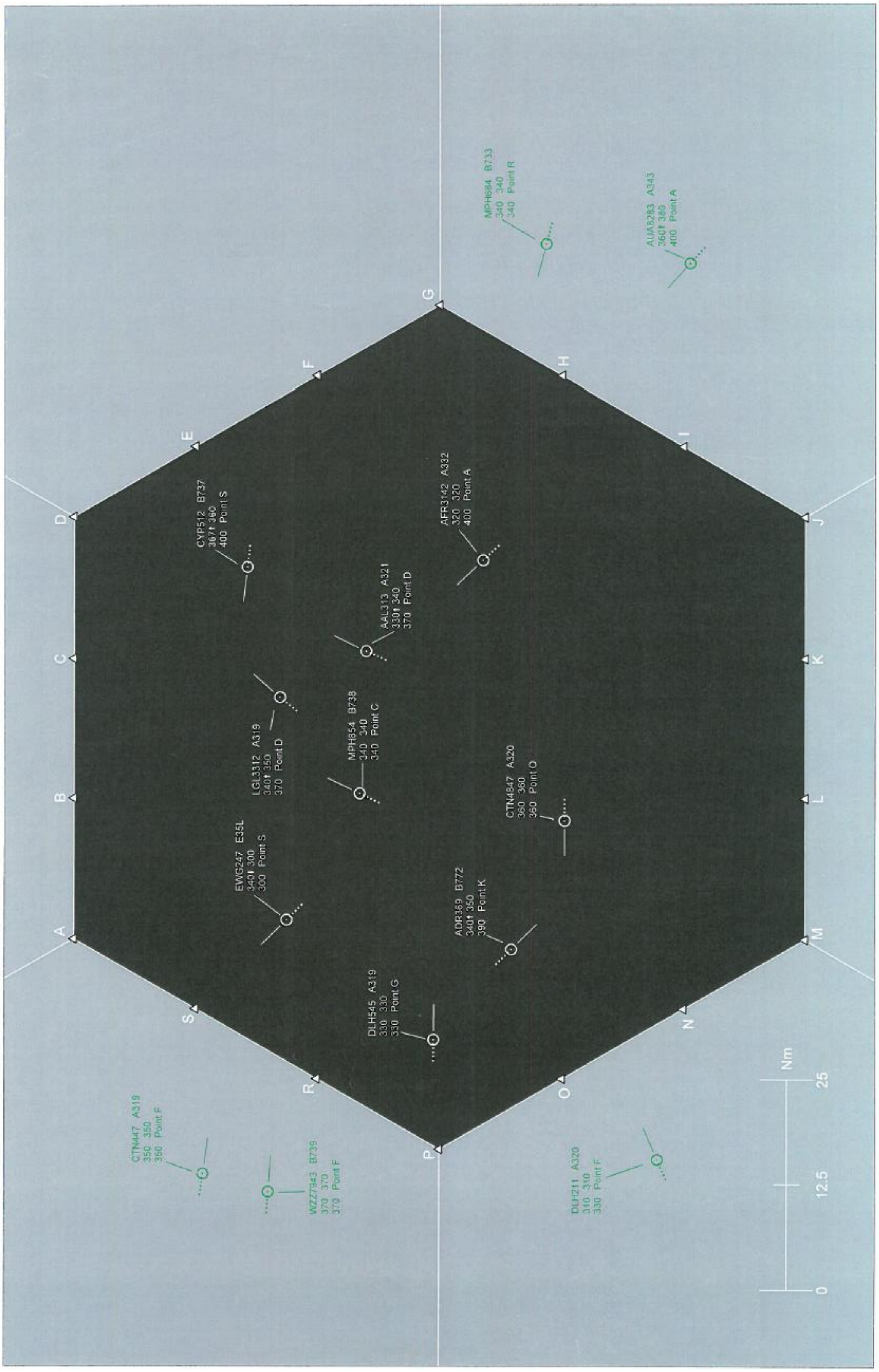
Traffic situation A1



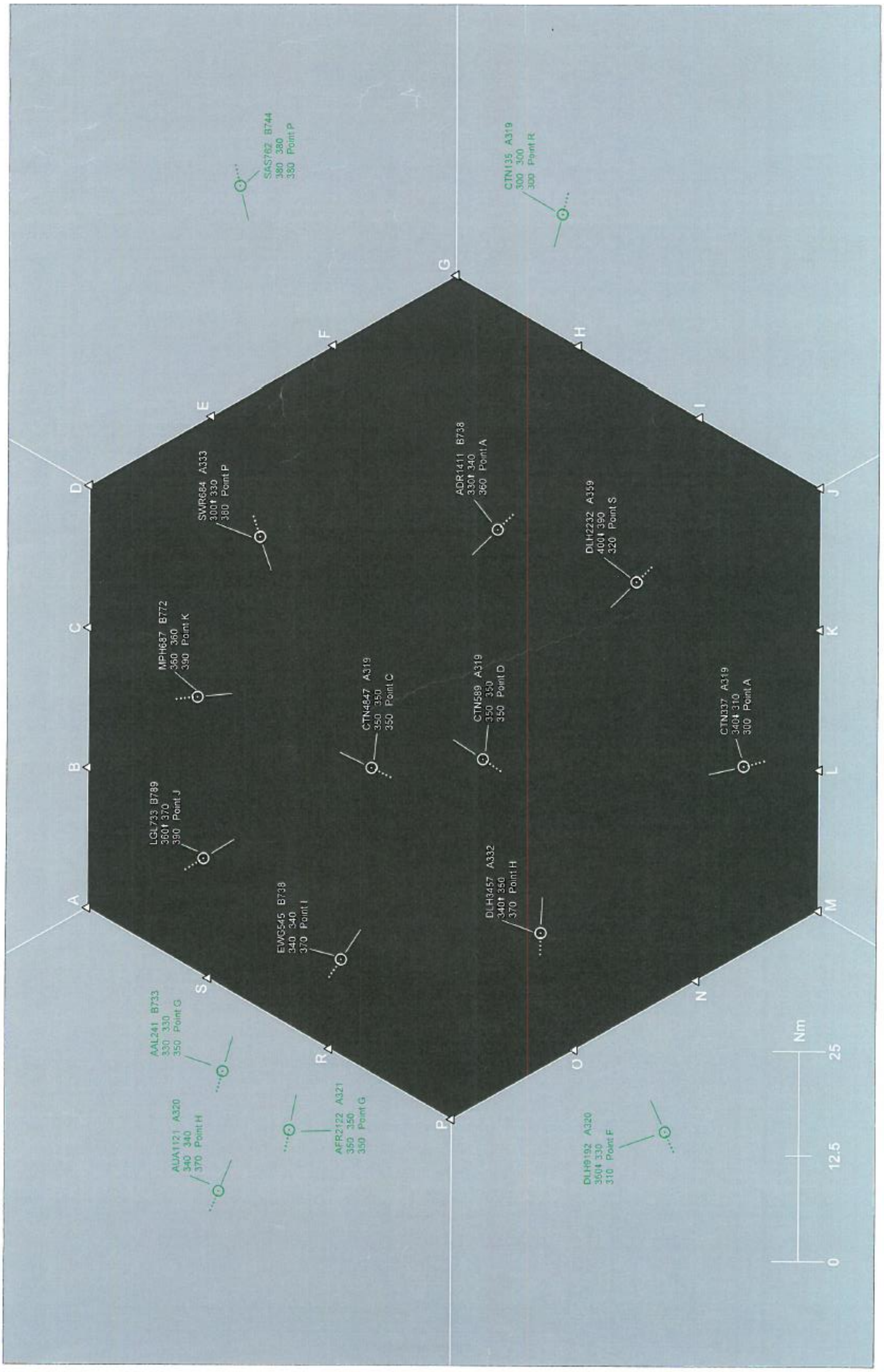
Traffic situation A2



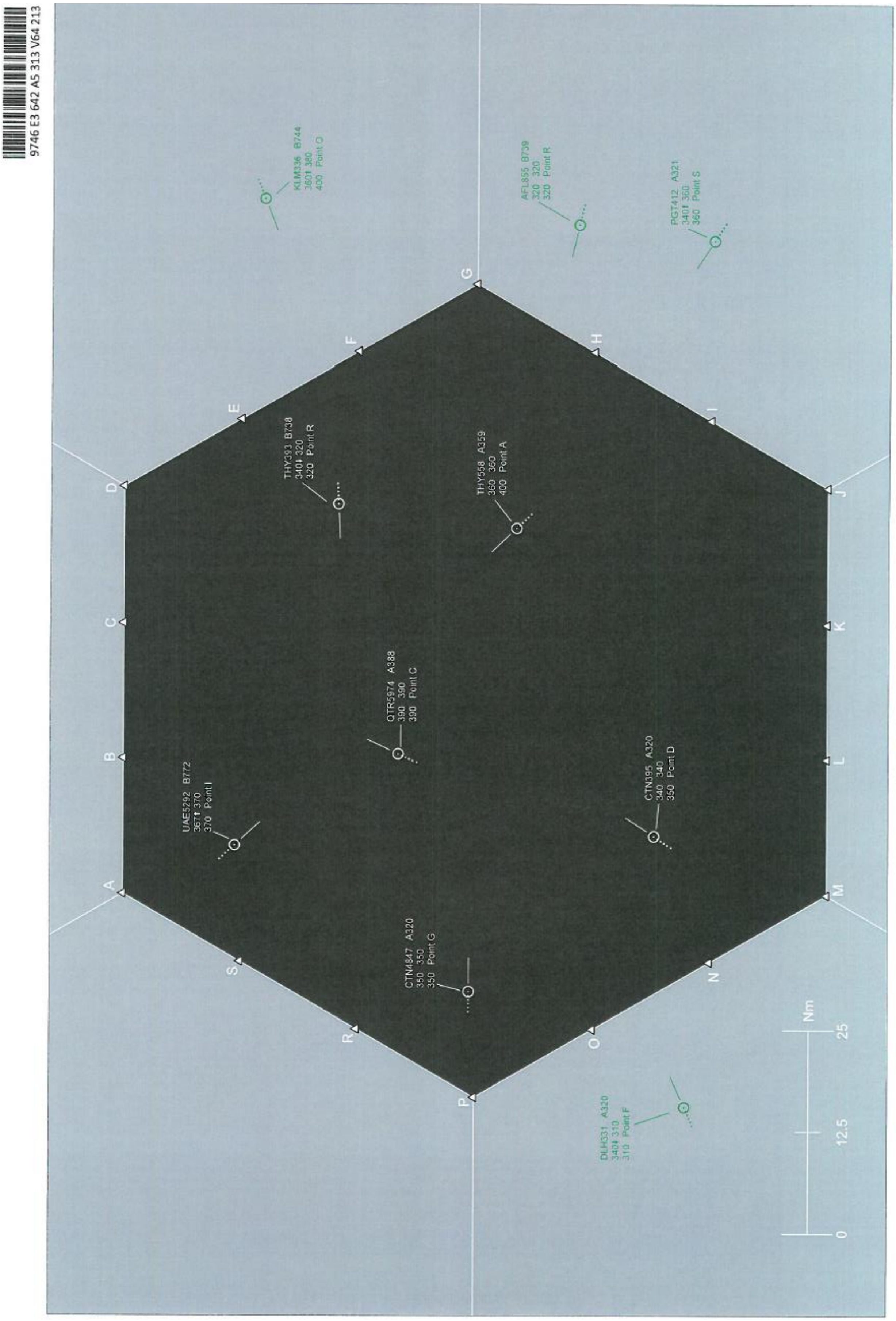
Traffic situation A3



Traffic situation A4

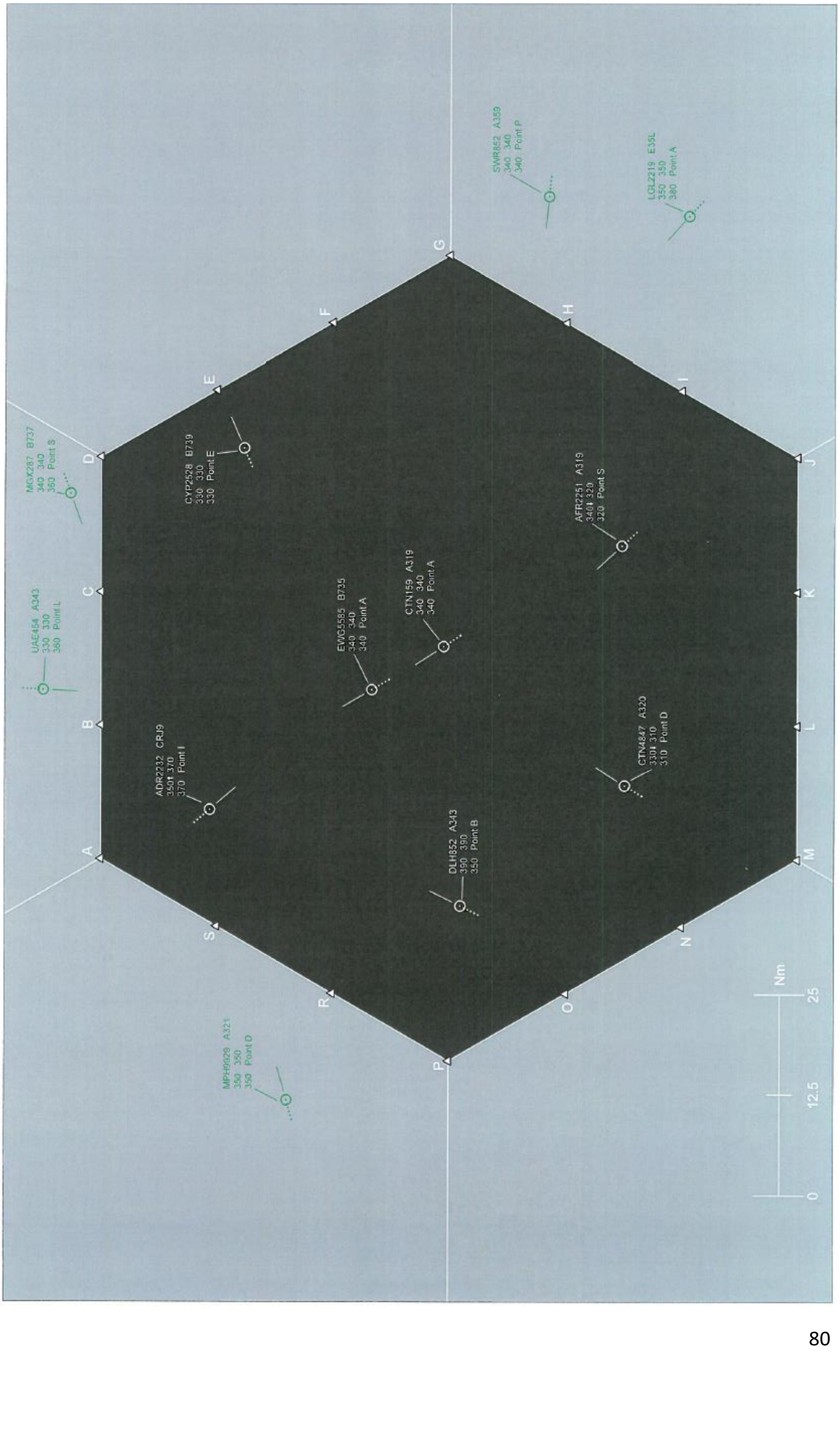


Traffic situation A5

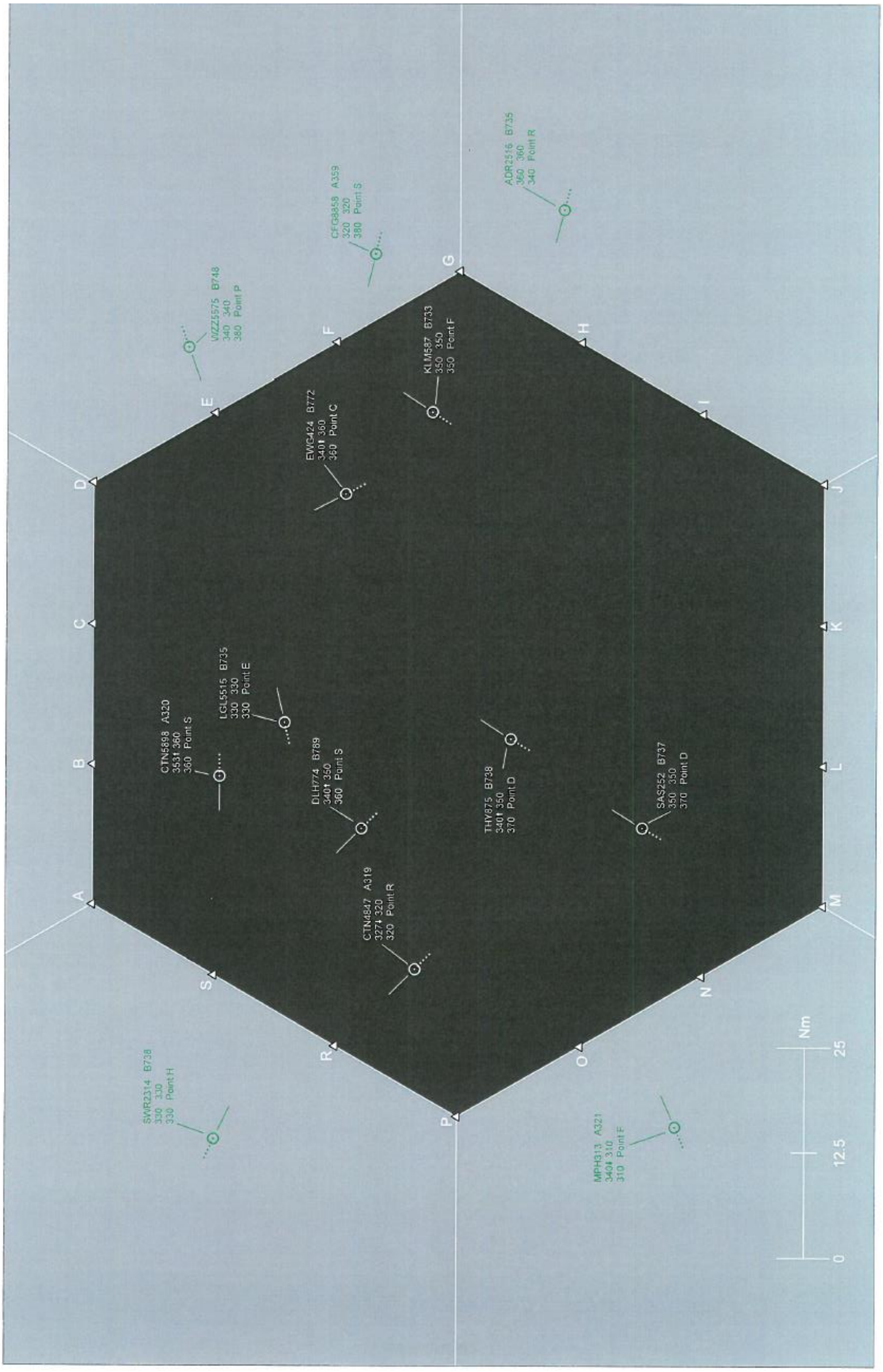


2294 X7 364 A6 294 K42 316

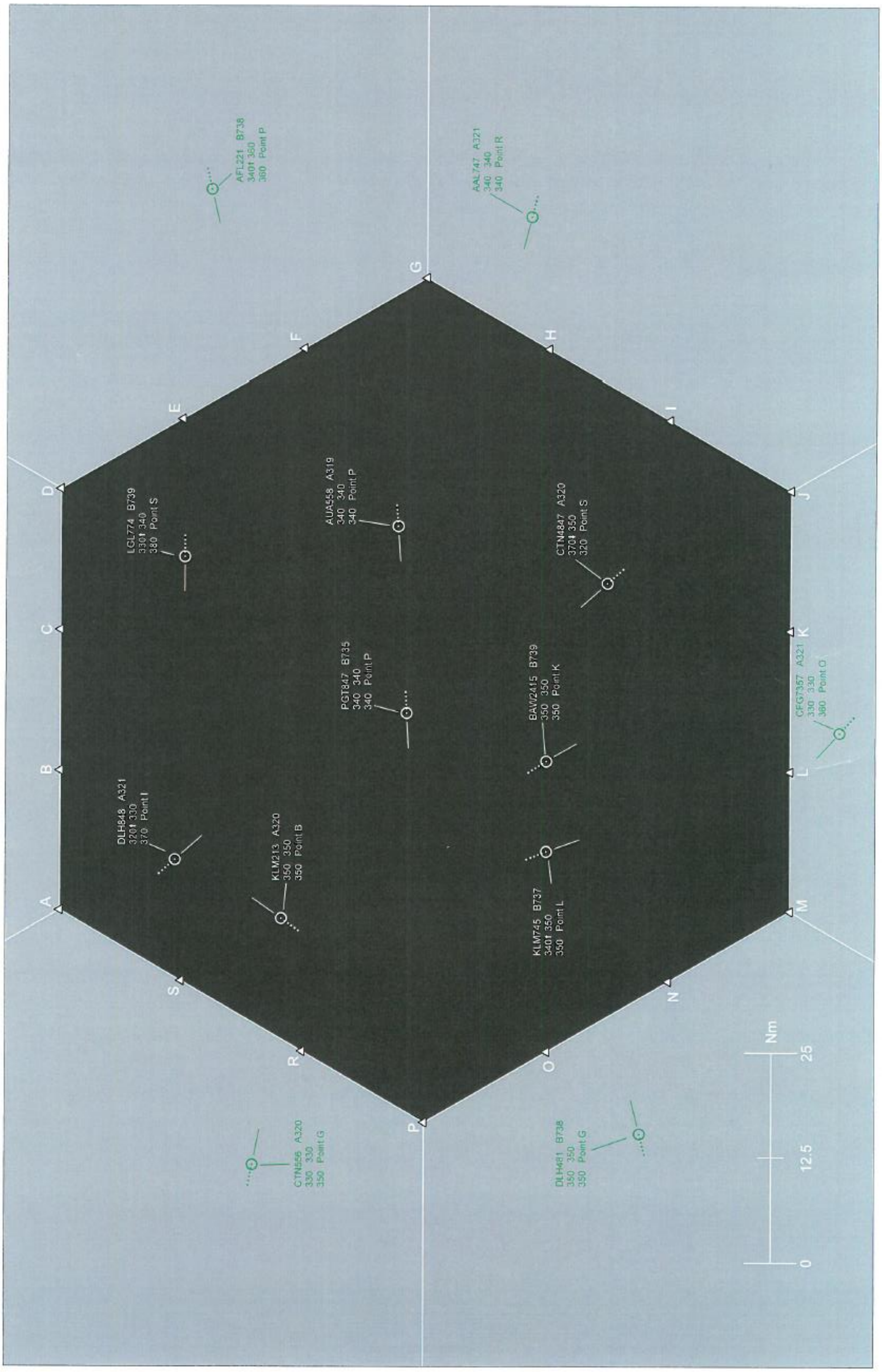
Traffic situation A6



Traffic situation A7

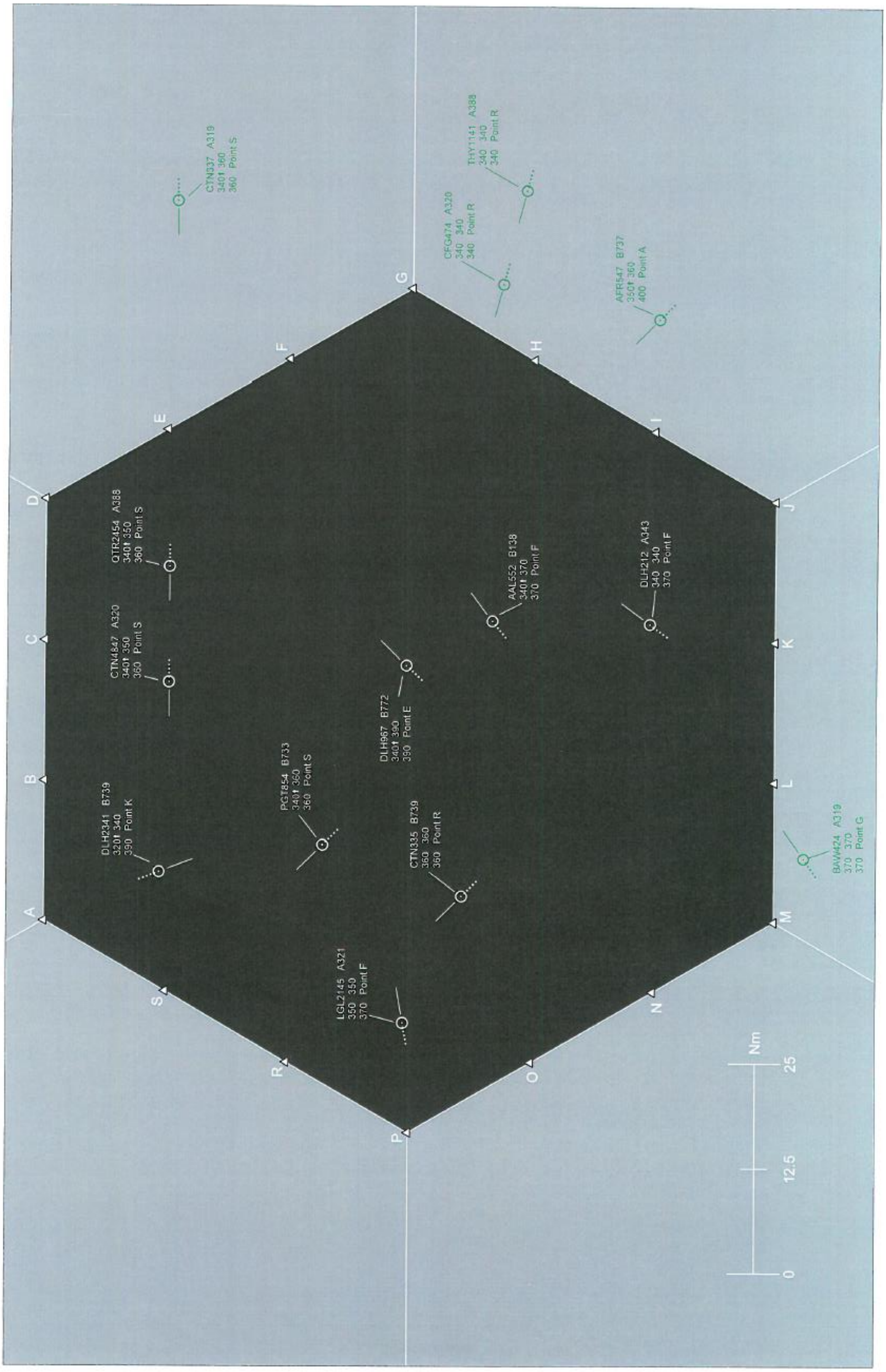


Traffic situation A8

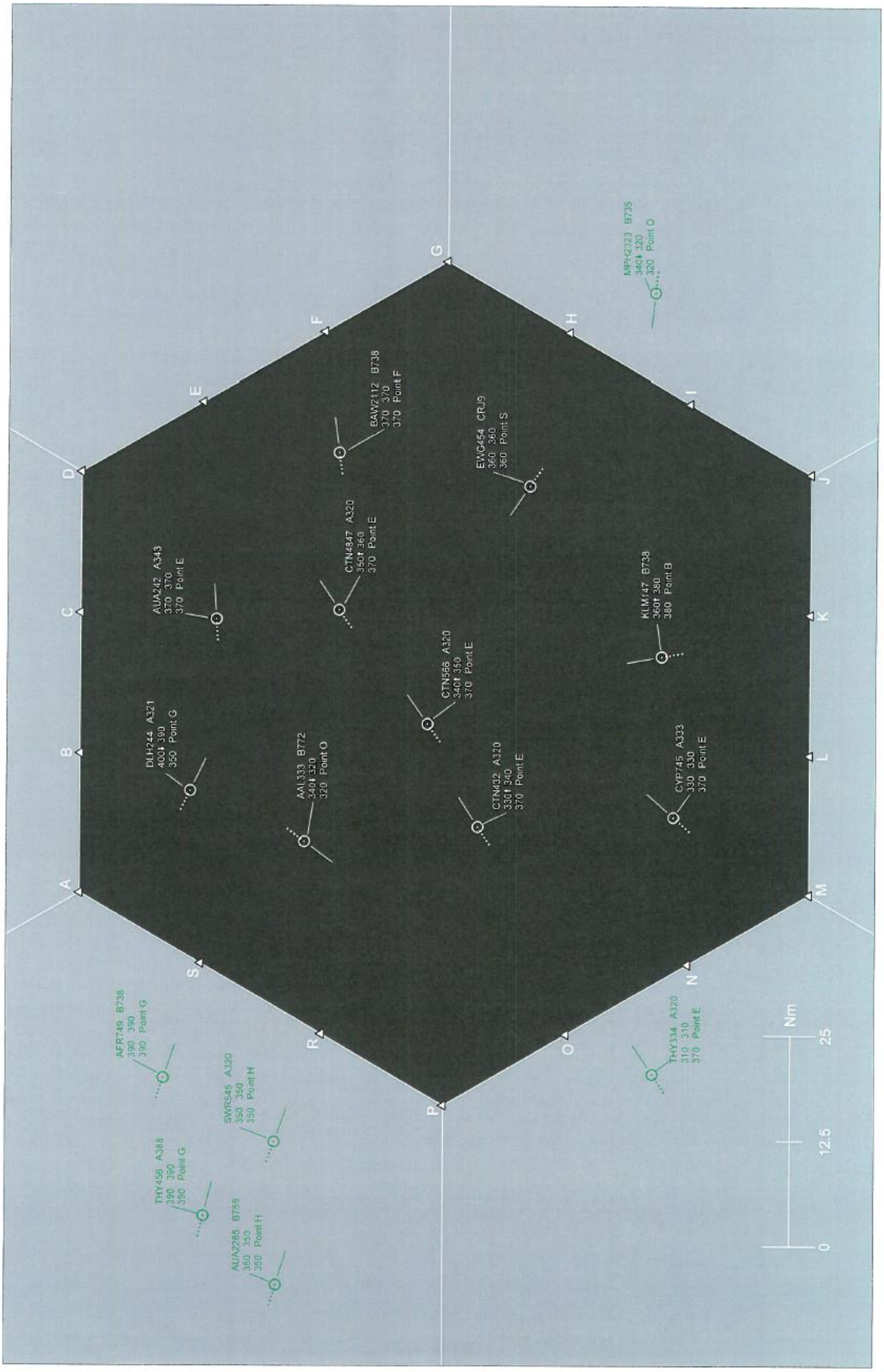


Traffic situation A9

9143 D6 757 A9 643 236 746

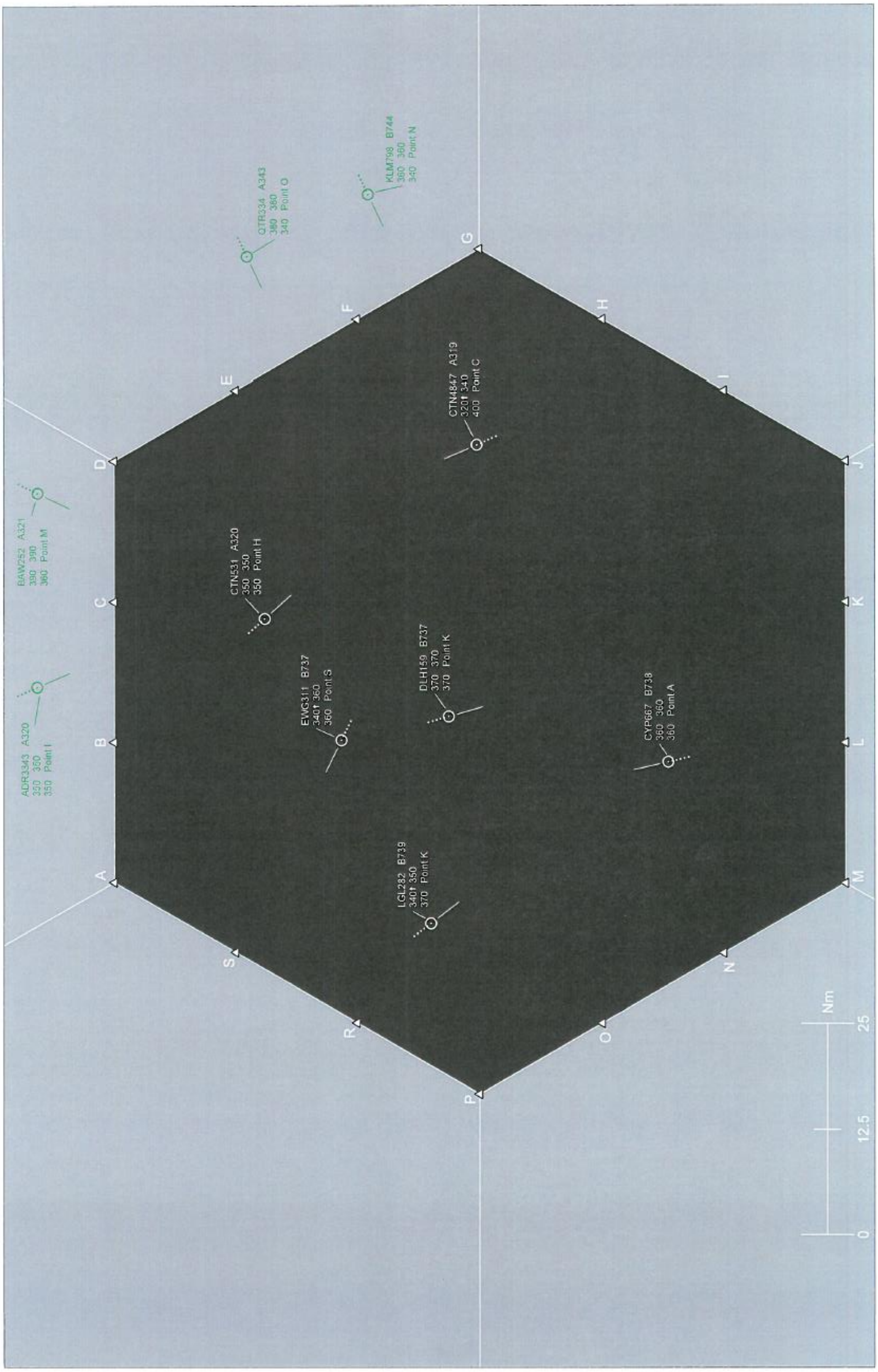


Traffic situation A10

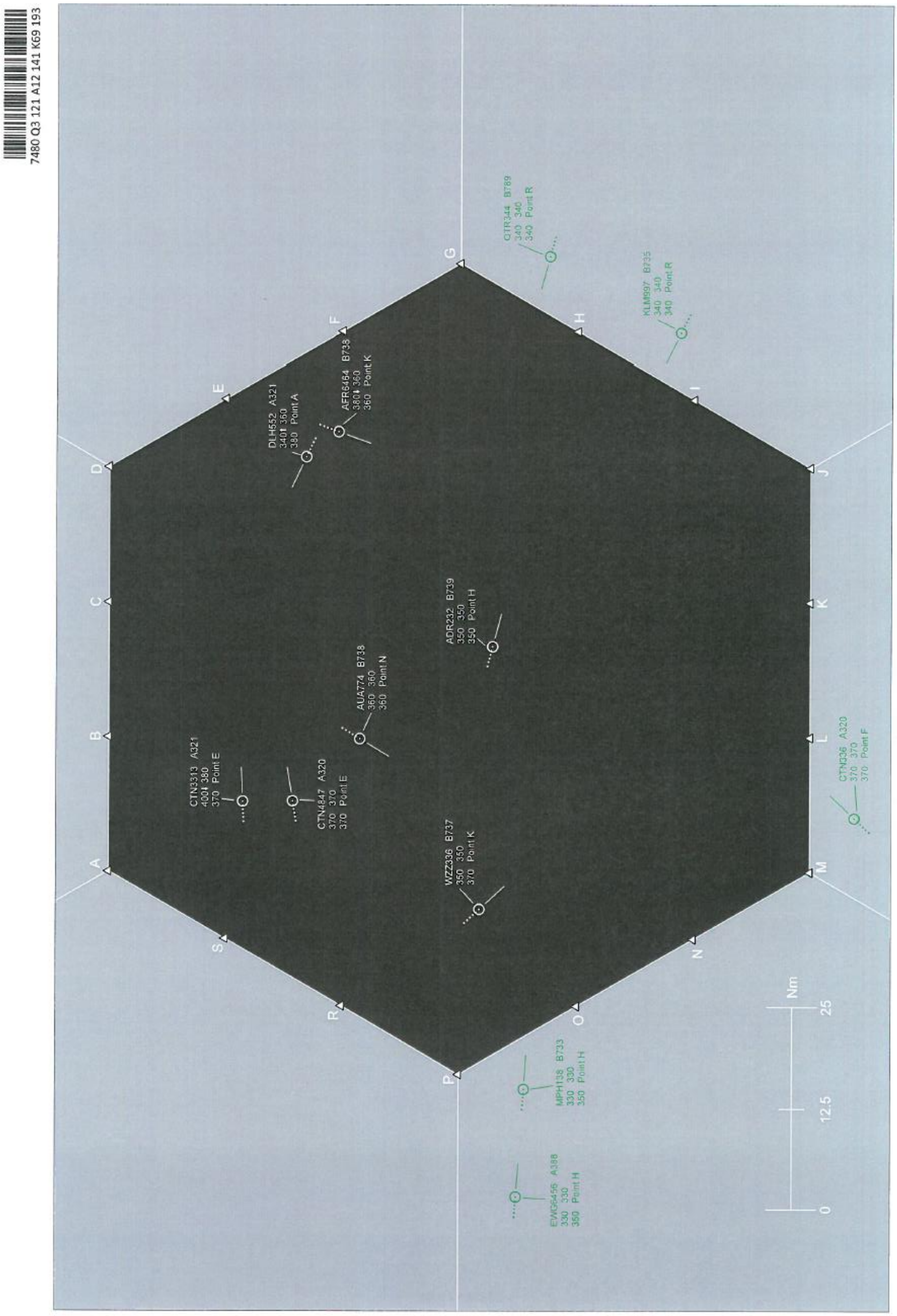


Traffic situation A11

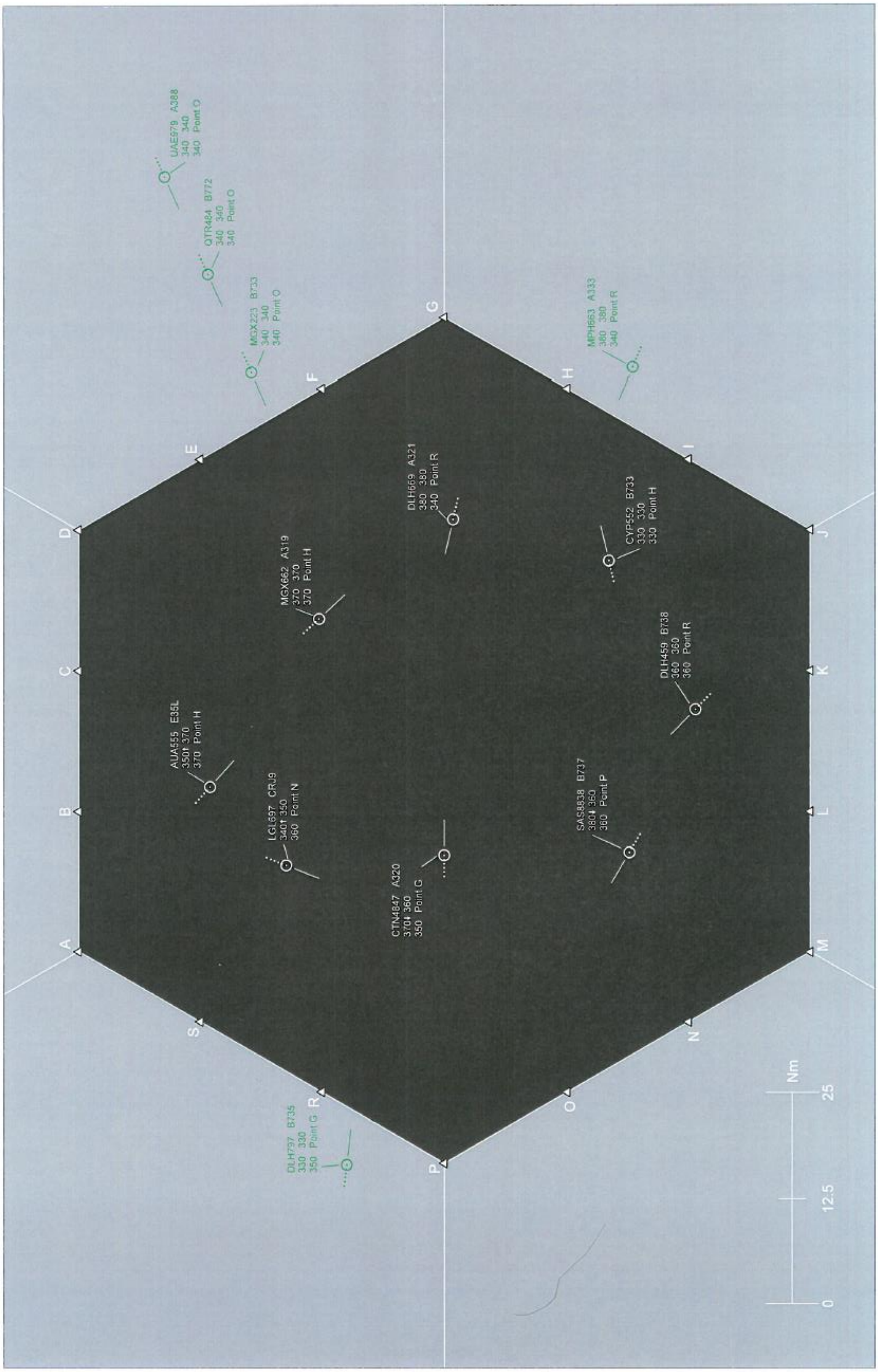
3305 U2 858 A11 340 Z29 333



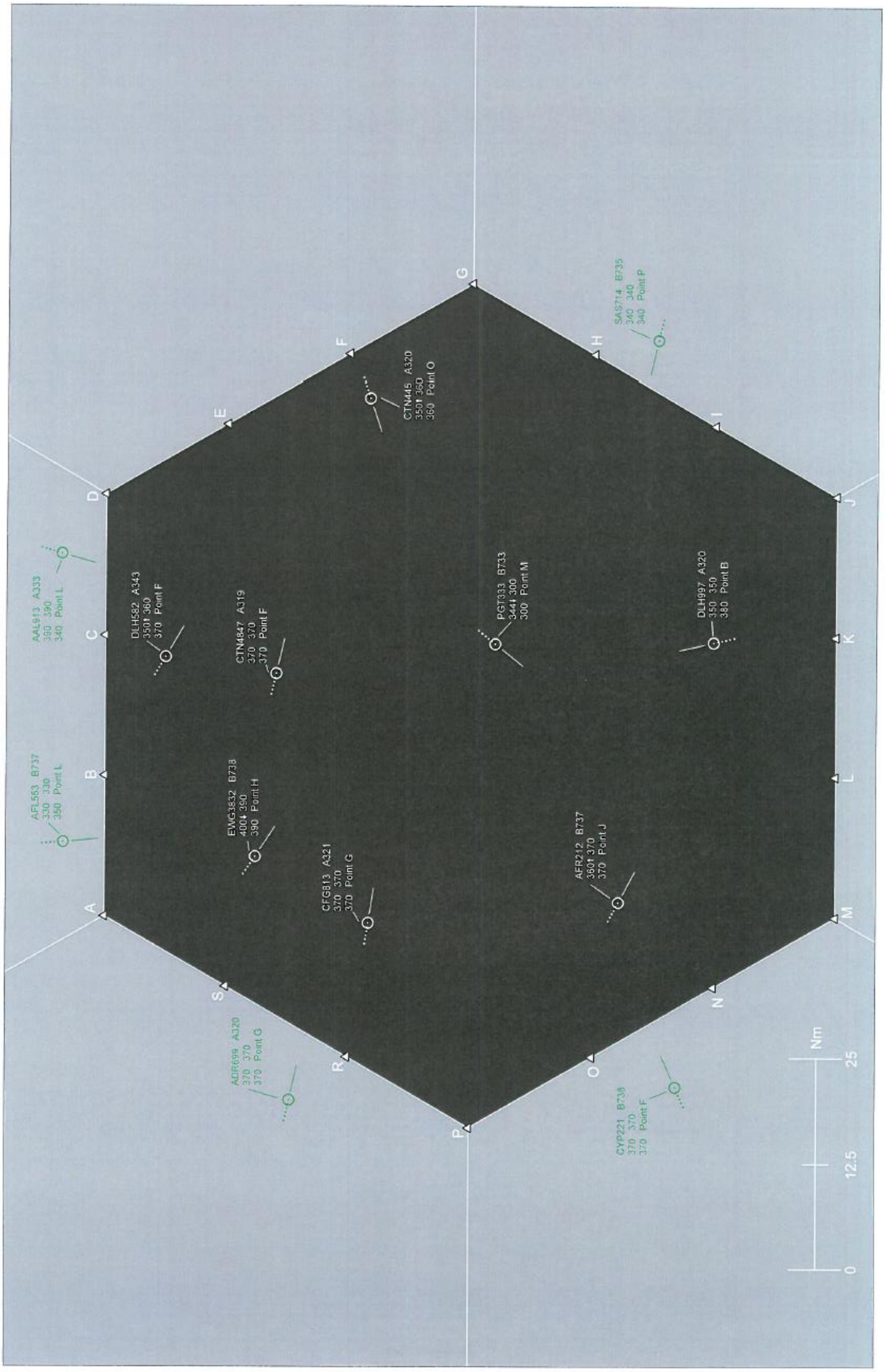
Traffic situation A12



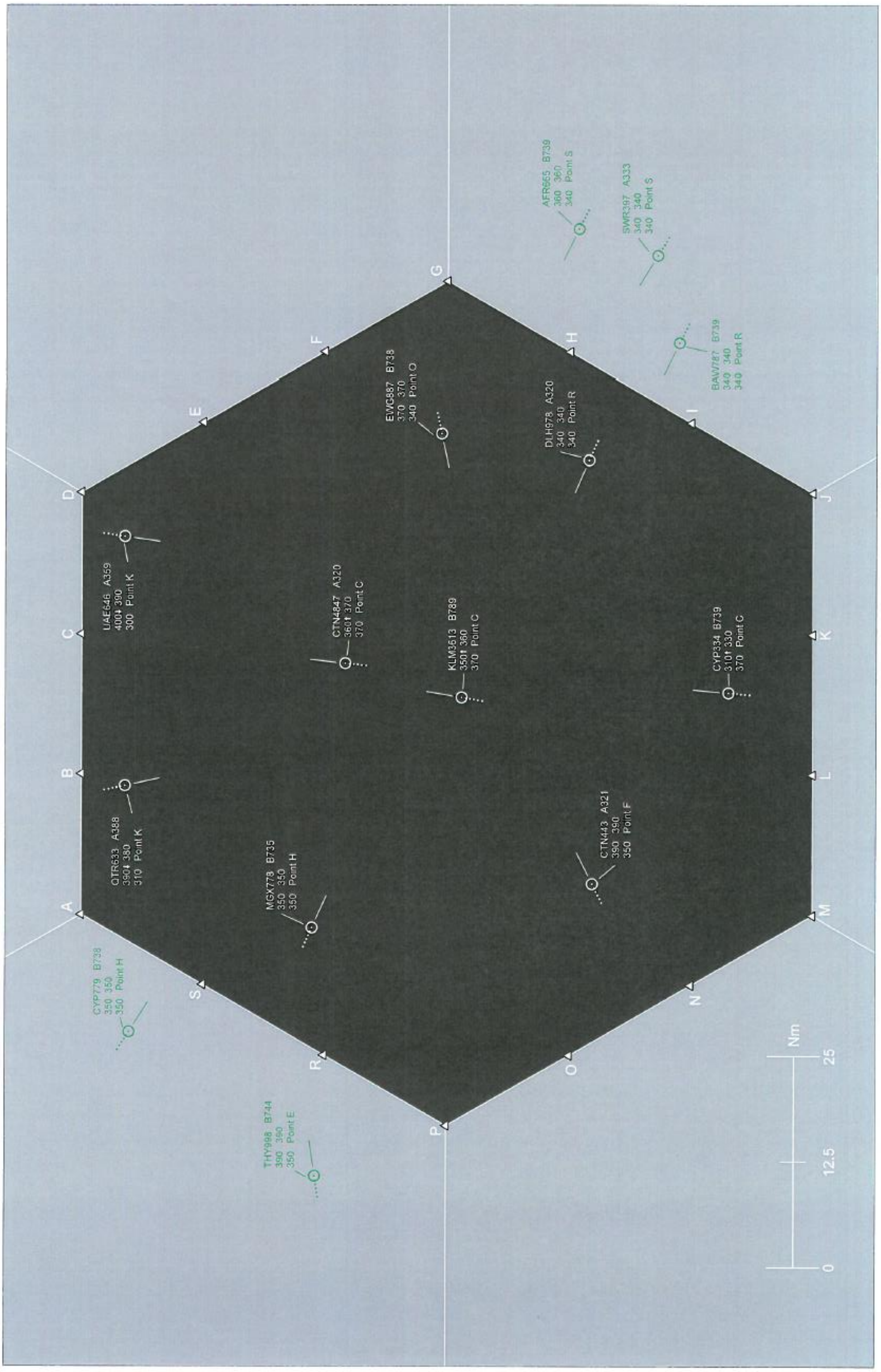
Traffic situation A13



Traffic situation A14

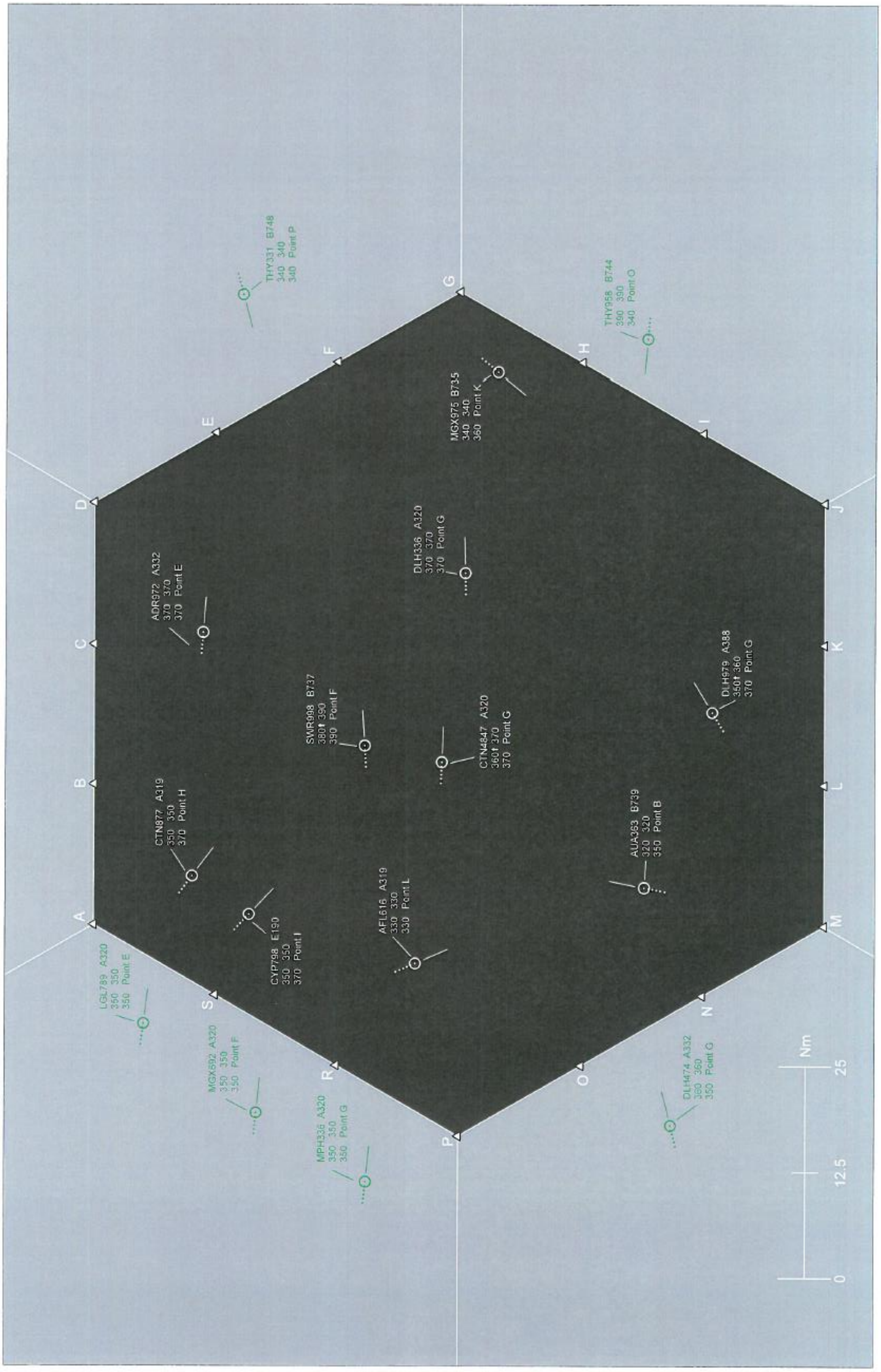


Traffic situation A15



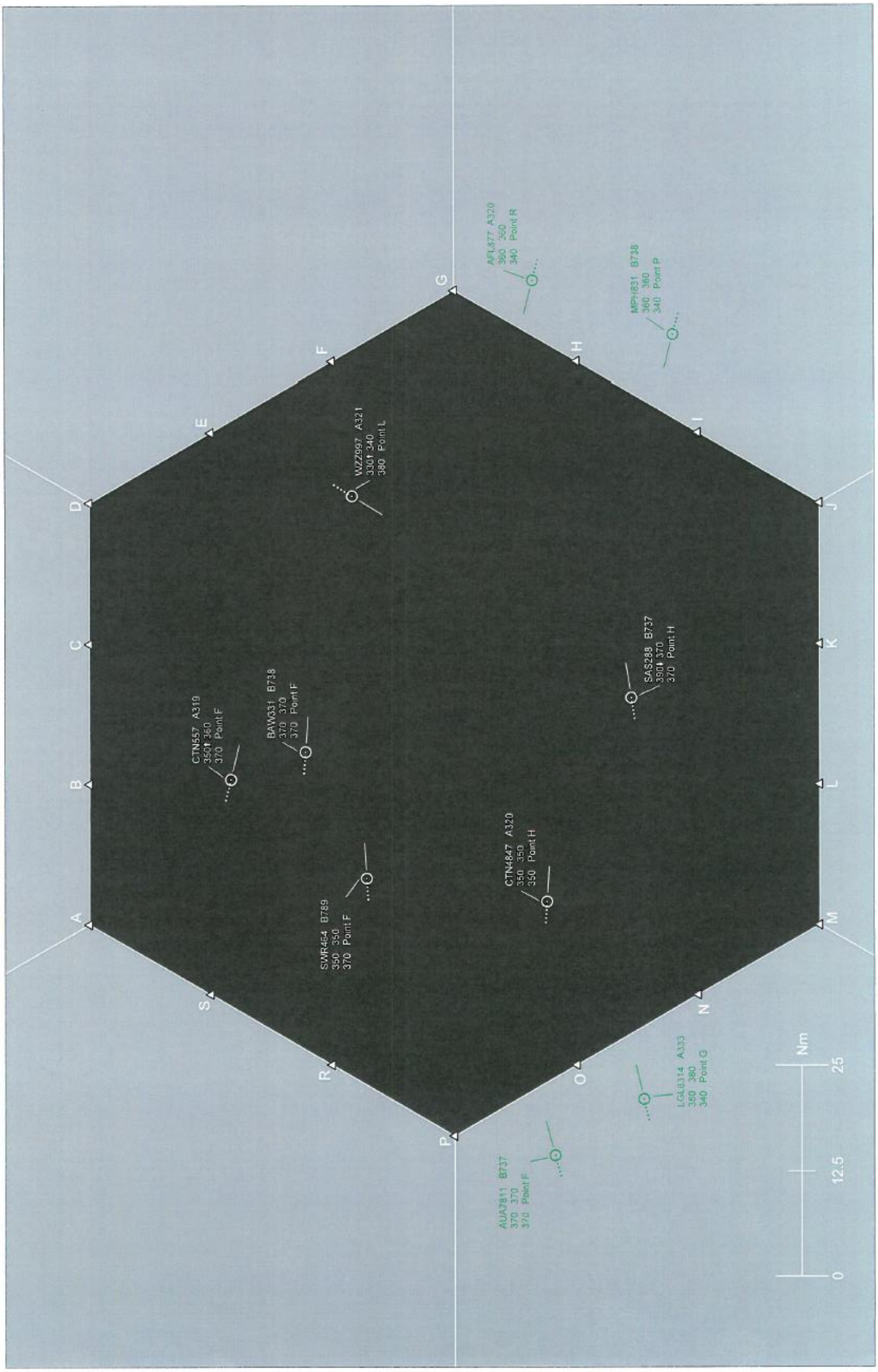
3194 T3 282 A16 371 Z72 787

Traffic situation A16



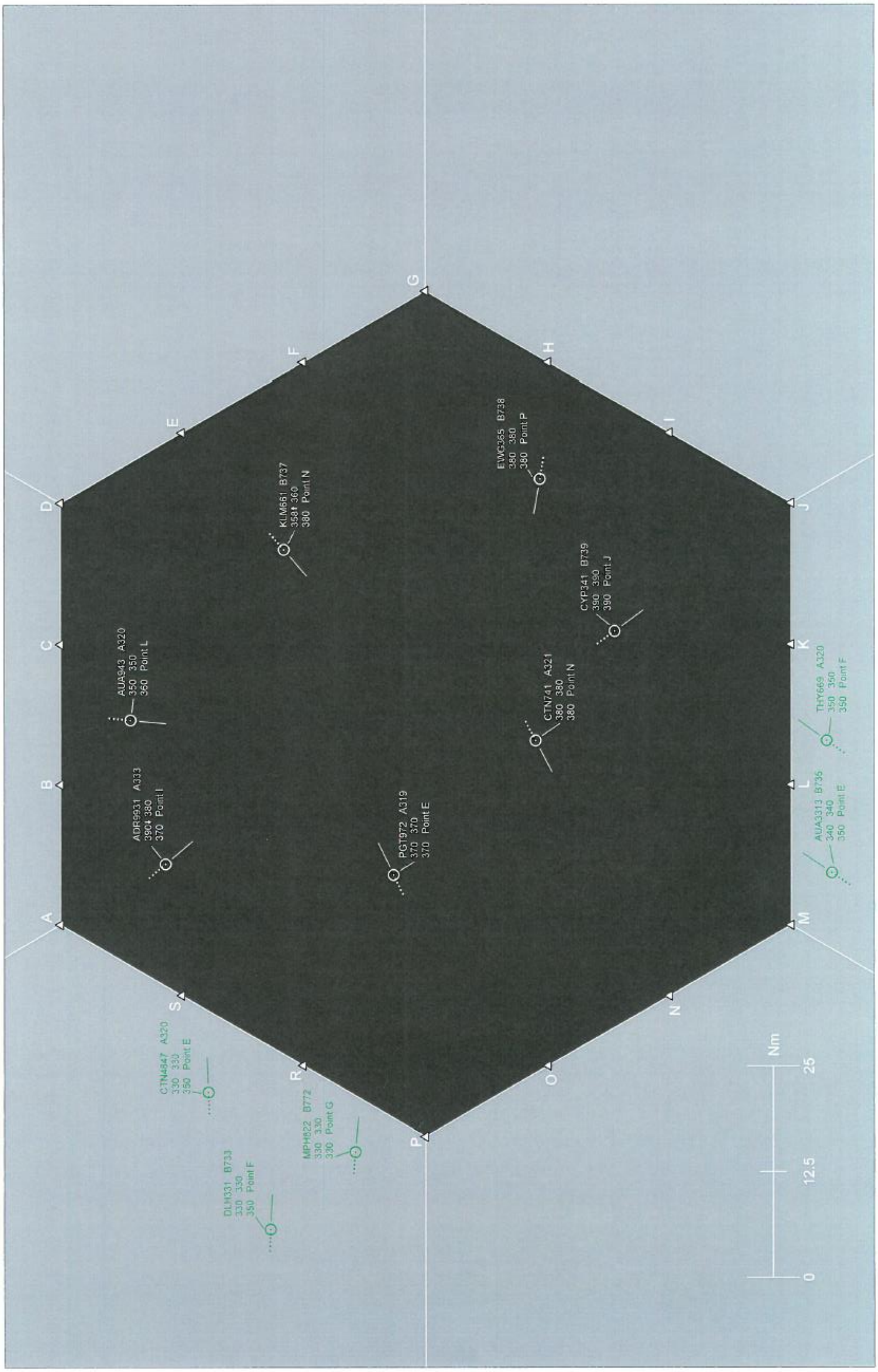
454 F5 464 A17 564 M55 545

Traffic situation A17



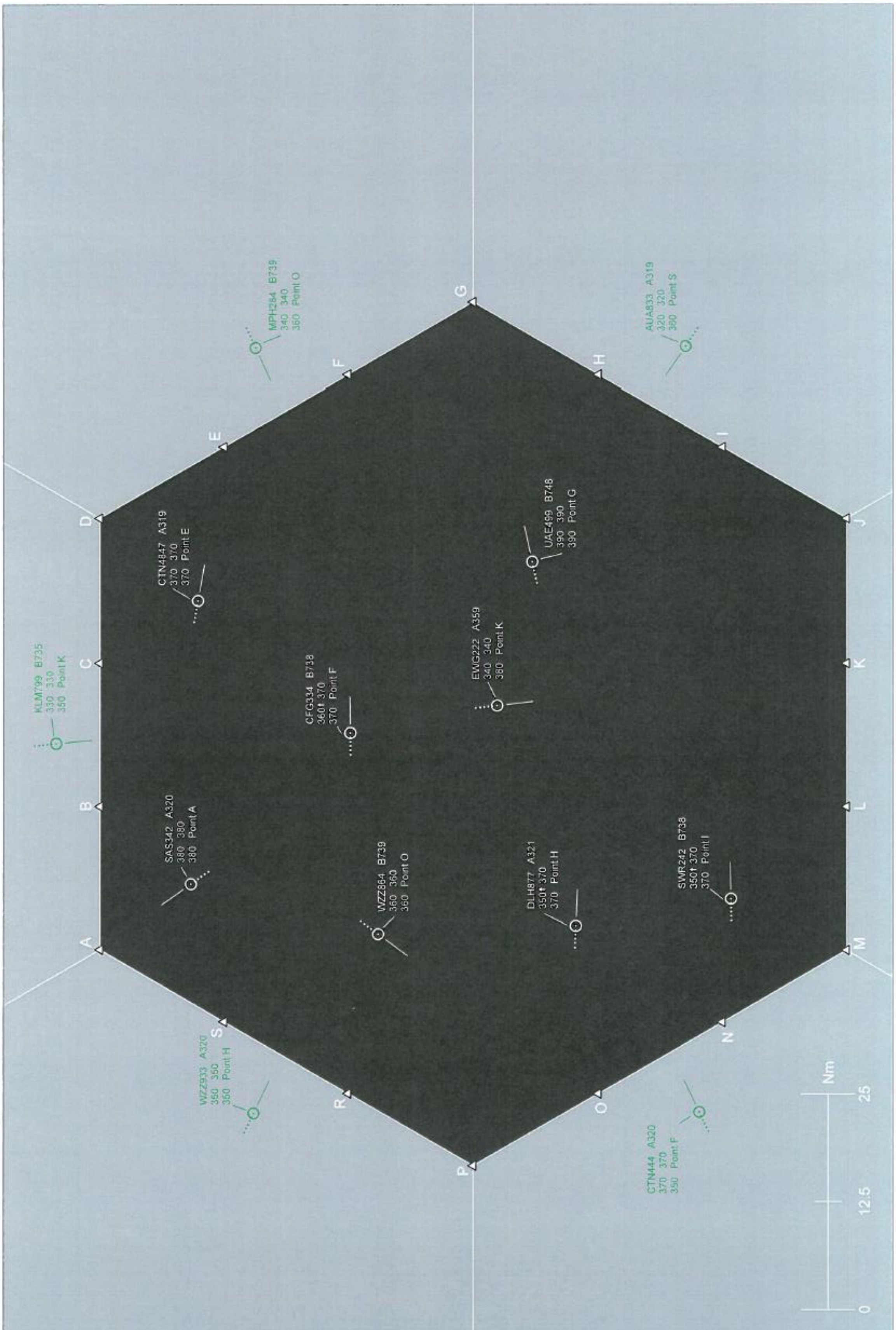
783 G7 681 A18 318 H97 824

Traffic situation A18



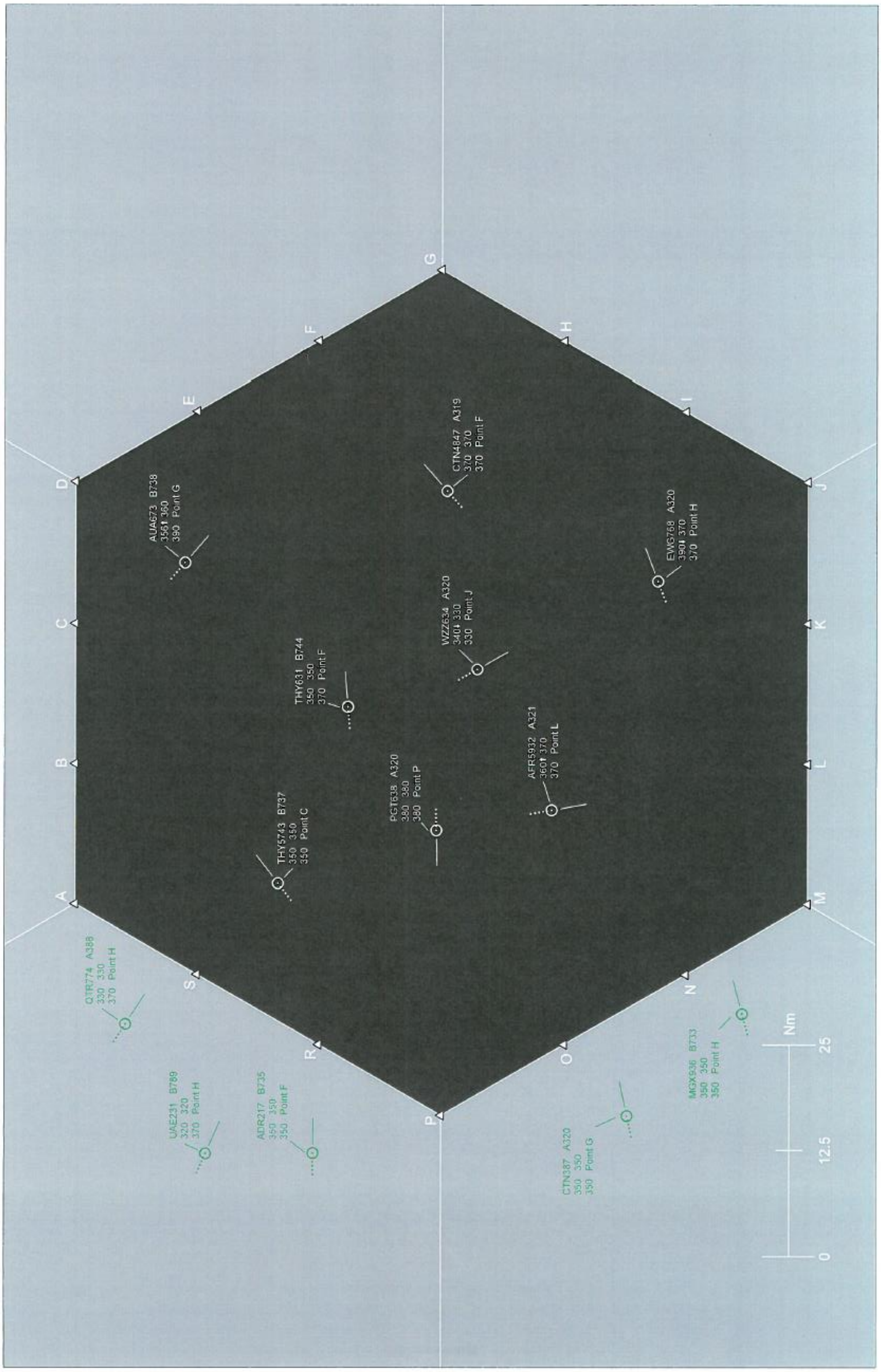
197 U3 145 A19 974 P14 249

Traffic situation A19

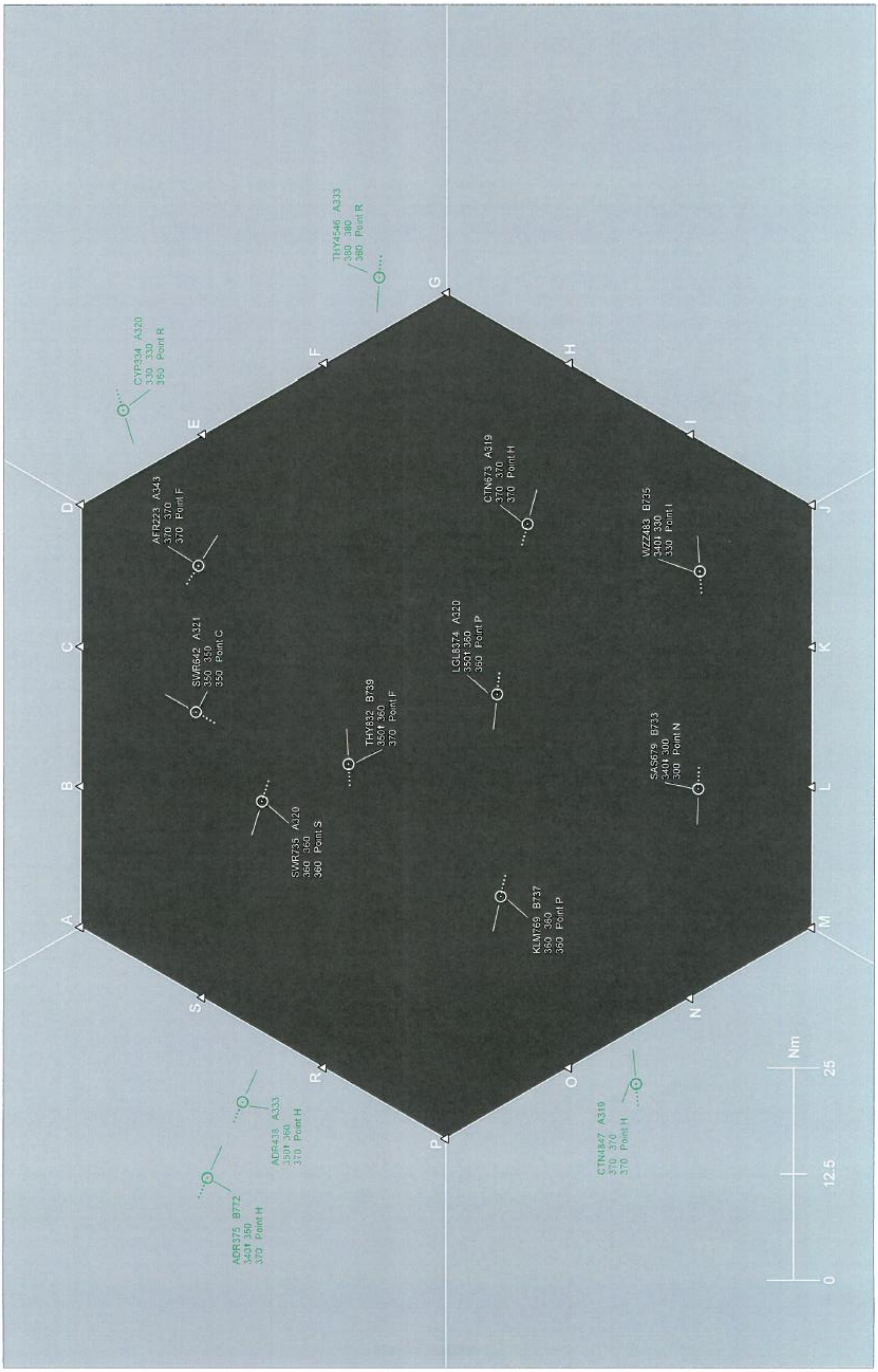


Traffic situation A20

6874 J8 349 A20 971 N74 647

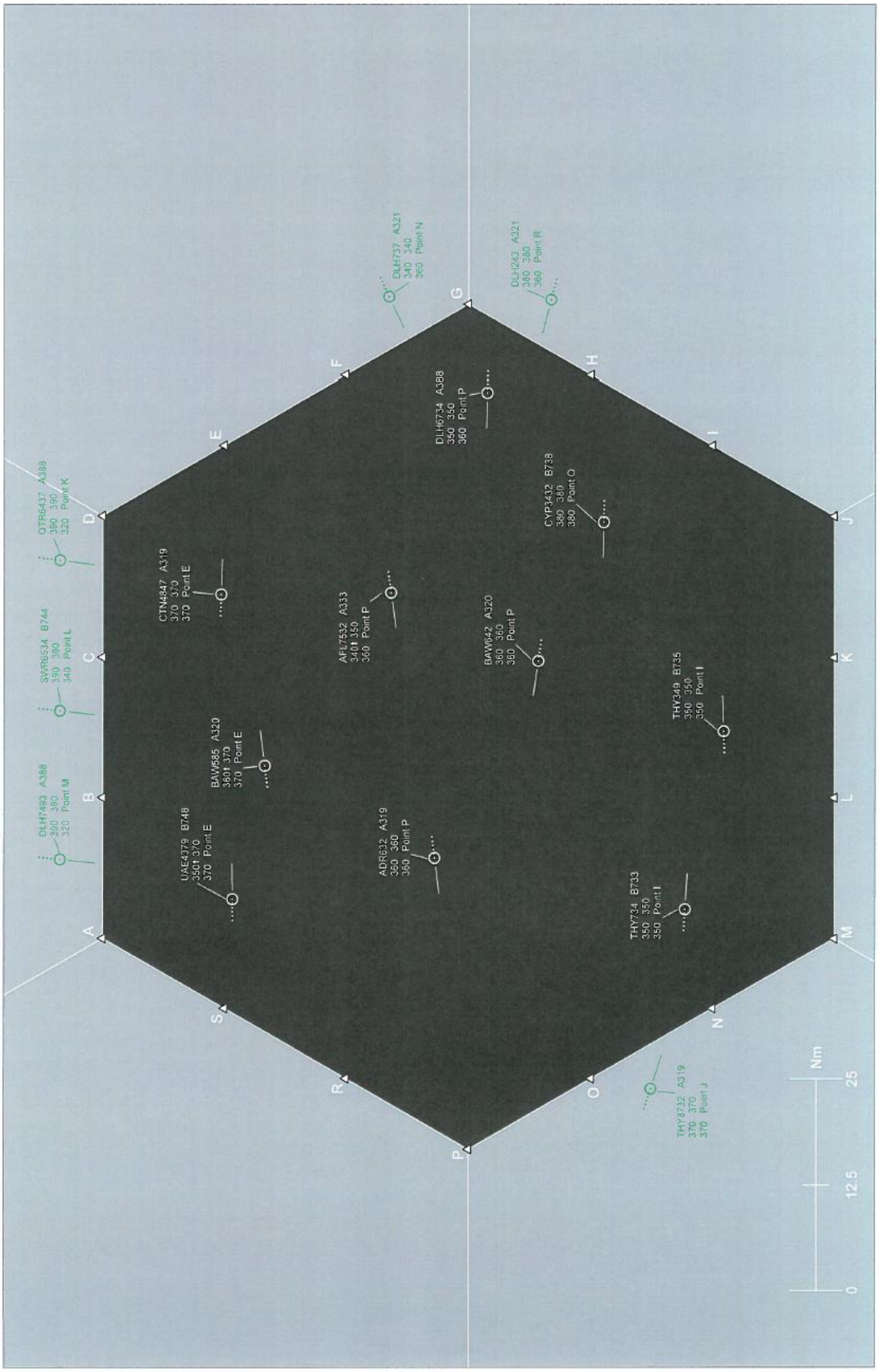


Traffic situation A21

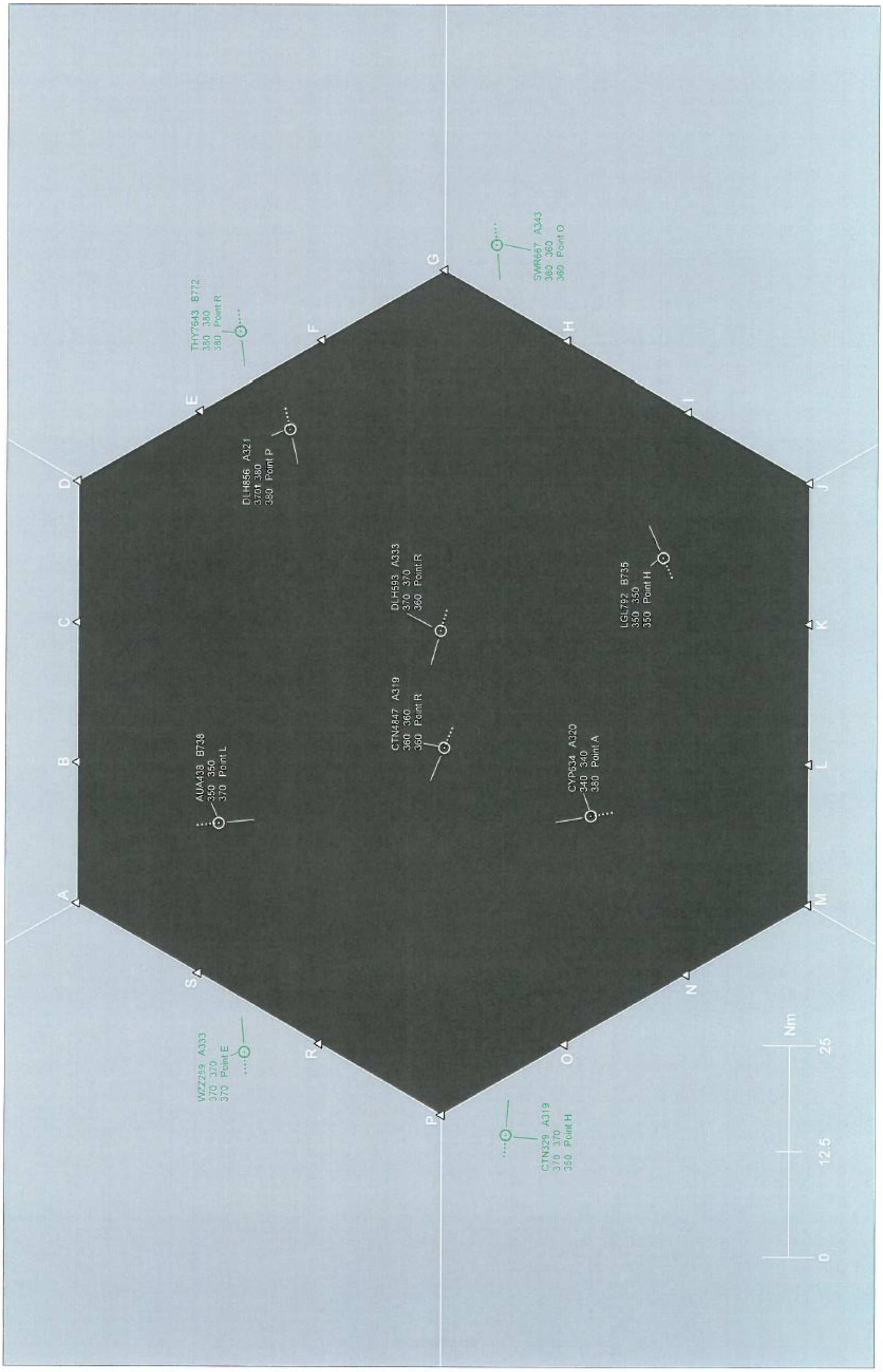


Traffic situation A22

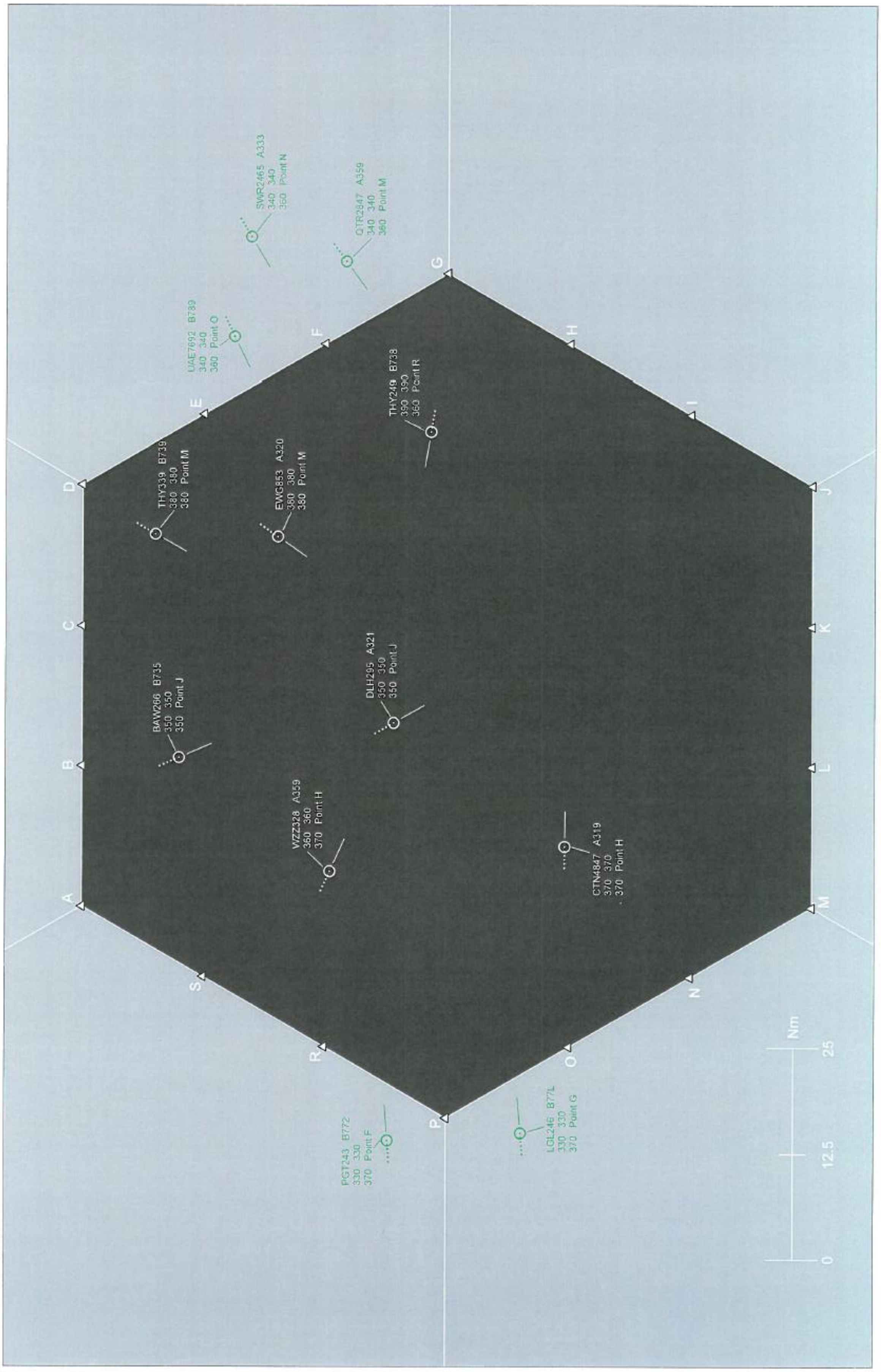
2294 X7 364 A22 294 K42 316



Traffic situation A23

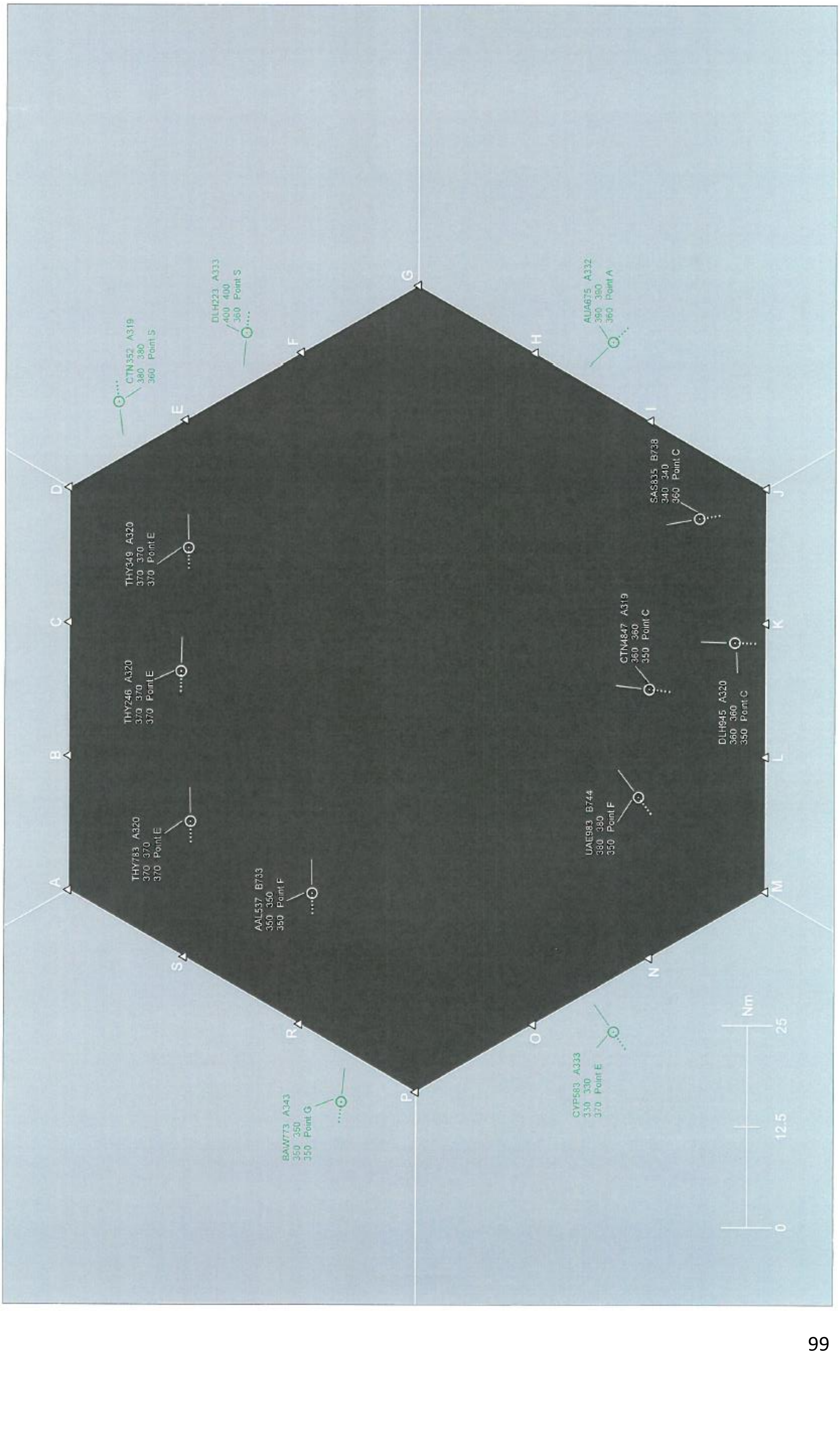


Traffic situation A24



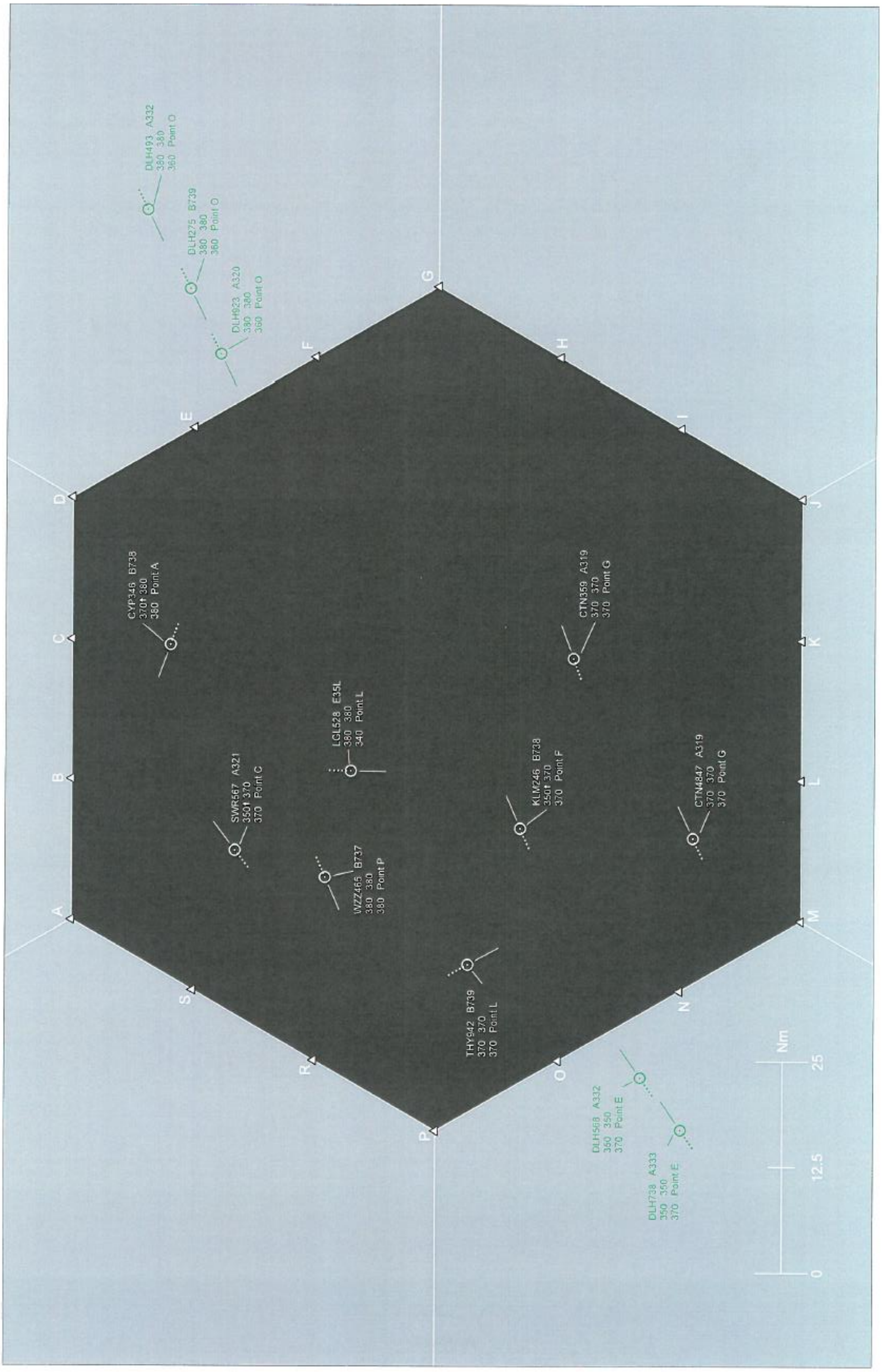
2294 X7 364 A25 294 K42 316

Traffic situation A25



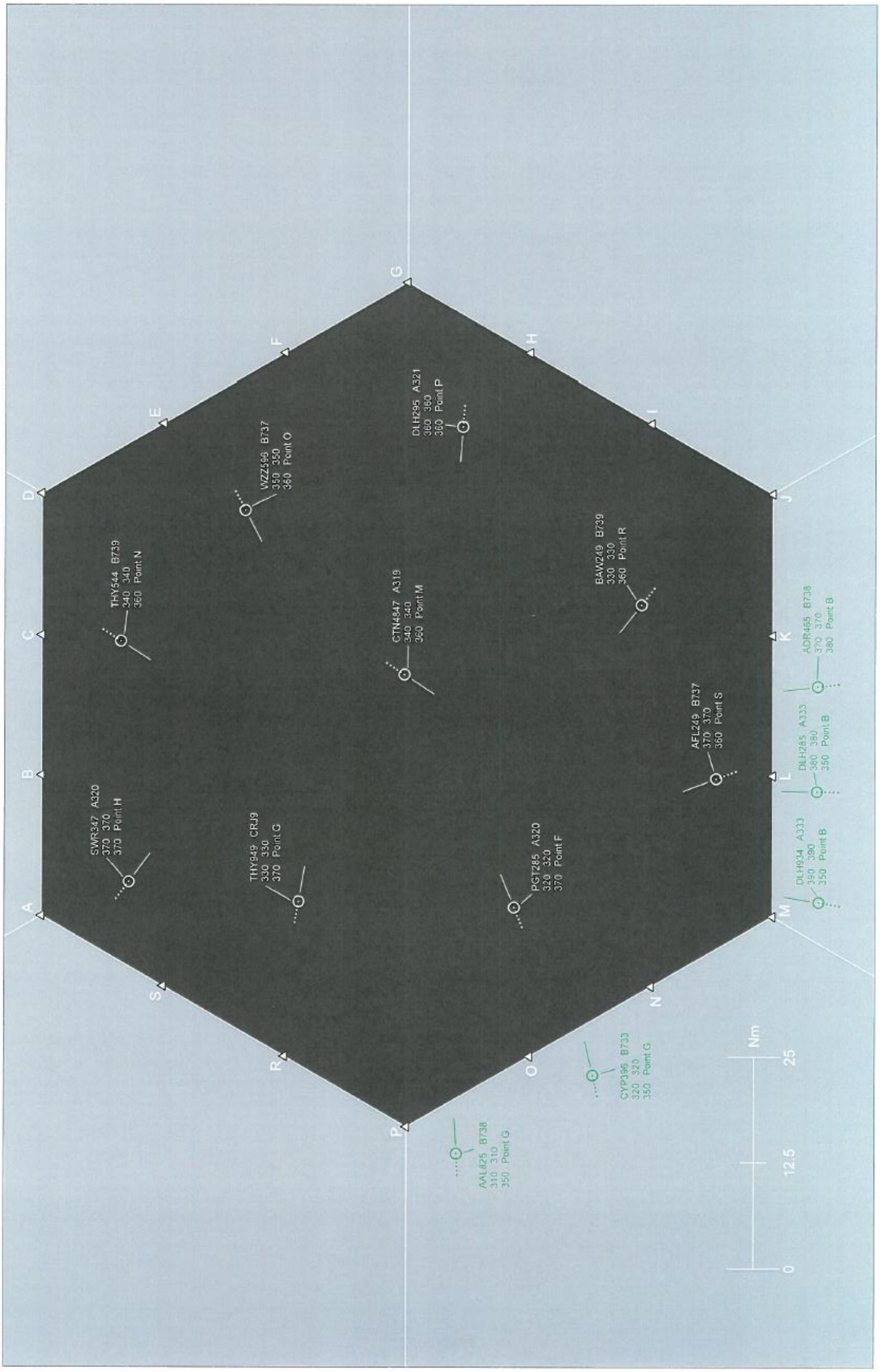
Traffic situation A26

9746 E3 642 A26 313 V64 213

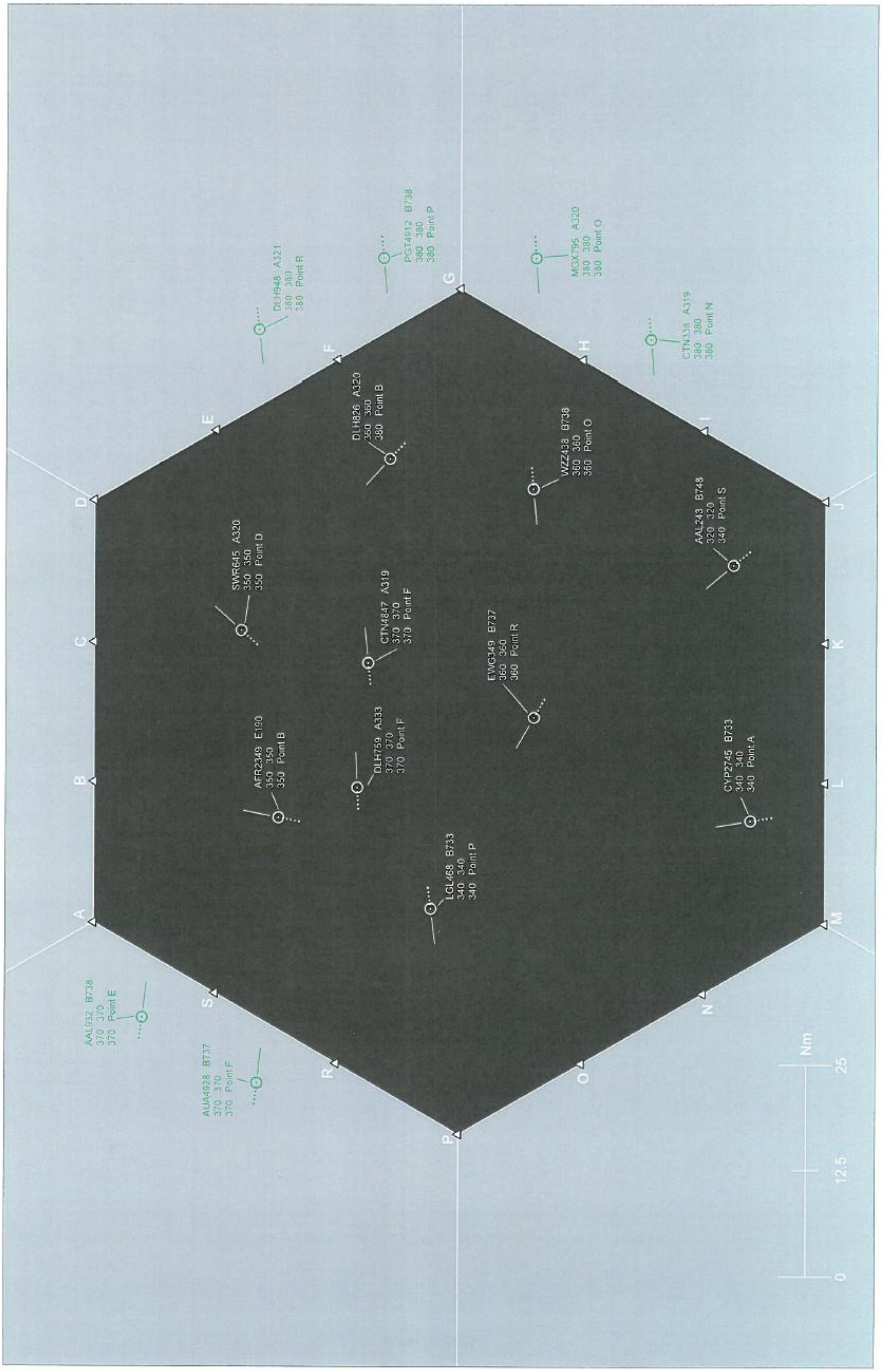


Traffic situation A27

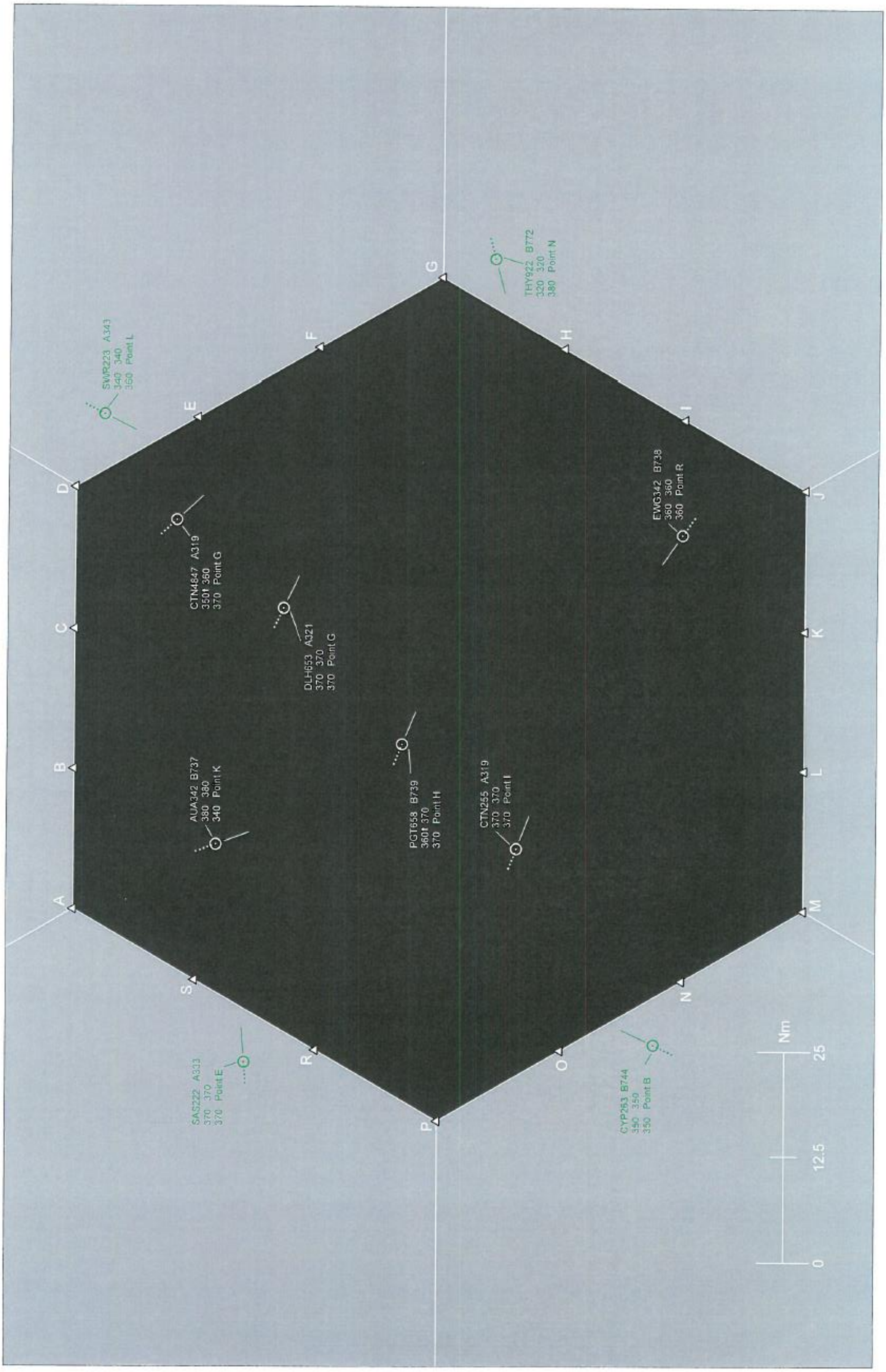
6874 J8 349 A27 971 N74 647



Traffic situation A28

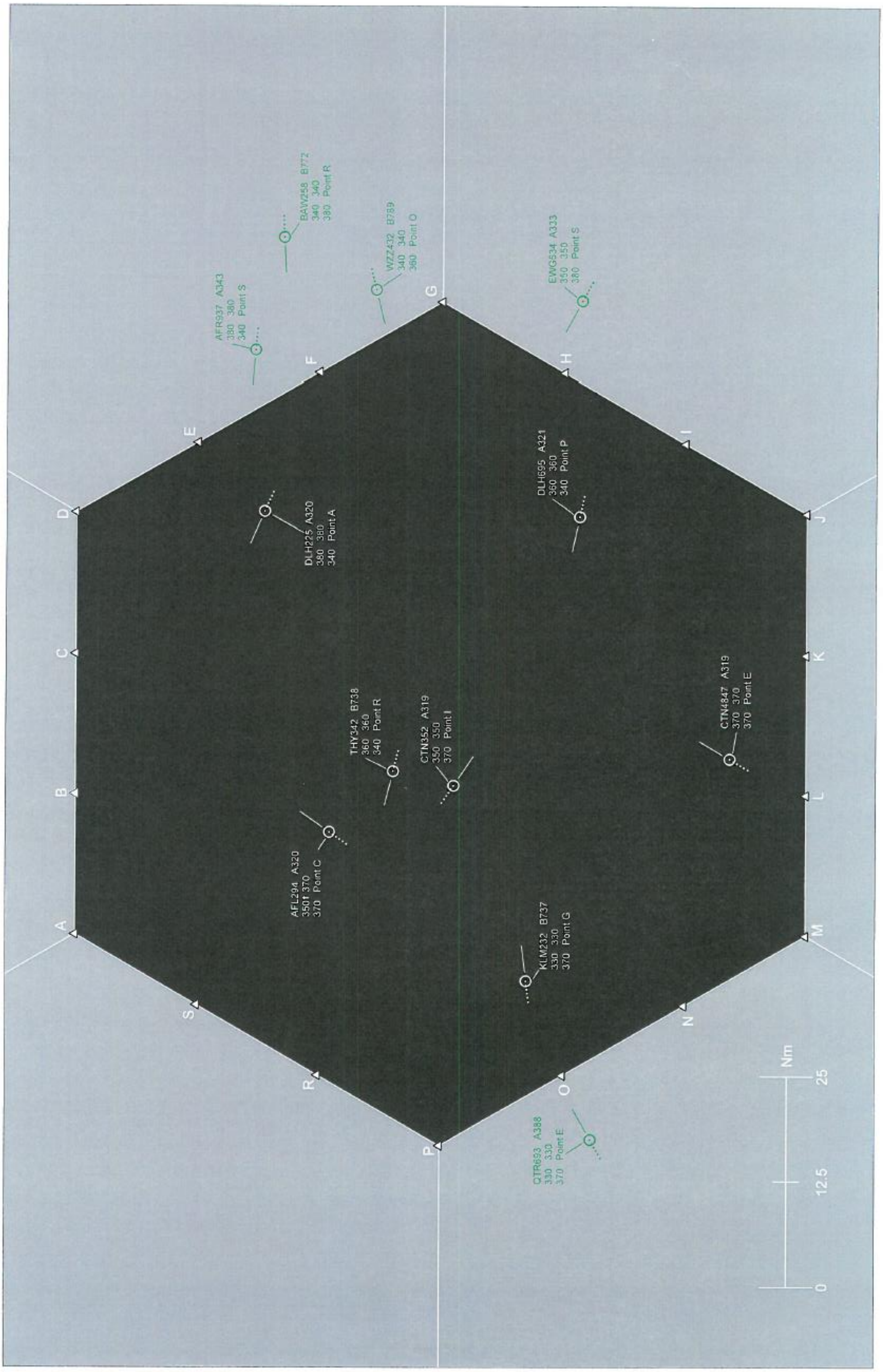


Traffic situation A29



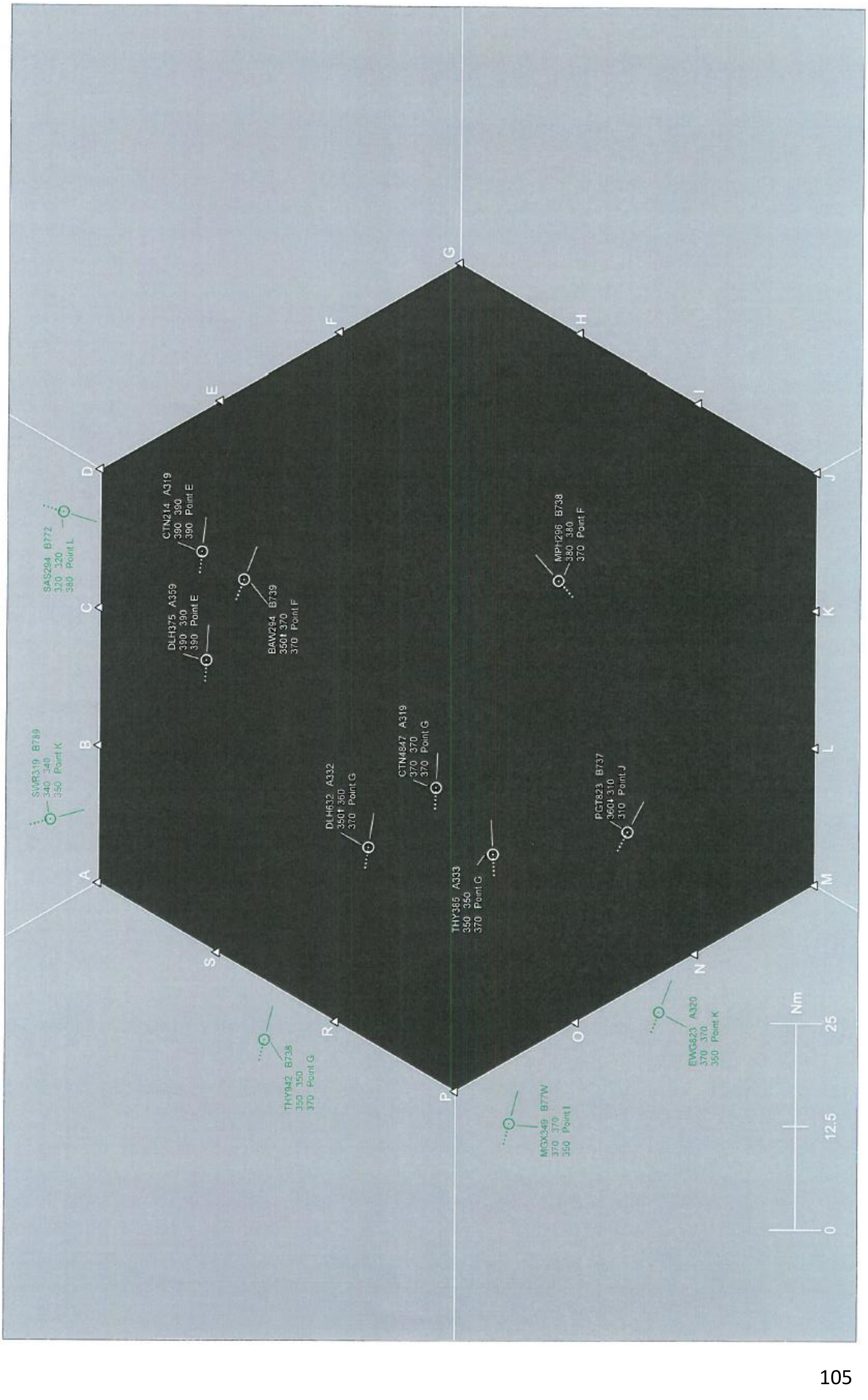
Traffic situation A30

783 G7 681 A30 318 H97 824



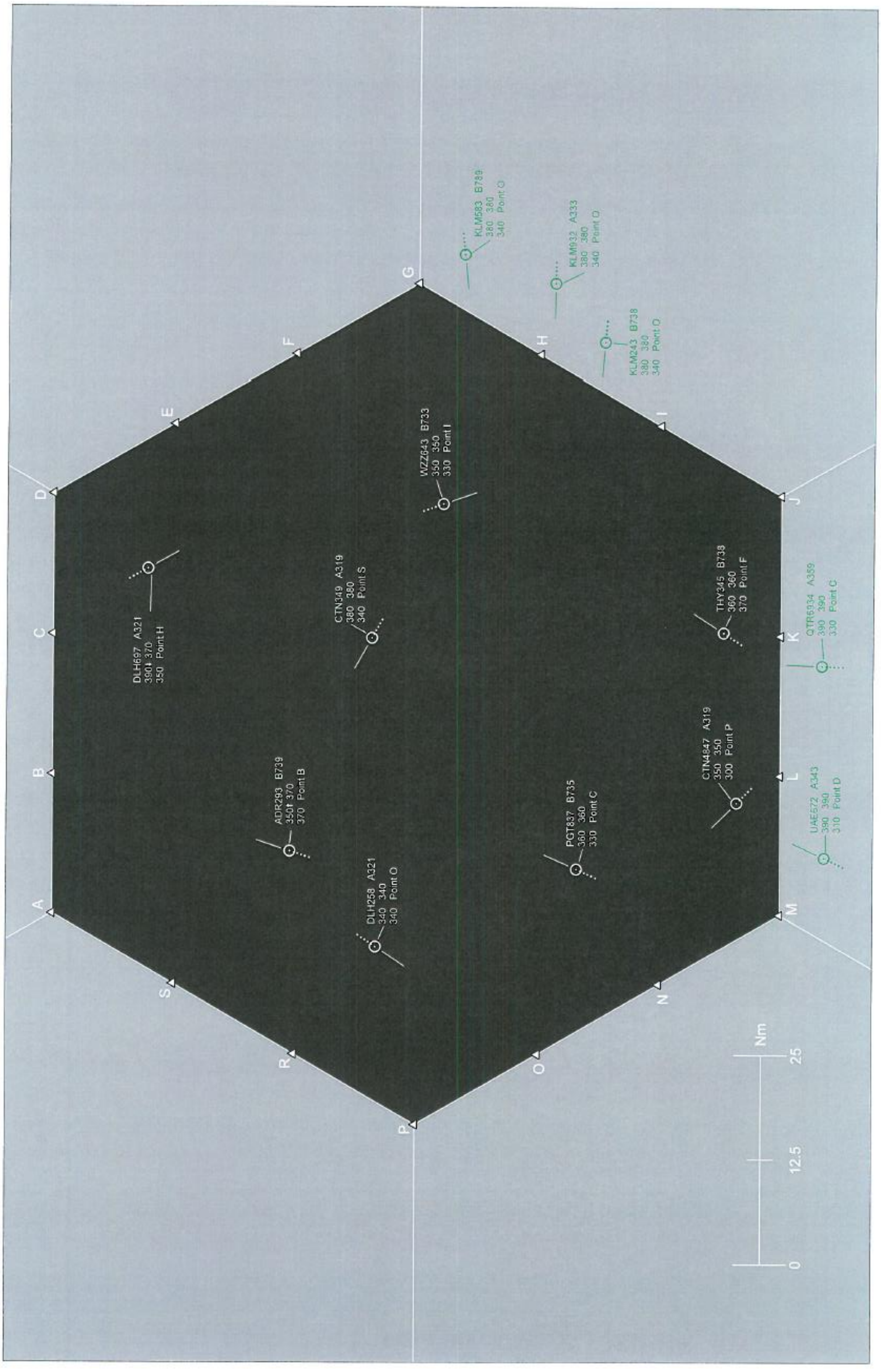
195 U3 145 A31 974 P14 249

Traffic situation A31



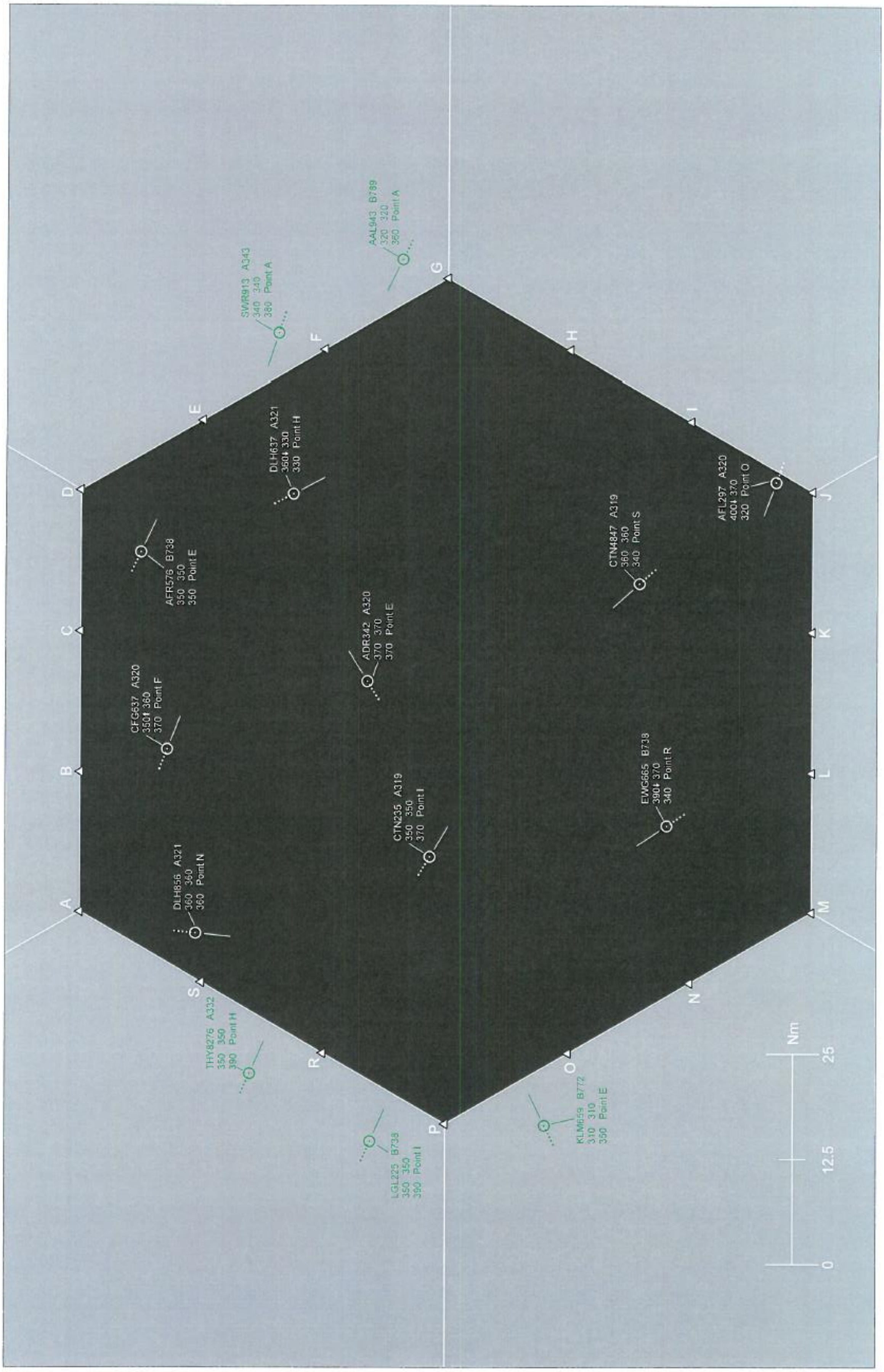
Traffic situation A32

6874 J8 349 A32 971 N74 647



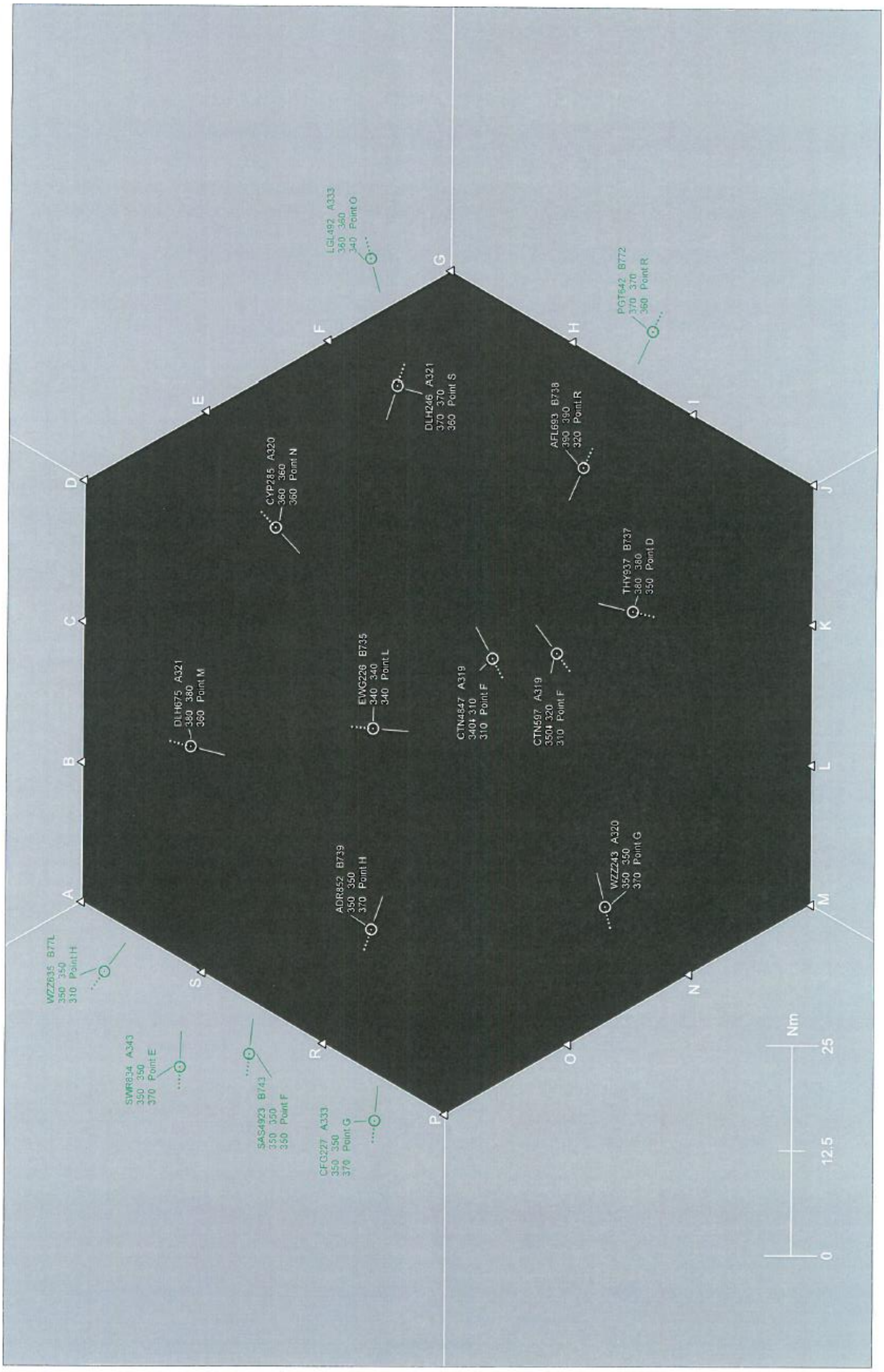
Traffic situation A33

976 E3 642 A33 313 V64 213

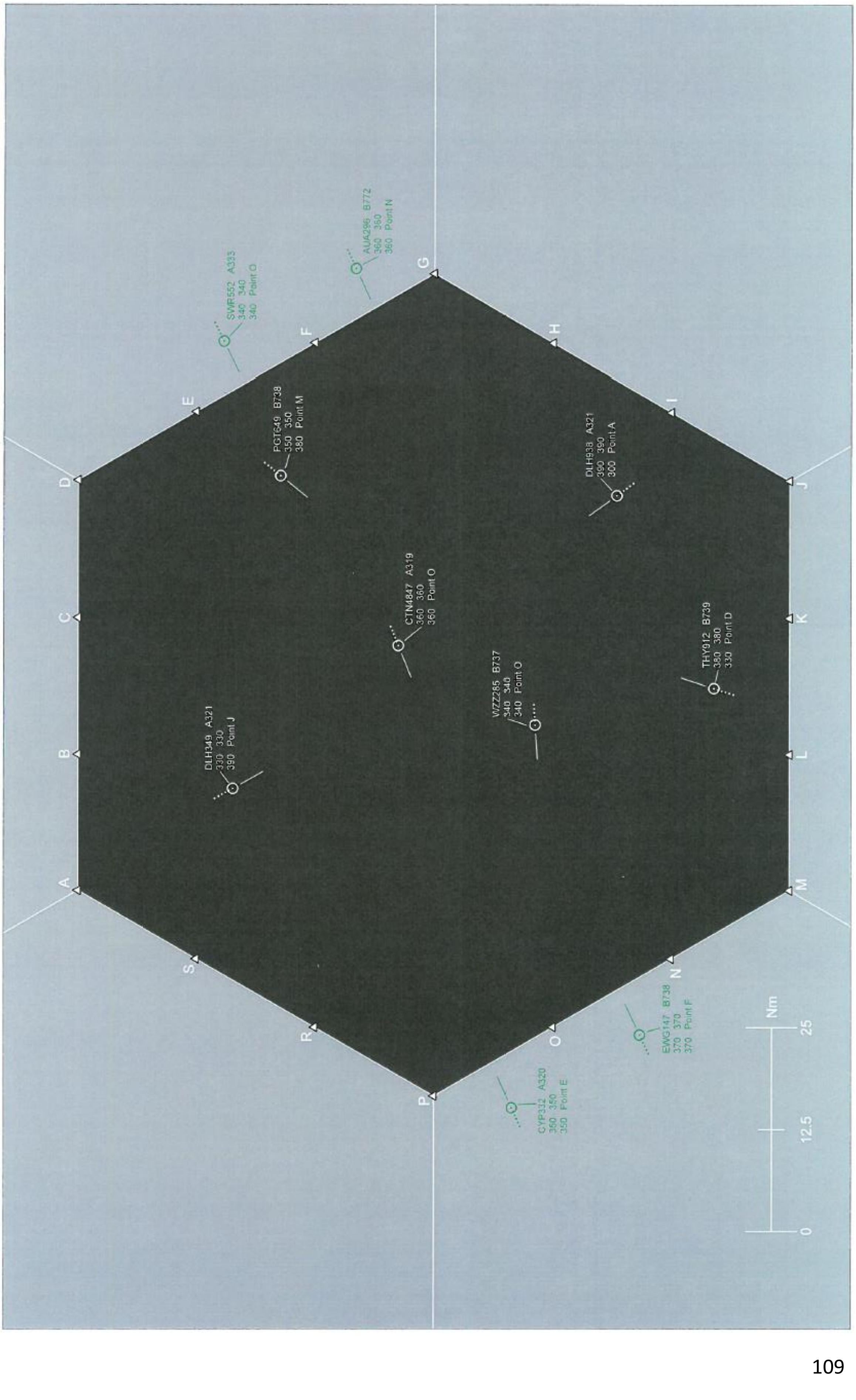


2291 X7 314 A34 294 K42 316

Traffic situation A34

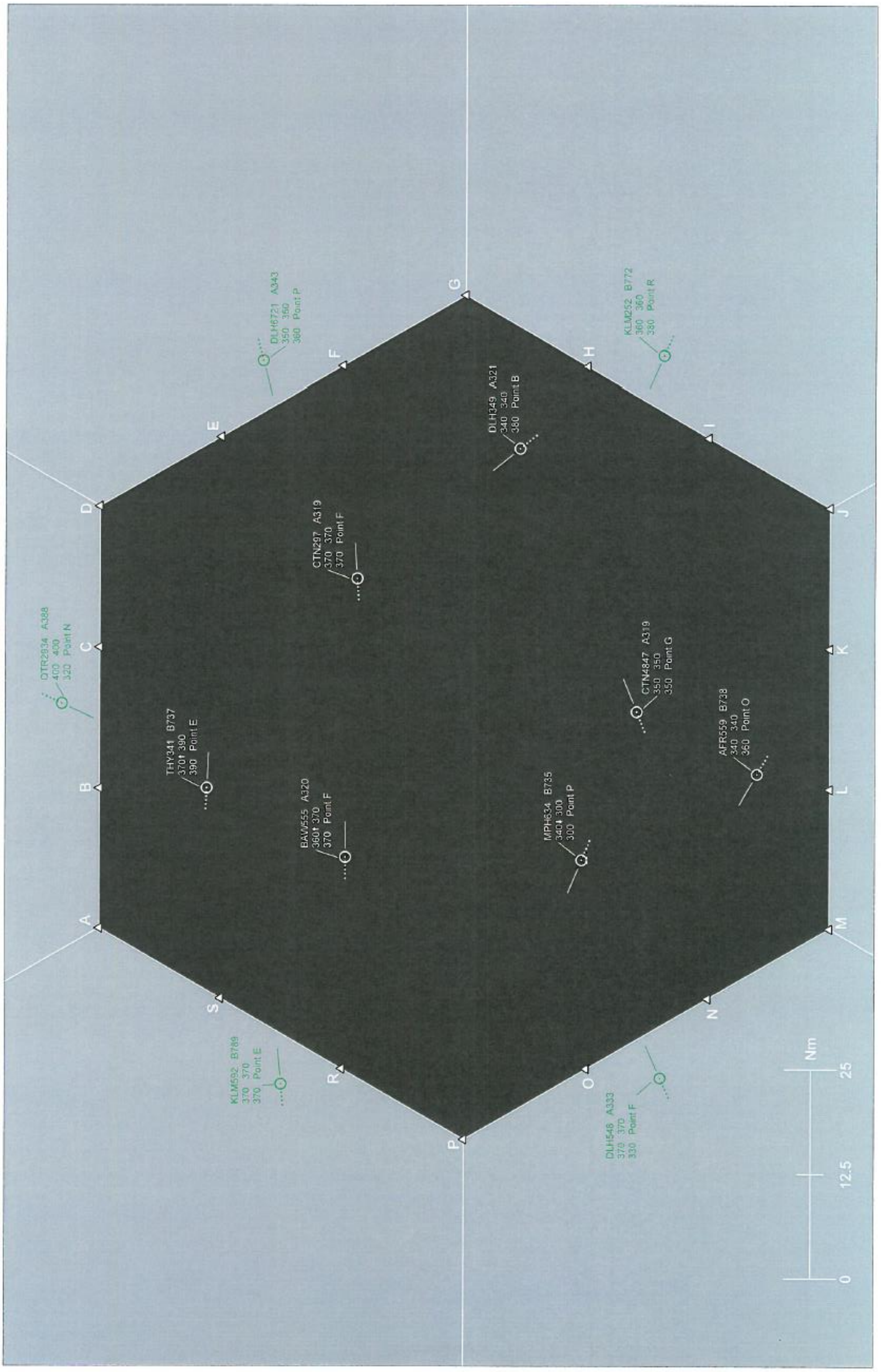


Traffic situation A35



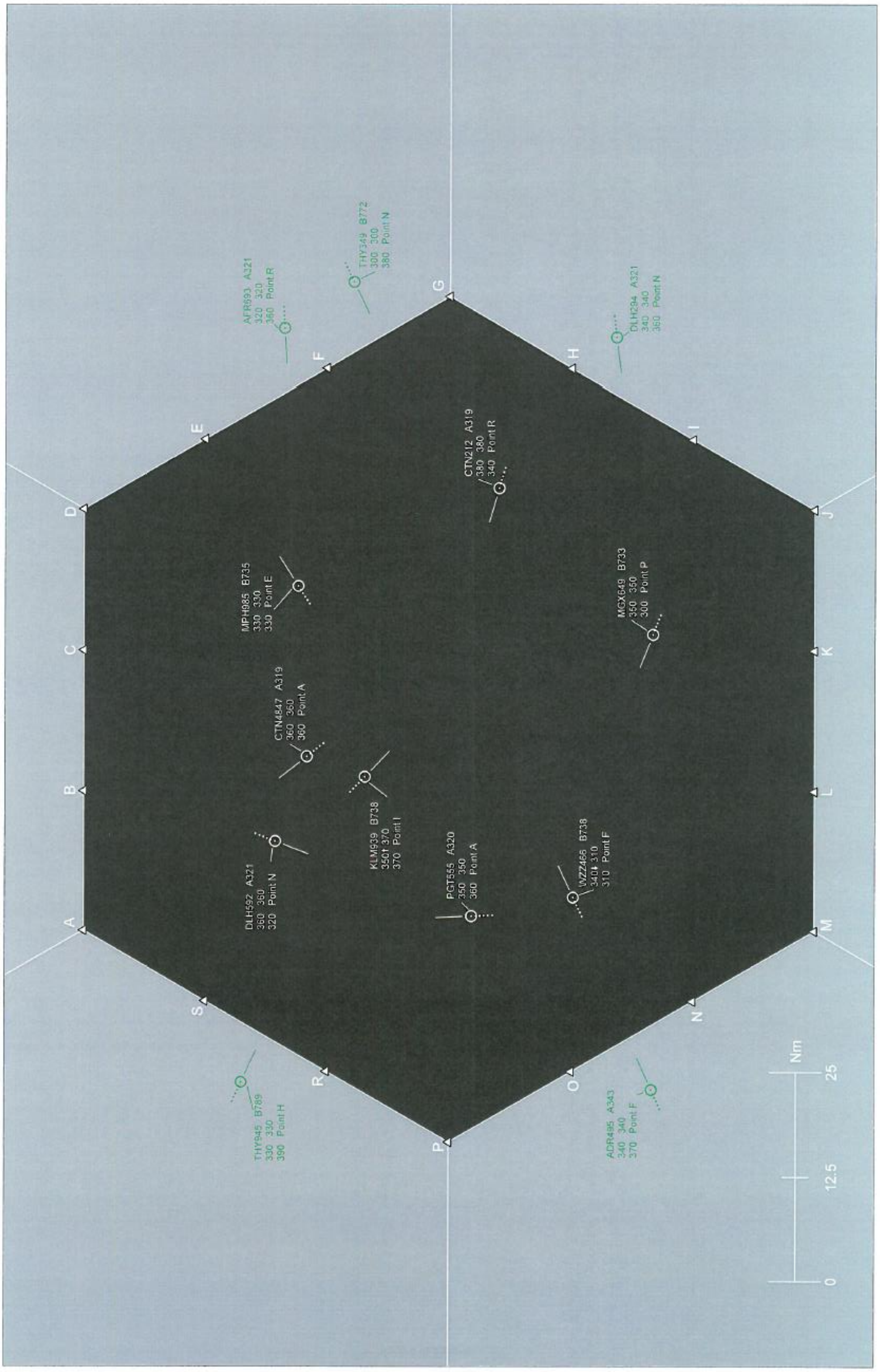
781 G7 681 A36 318 H97 824

Traffic situation A36



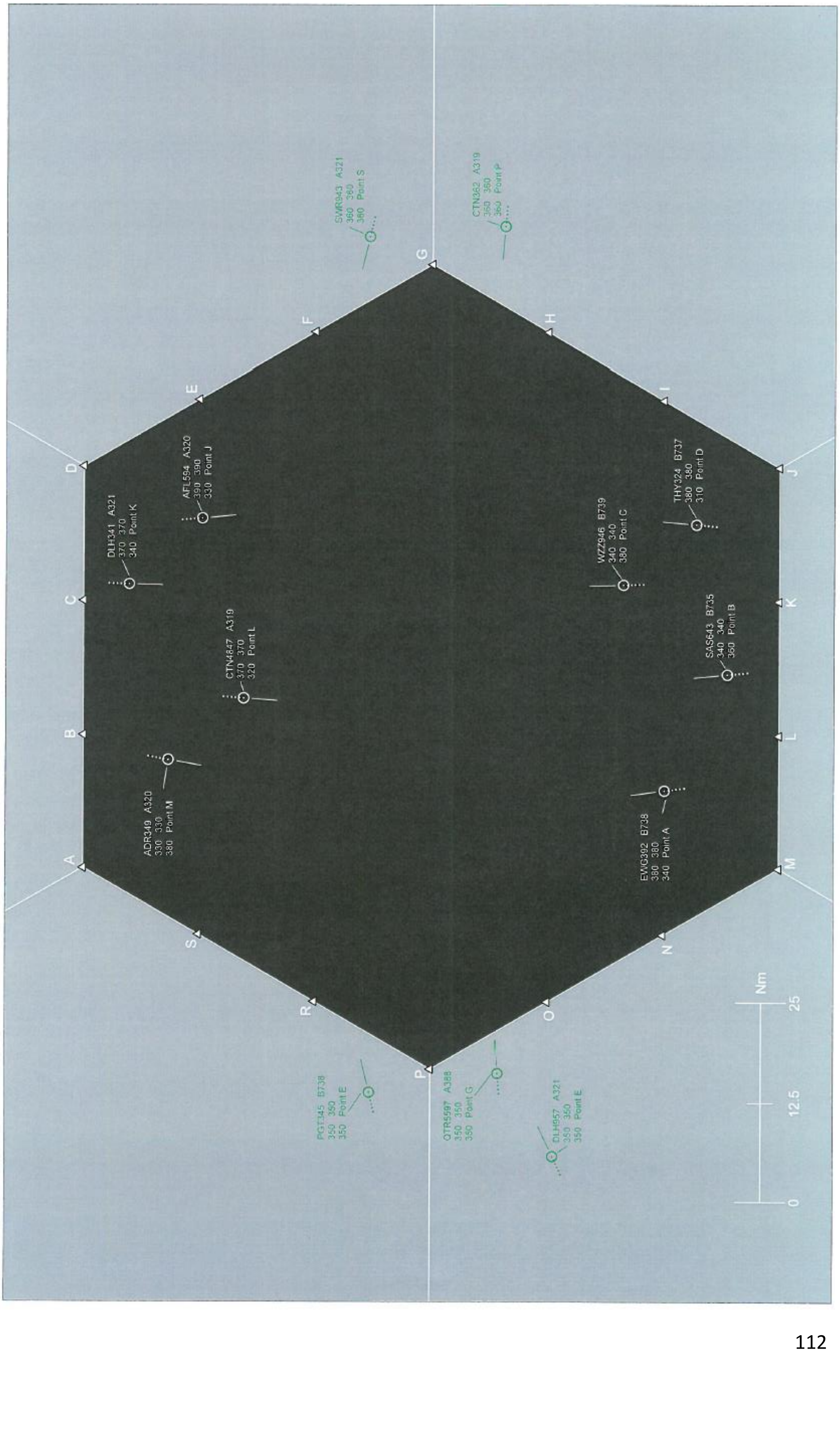
Traffic situation A37

975 U3 145 A37 974 P14 249



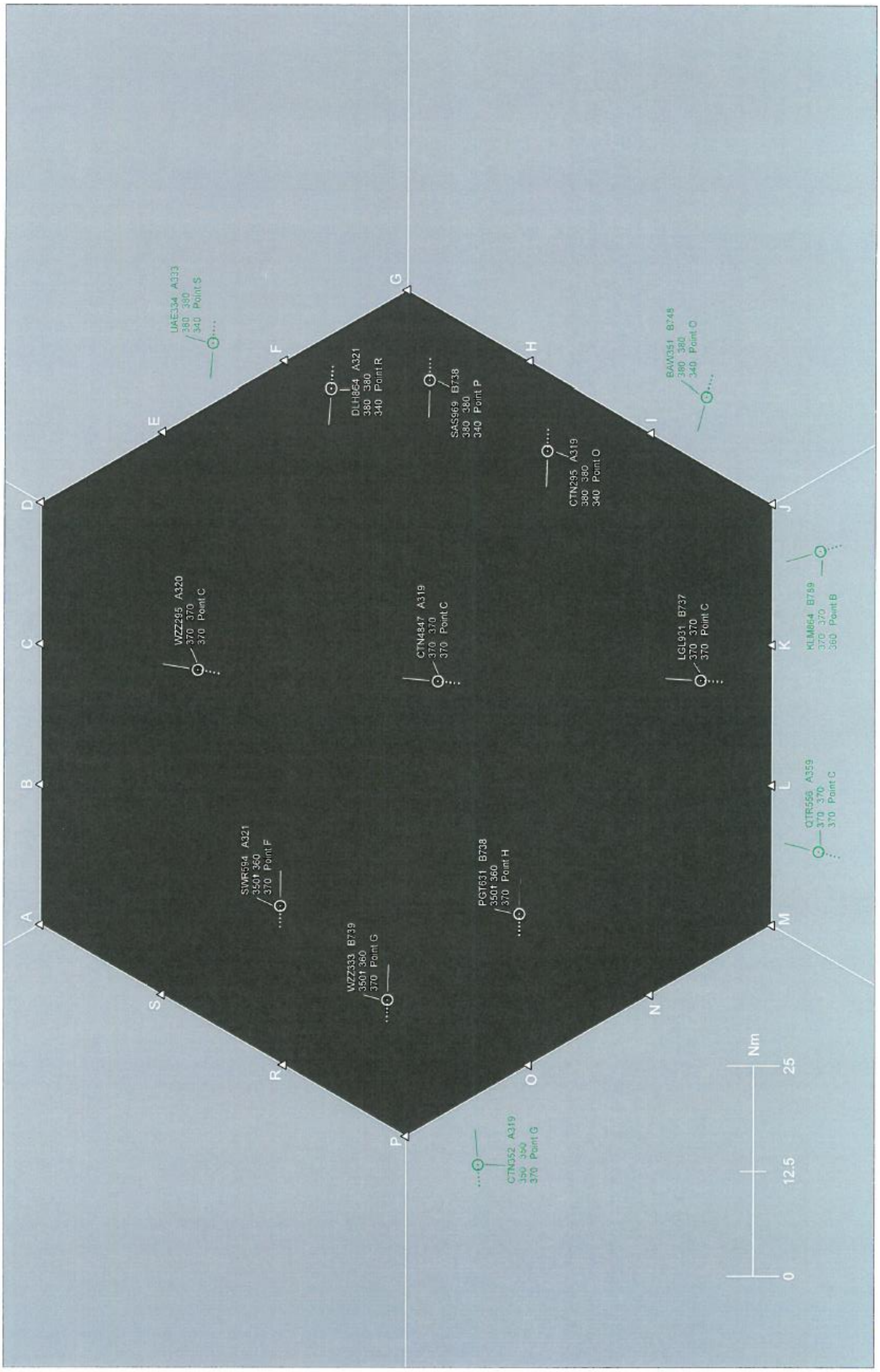
6874 J8 349 A38 971 N74 647

Traffic situation A38



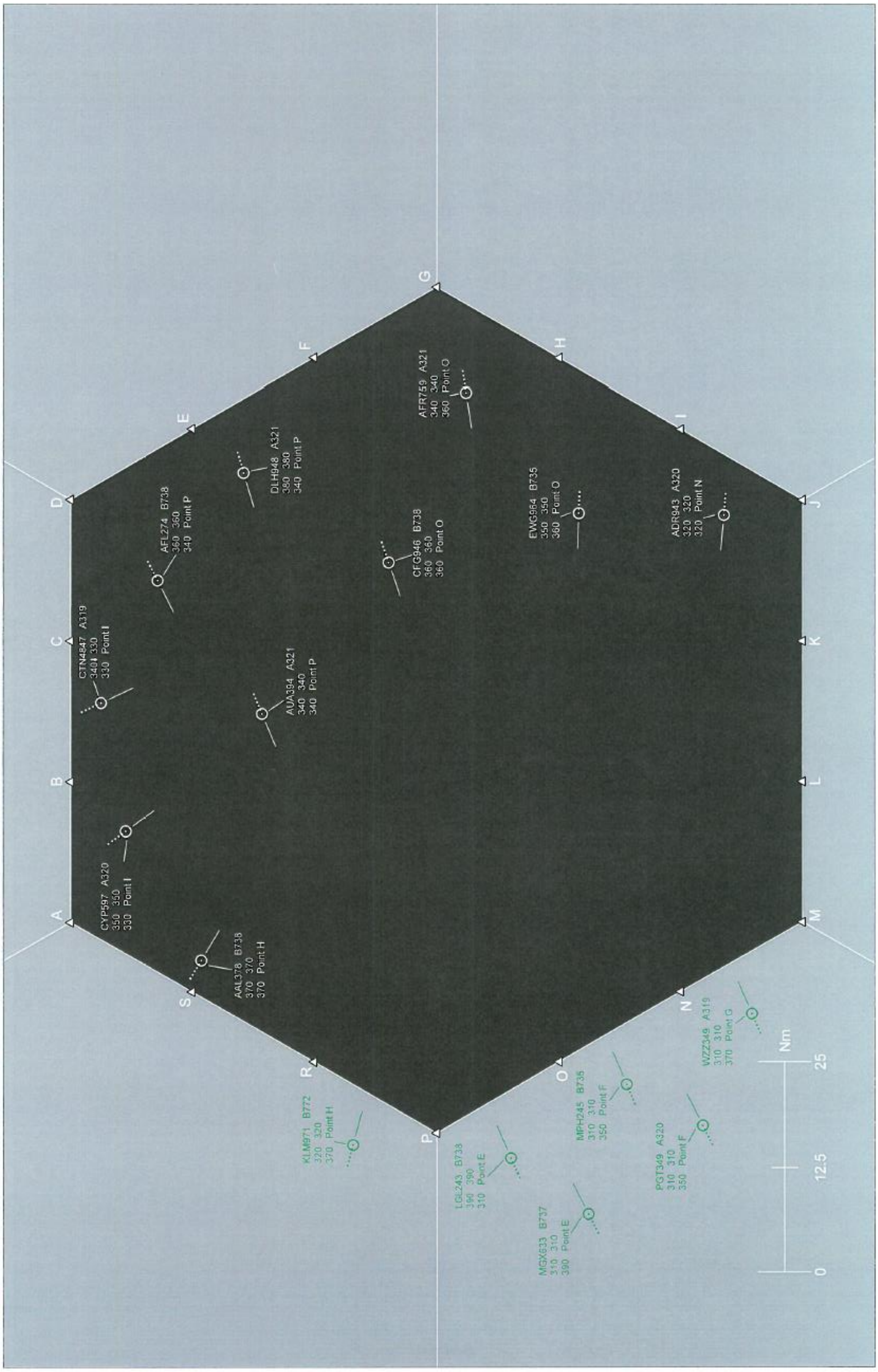
Traffic situation A39

976 E3 642 A39 313 V64 213



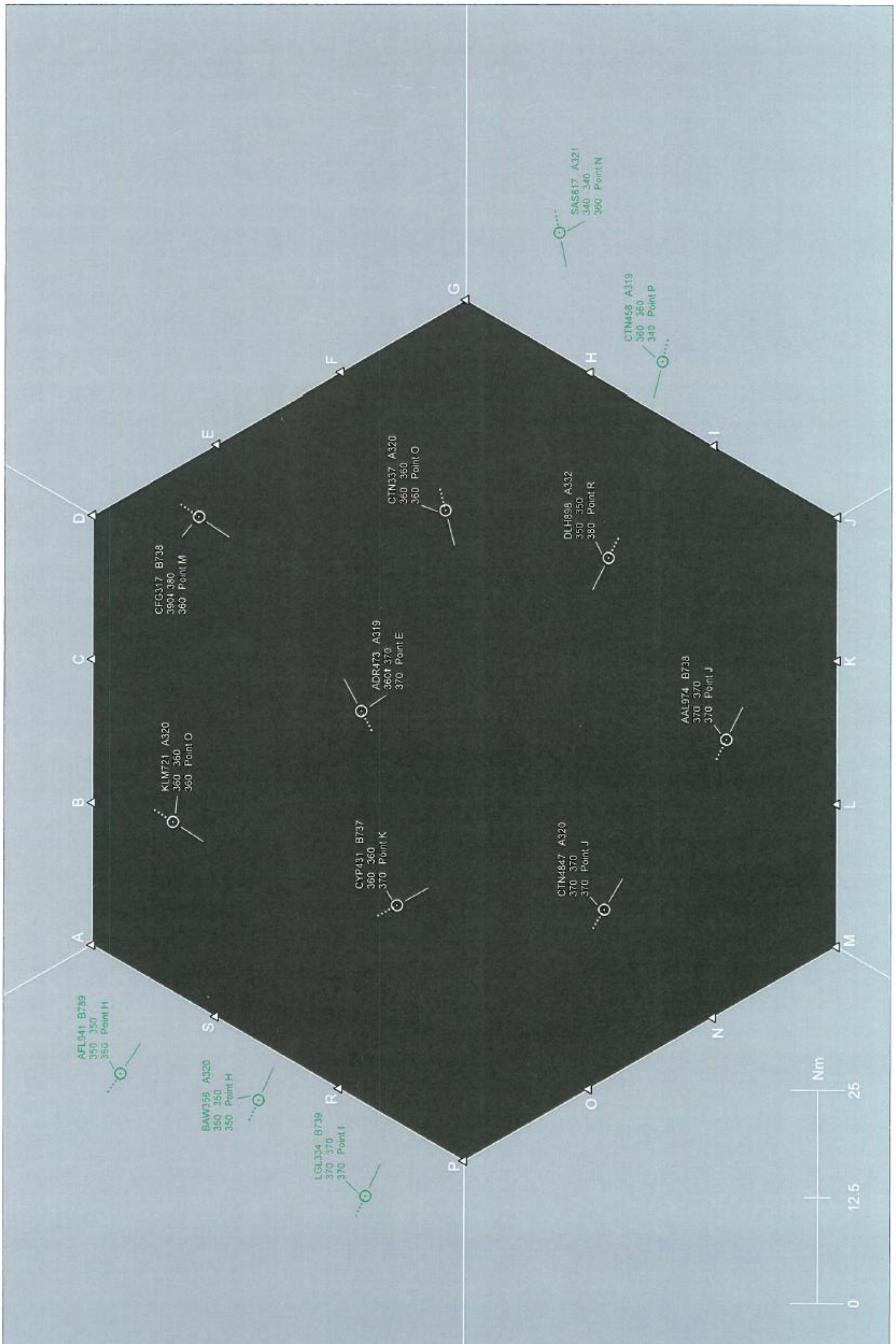
2294 X7 364 A40 294 K42 316

Traffic situation A40



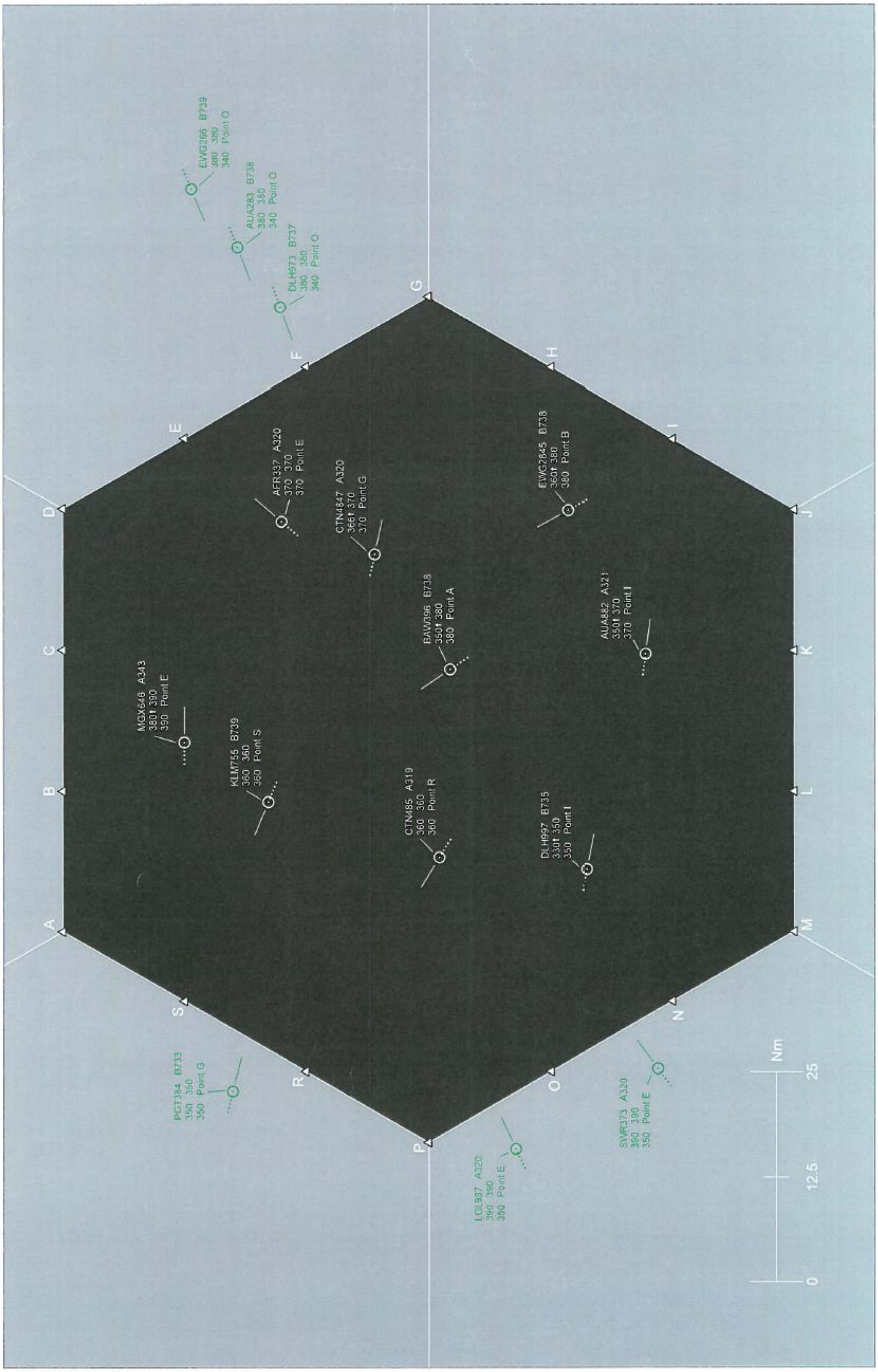
6781 V8 656 B1 373 P26 364

Traffic situation B1

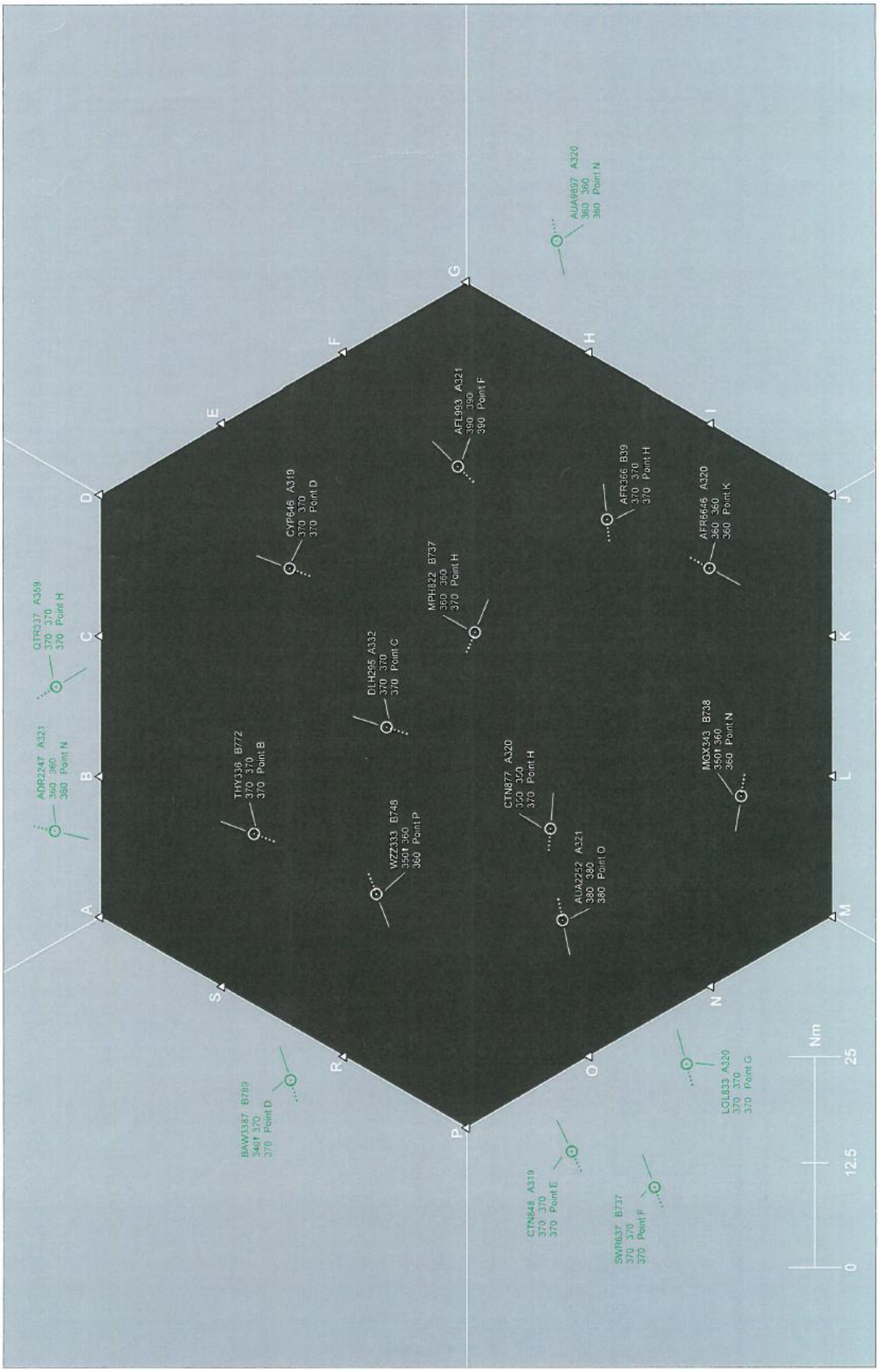


5423 W2 111 B2 214 I21 648

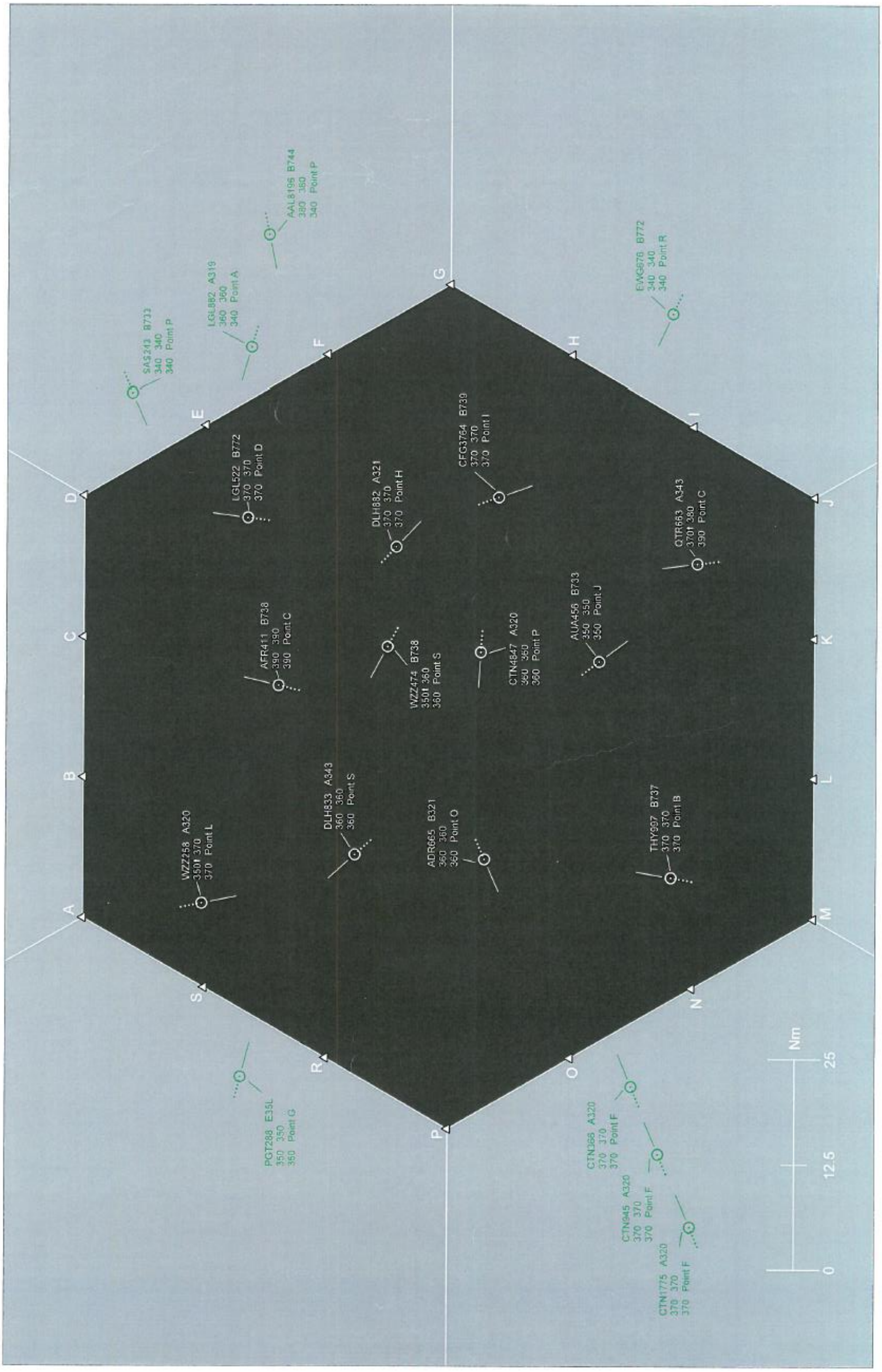
Traffic situation B2



Traffic situation B3

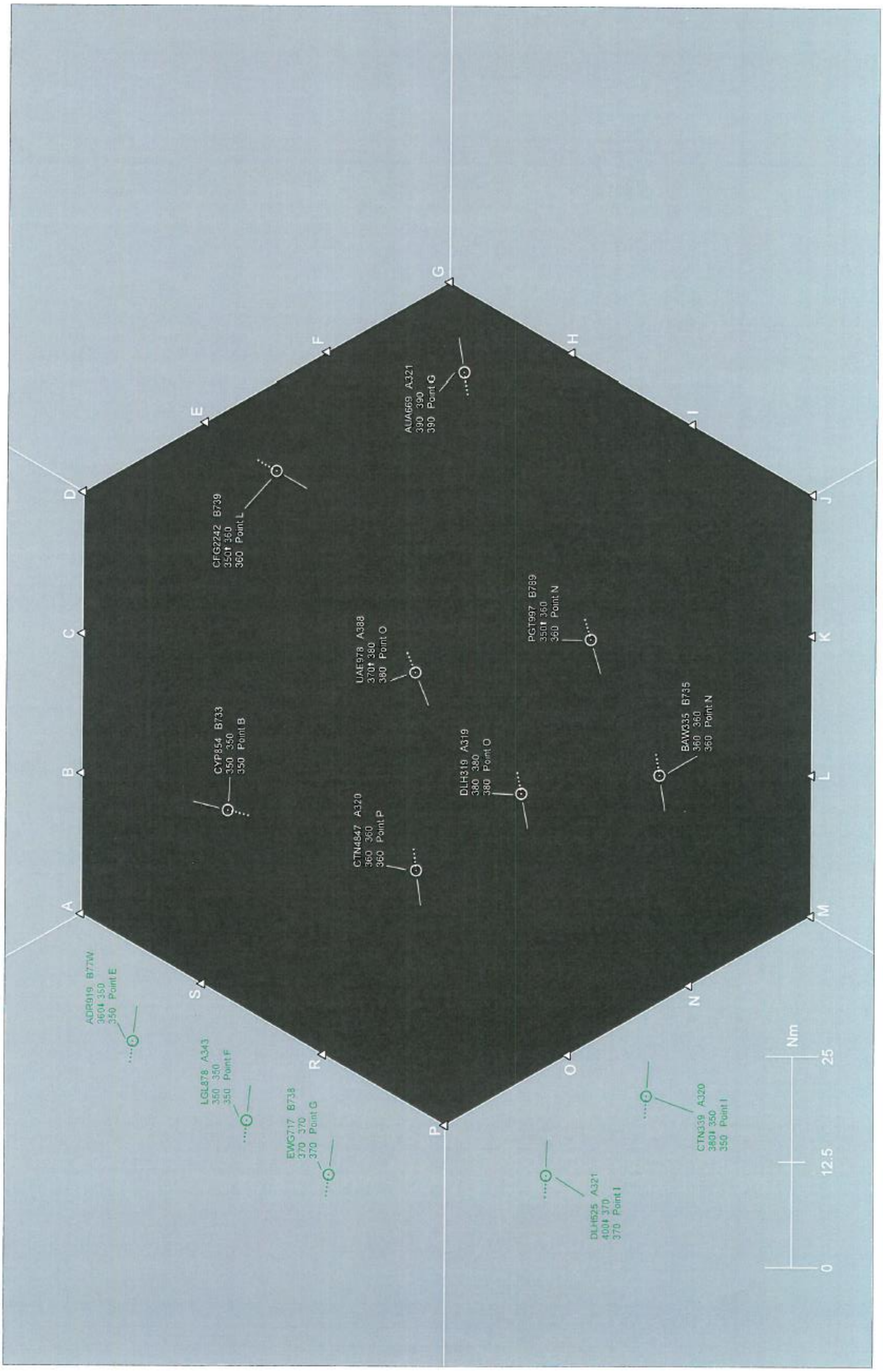


Traffic situation B4

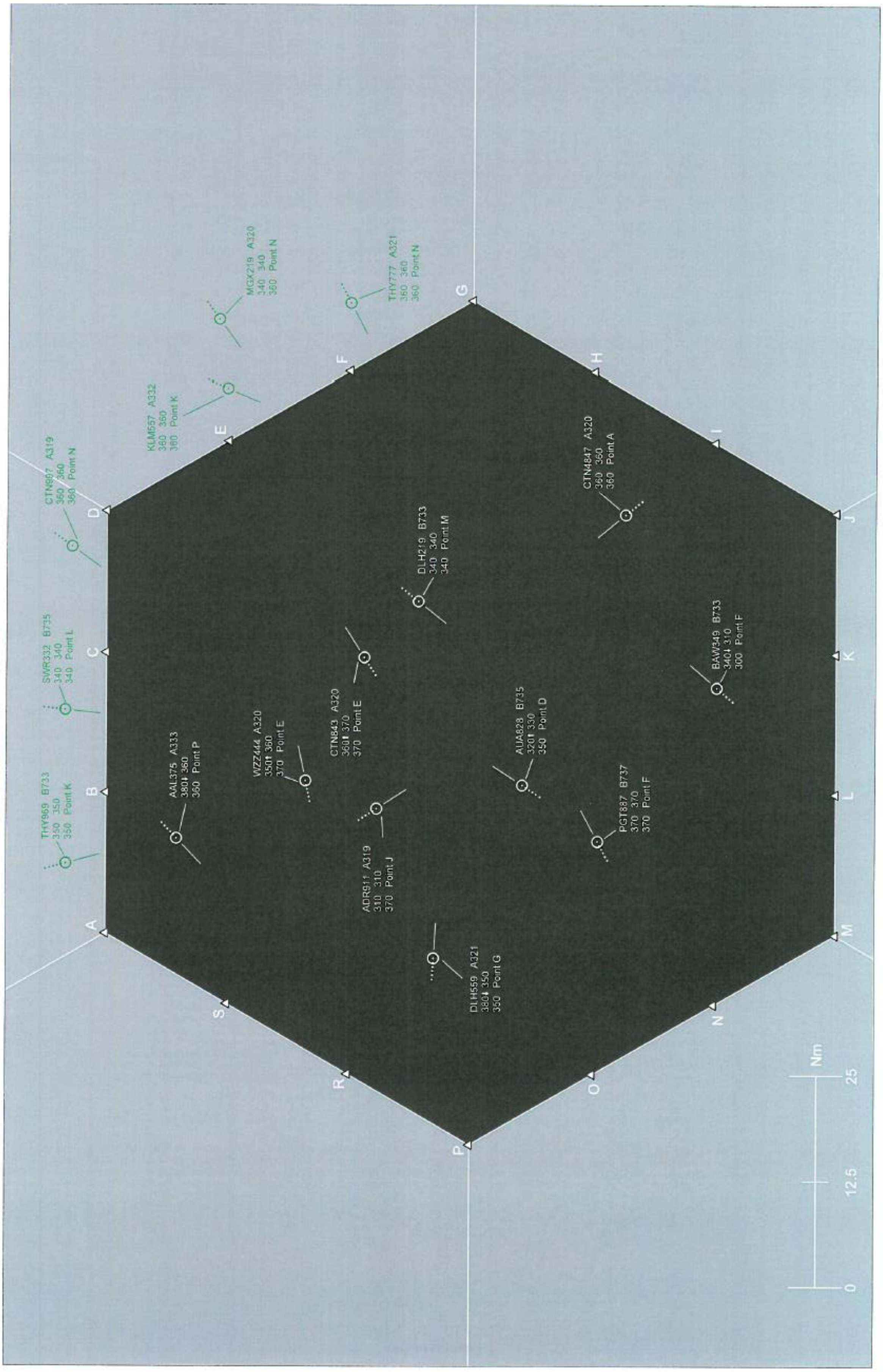


Traffic situation B5

6739 G3 696 B5 191 F78 197

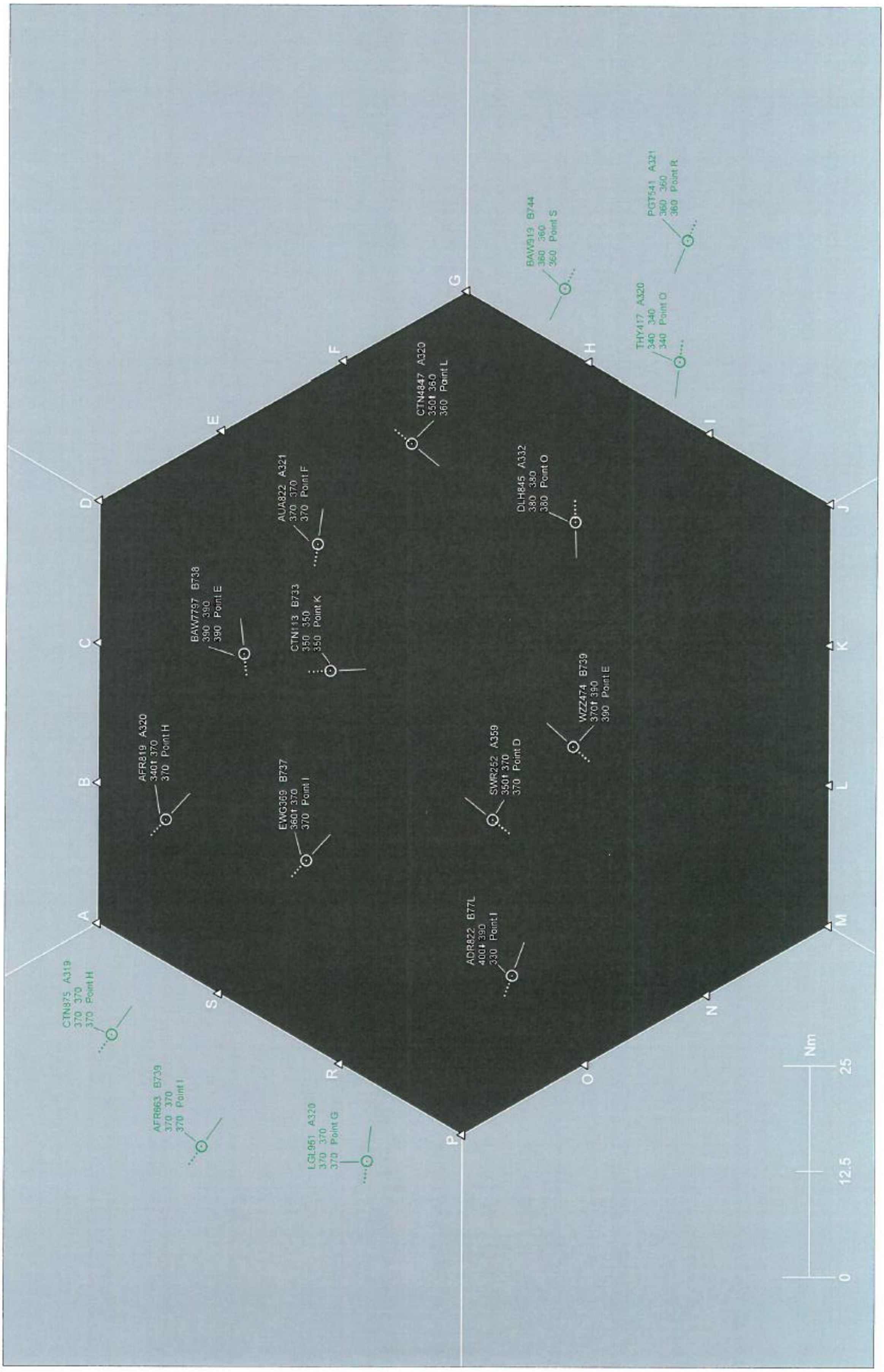


Traffic situation B6

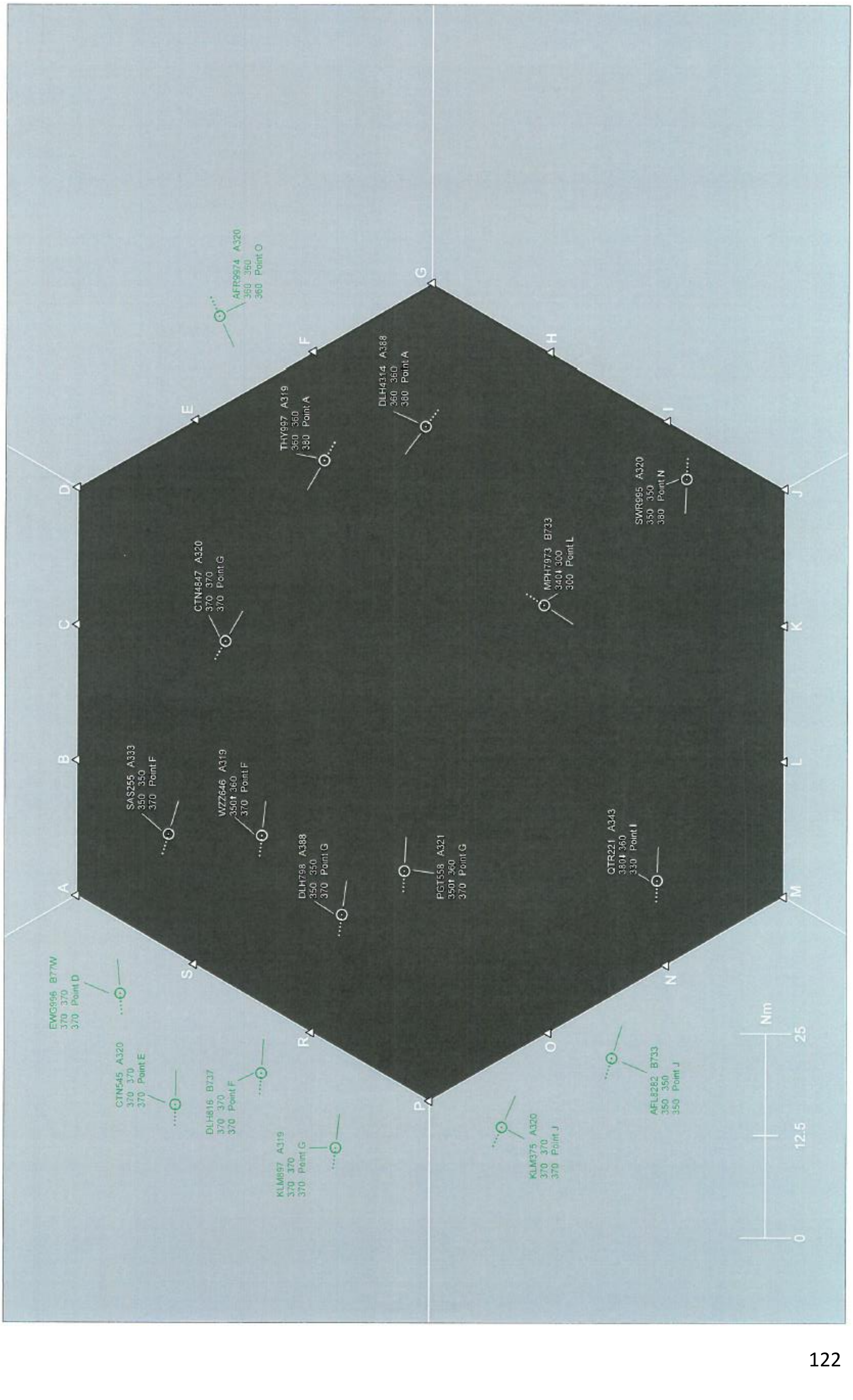


Traffic situation B7

7480 Q3 121 B7 141 K69 193

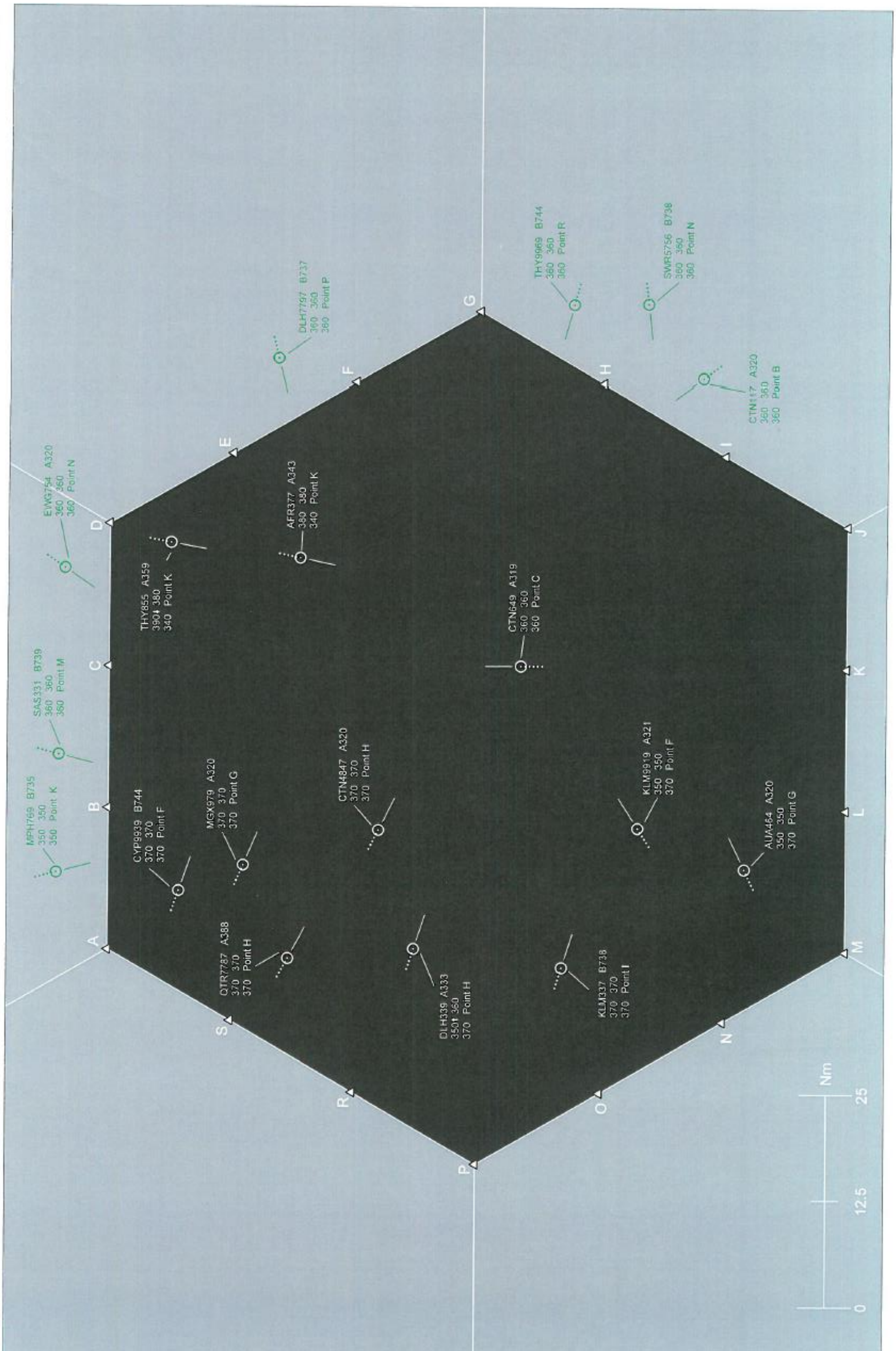


Traffic situation B8



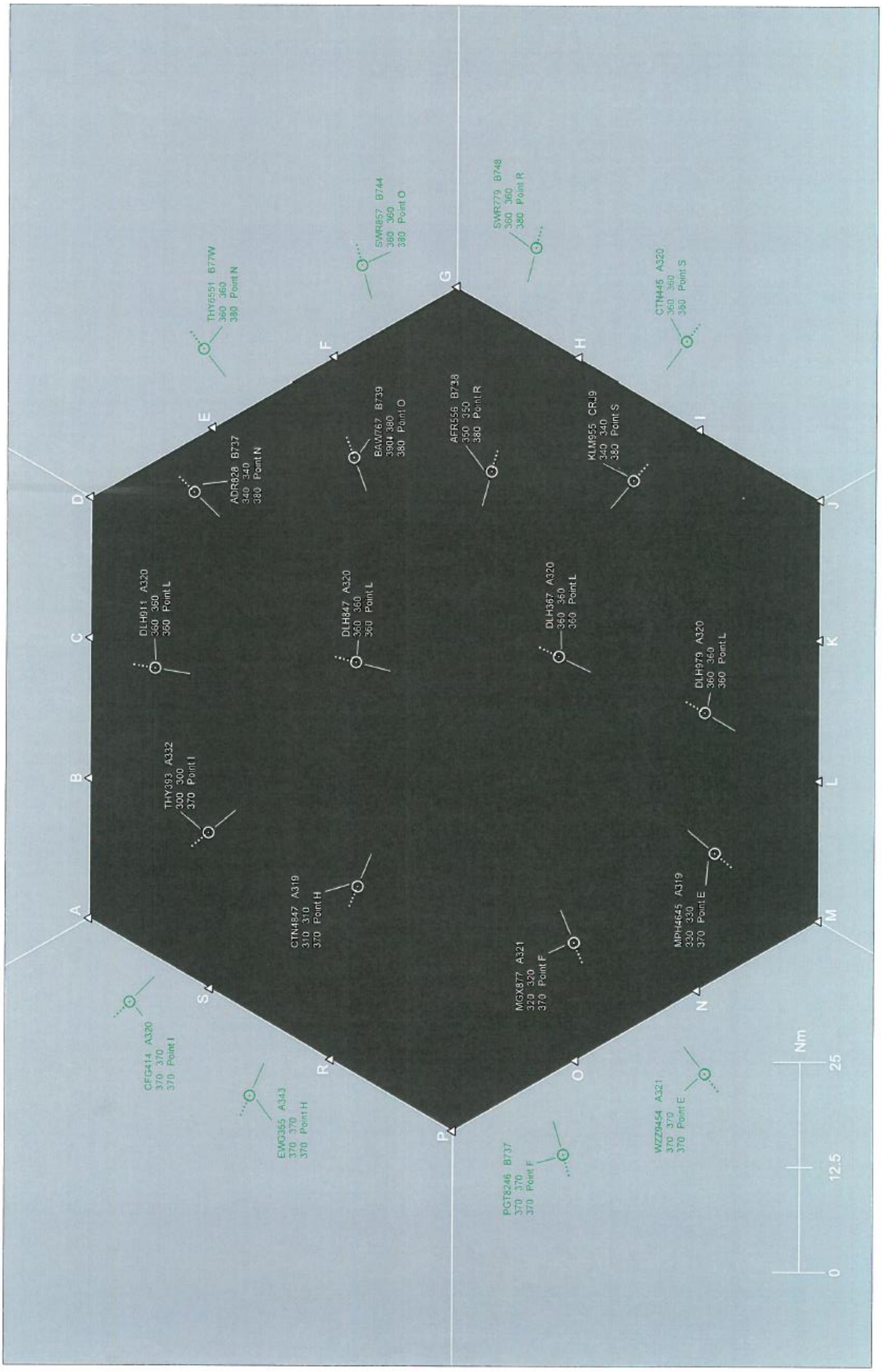
Traffic situation B9

2947 M3 111 B9 194 K33 549

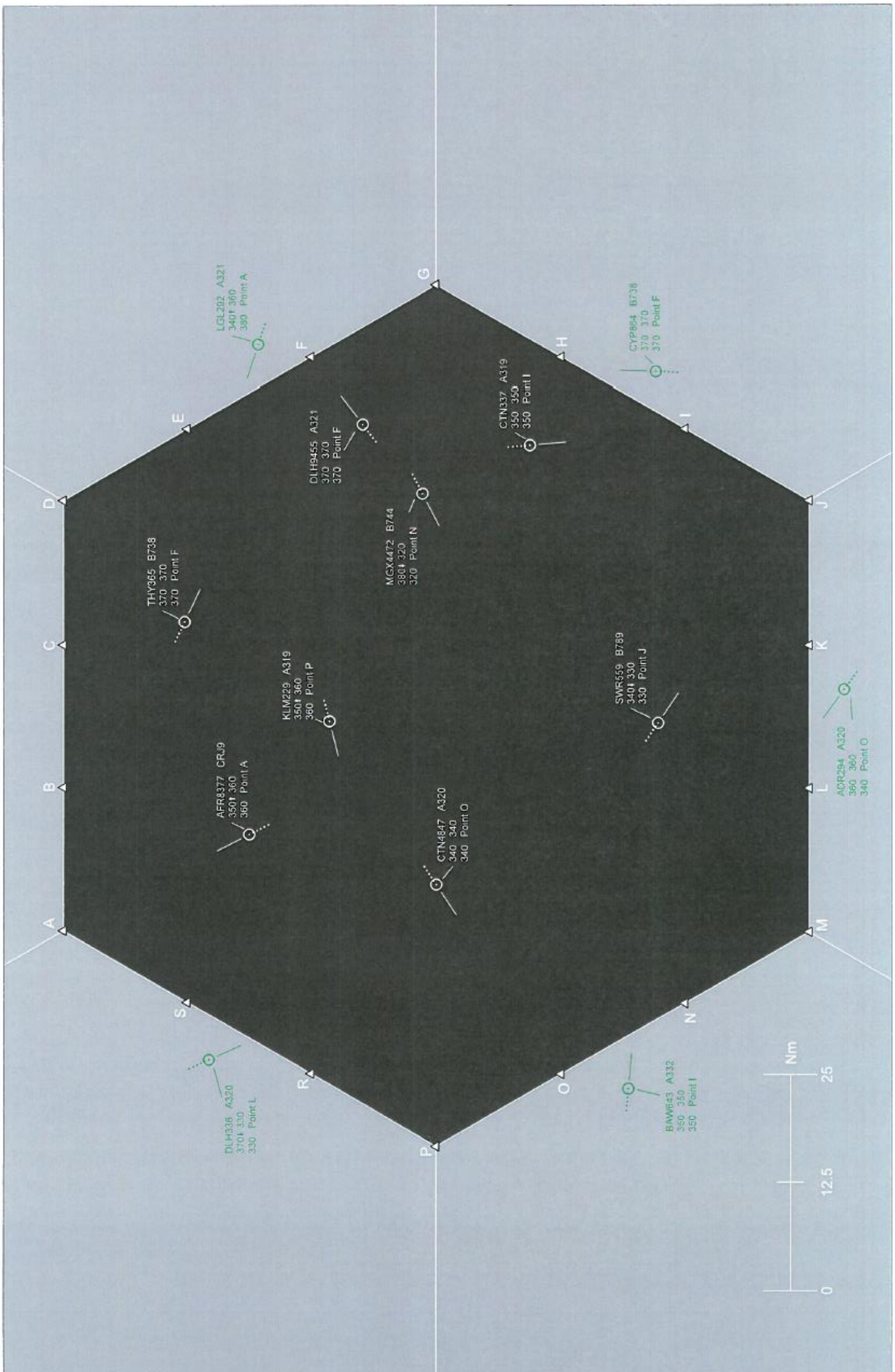


Traffic situation B10

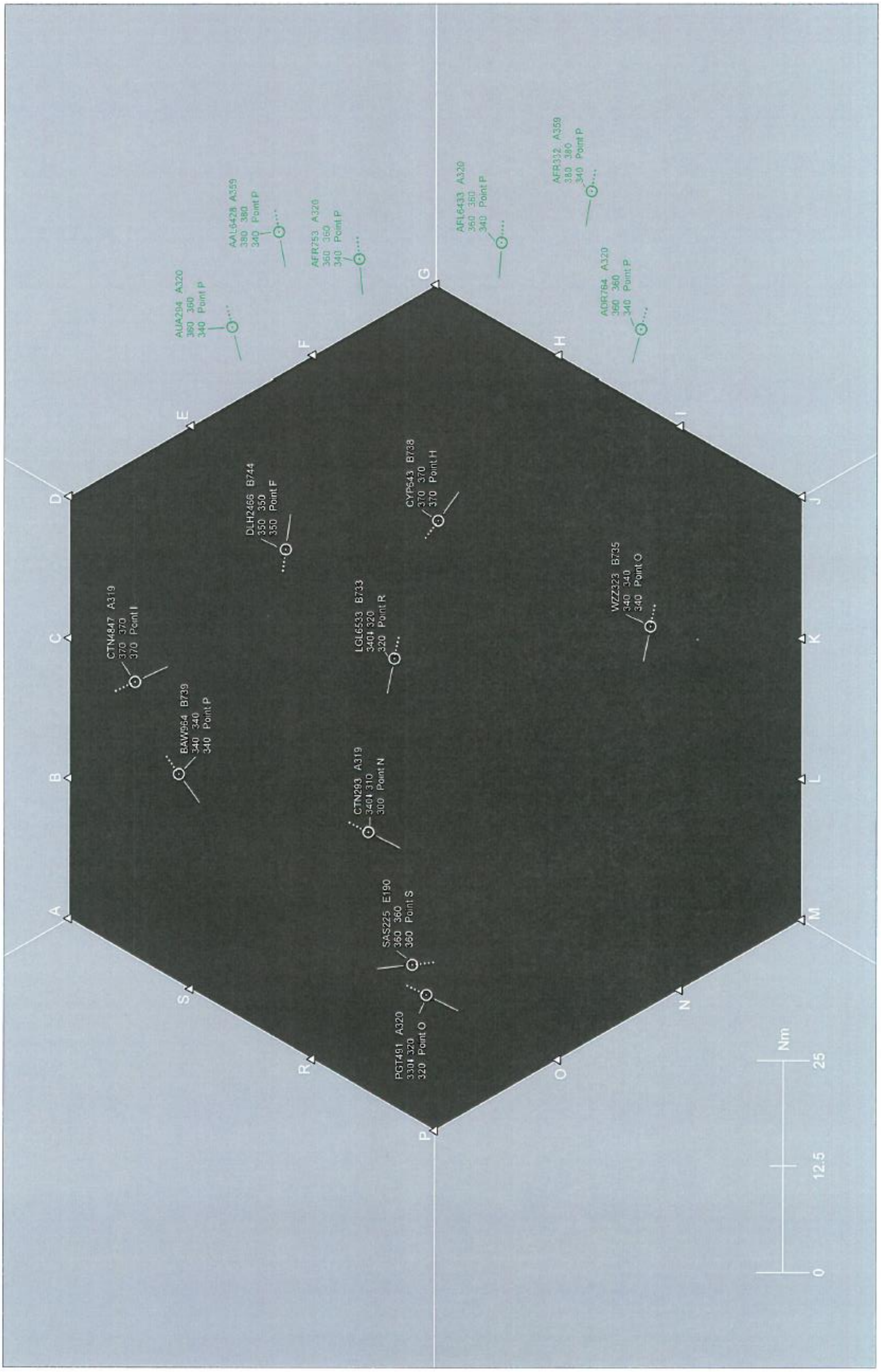
7719 T2 964 B10 137 R6 876



Traffic situation B11

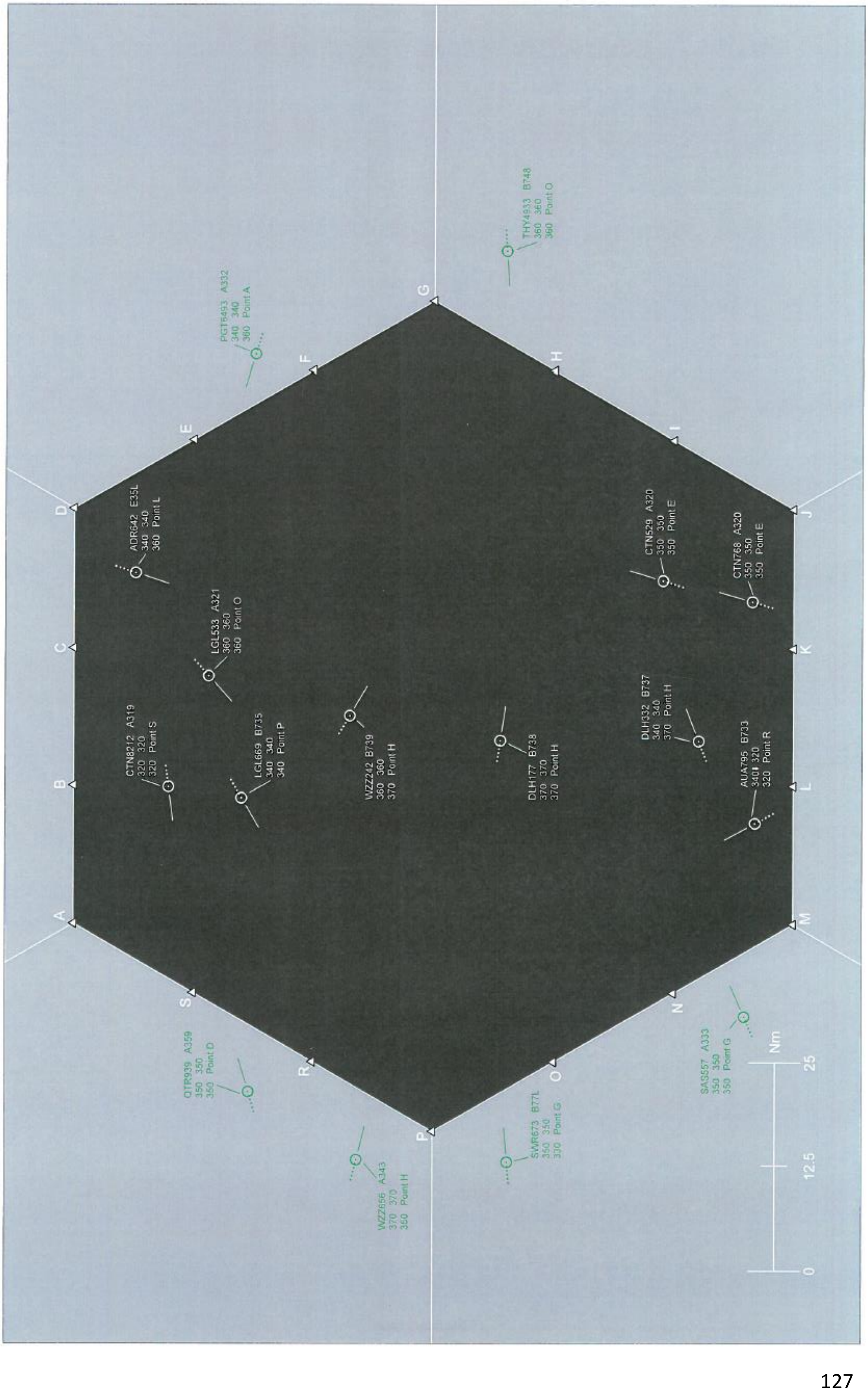


Traffic situation B12

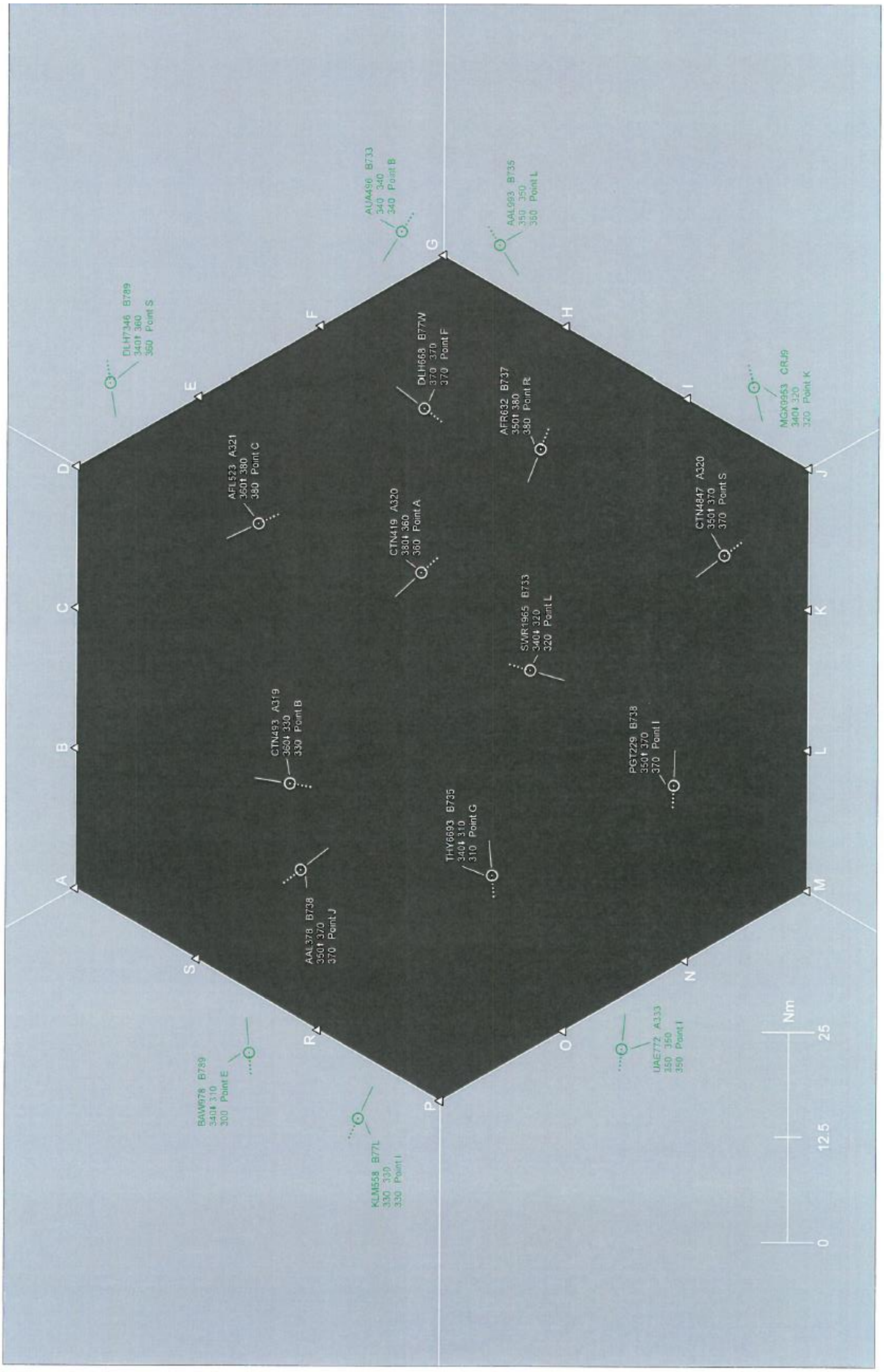


1937 U2 597 B13 697 R8 997

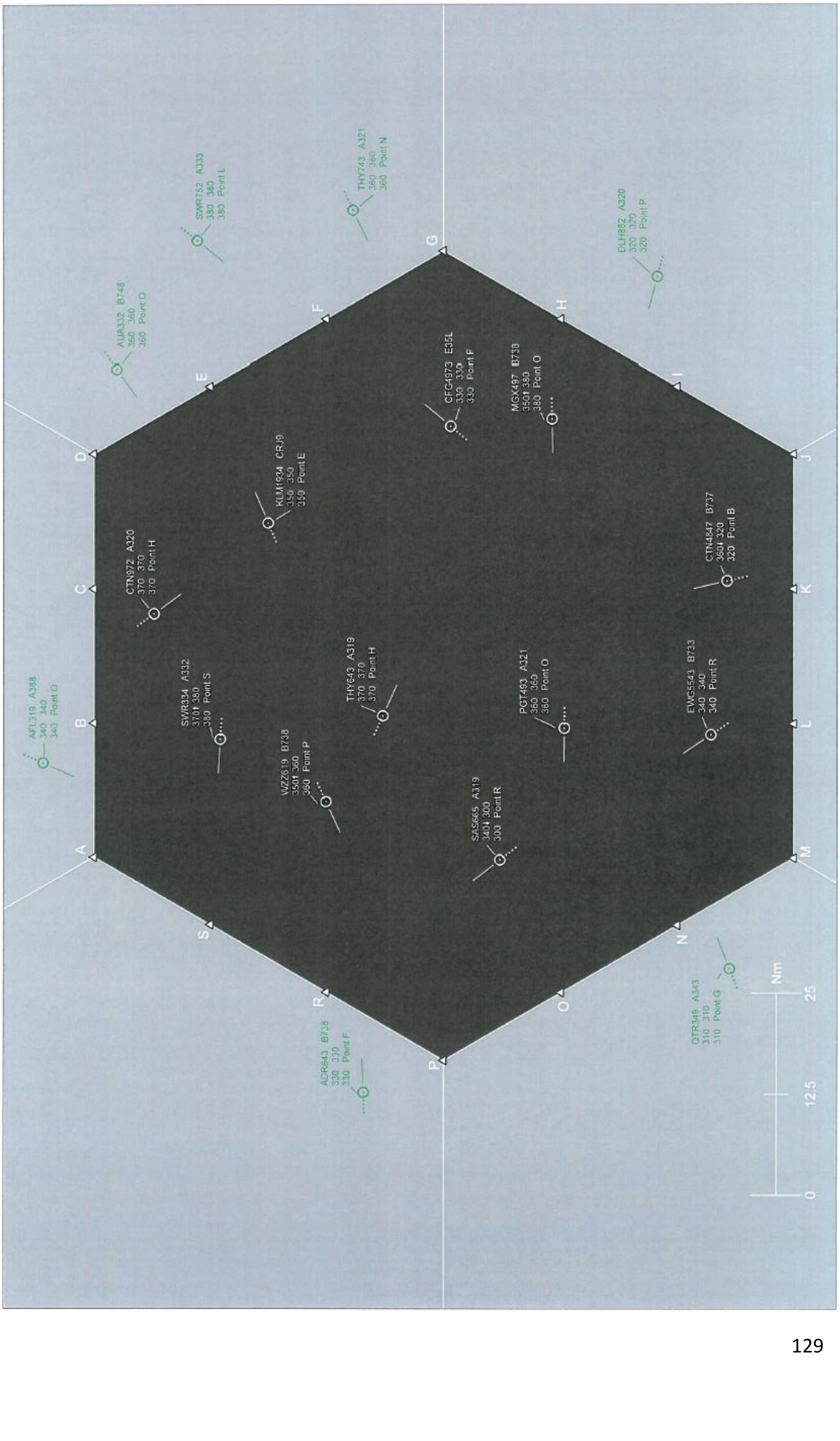
Traffic situation B13



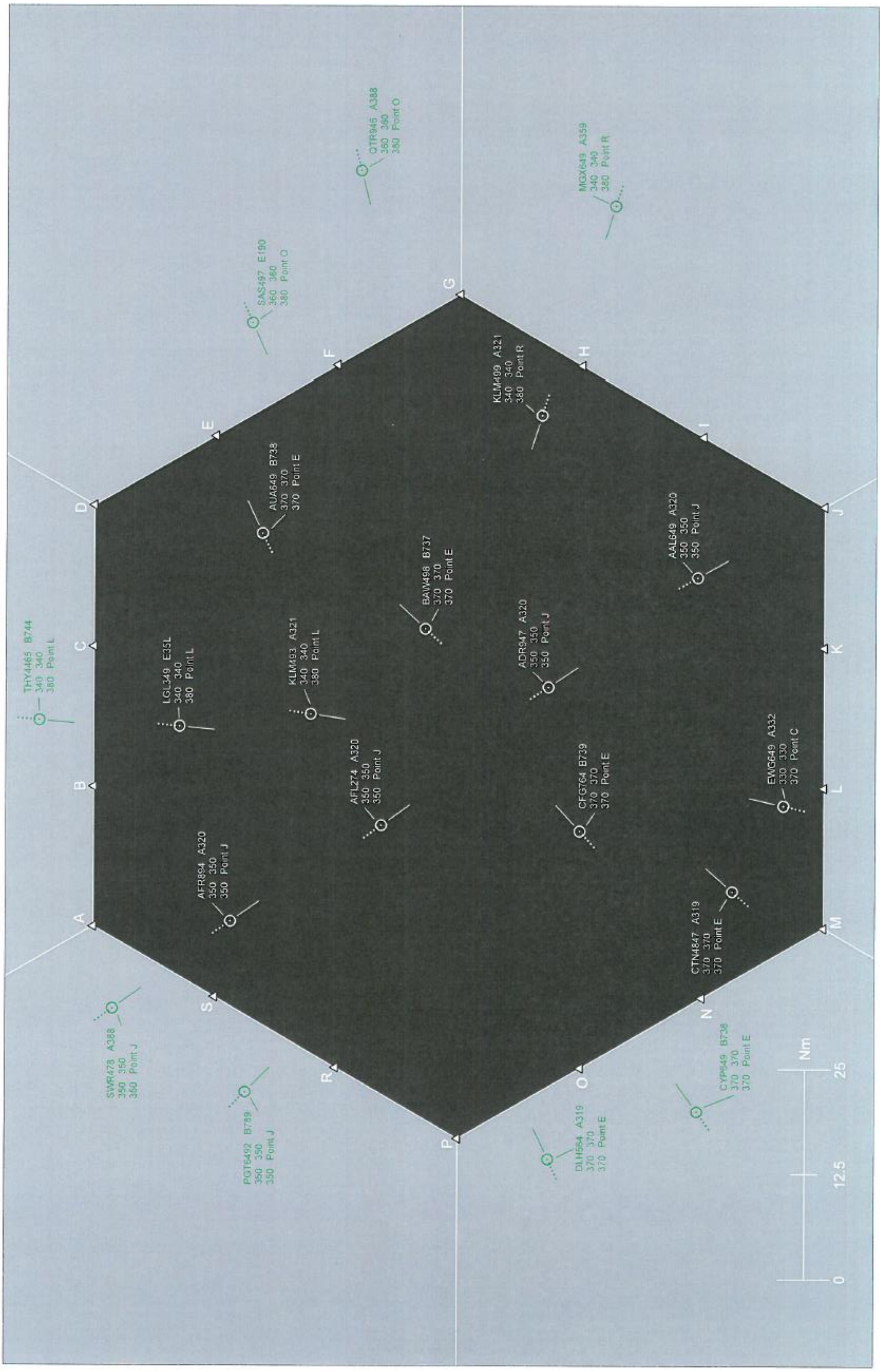
Traffic situation B14



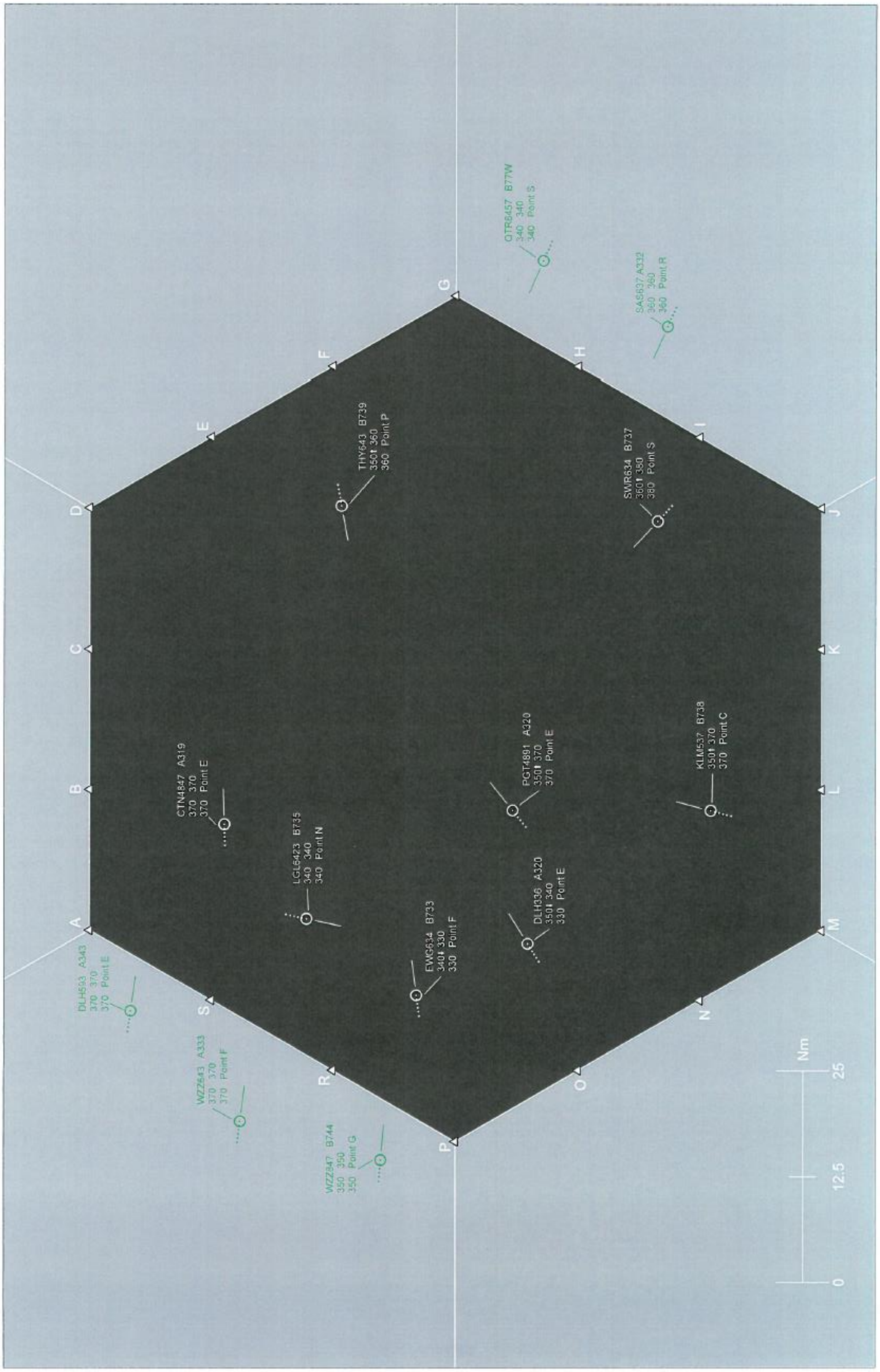
Traffic situation B15



Traffic situation B16

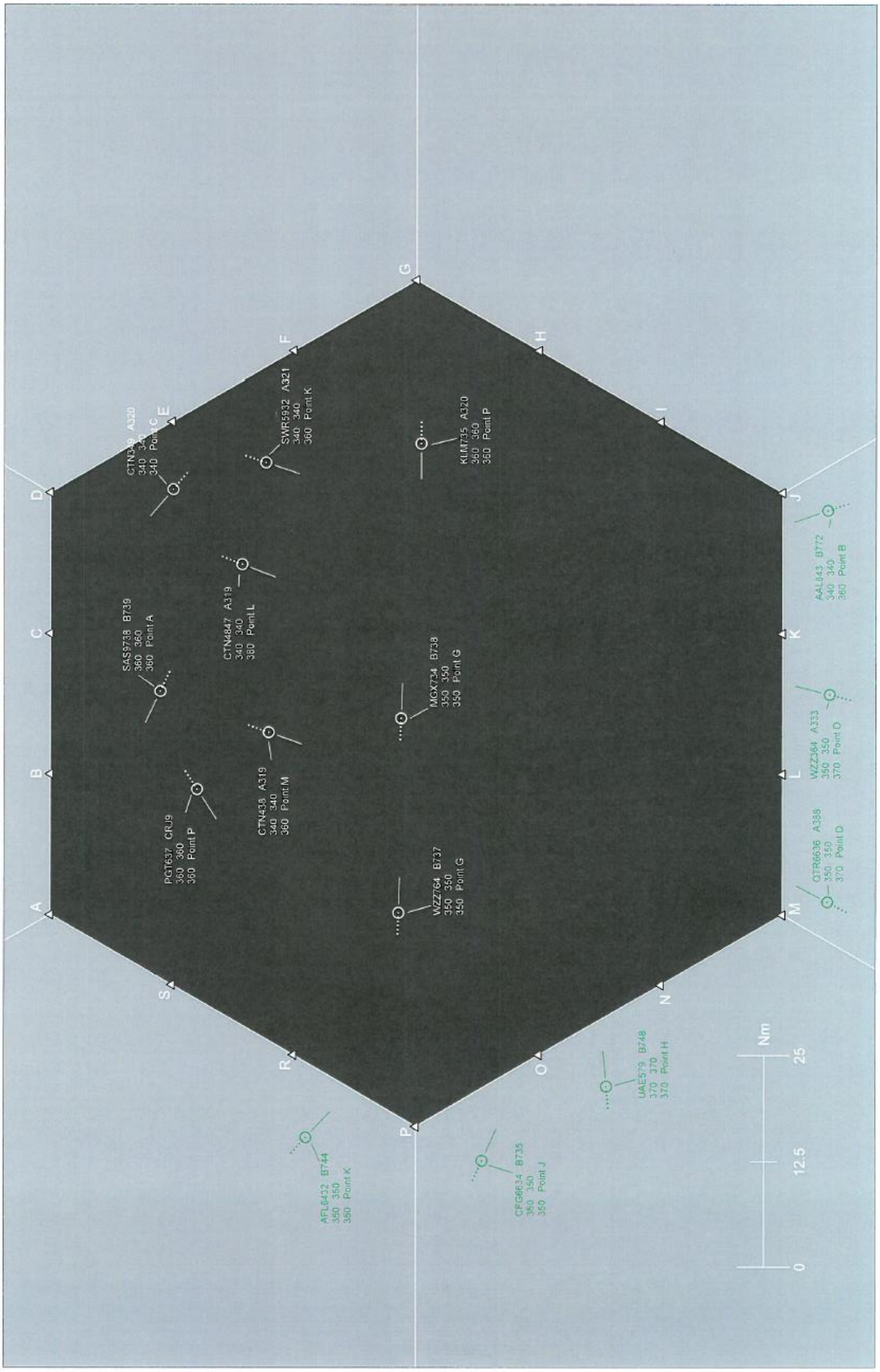


Traffic situation B17



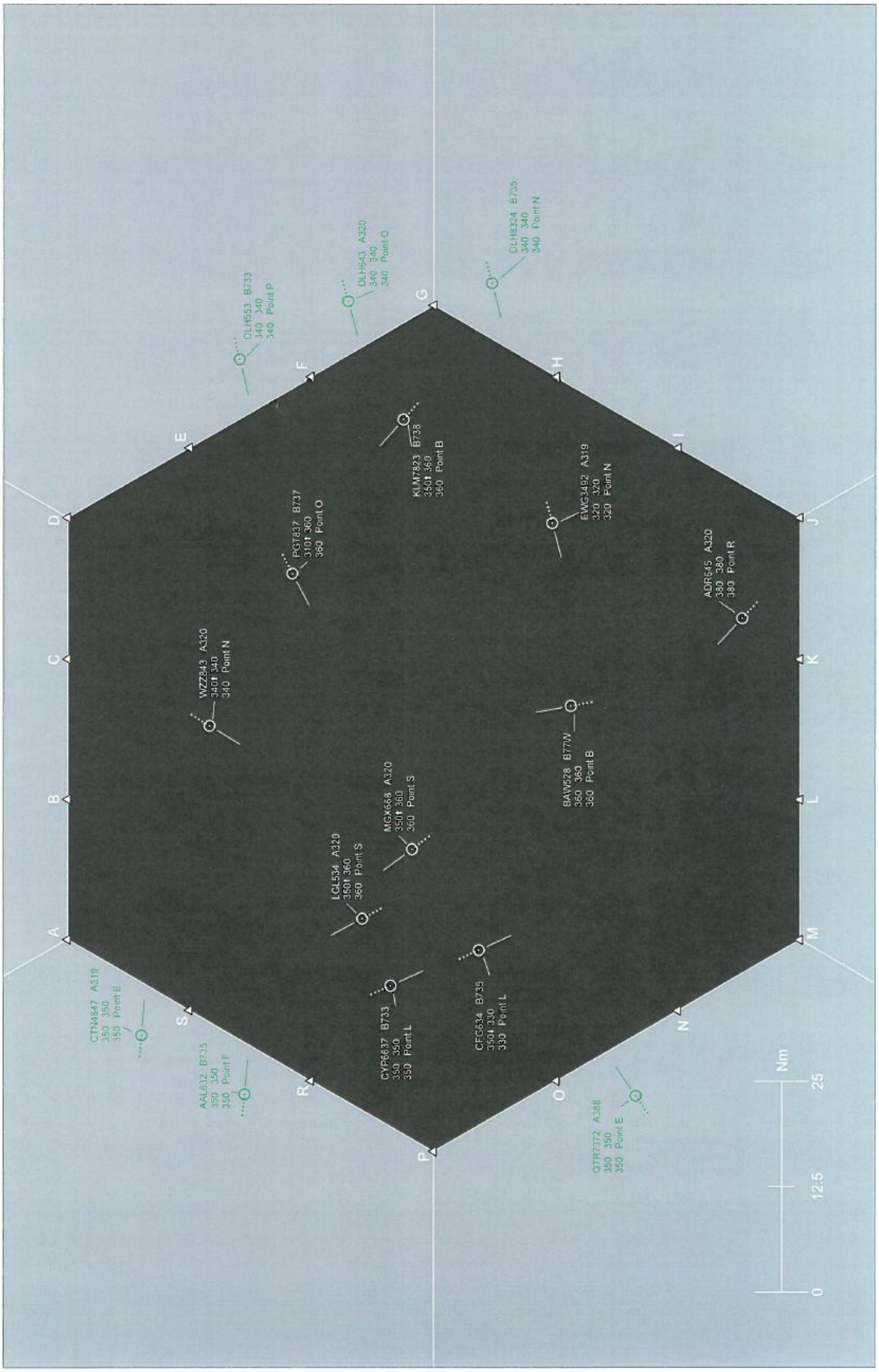
6781 V8 656 B17 373 P26 364

Traffic situation B18

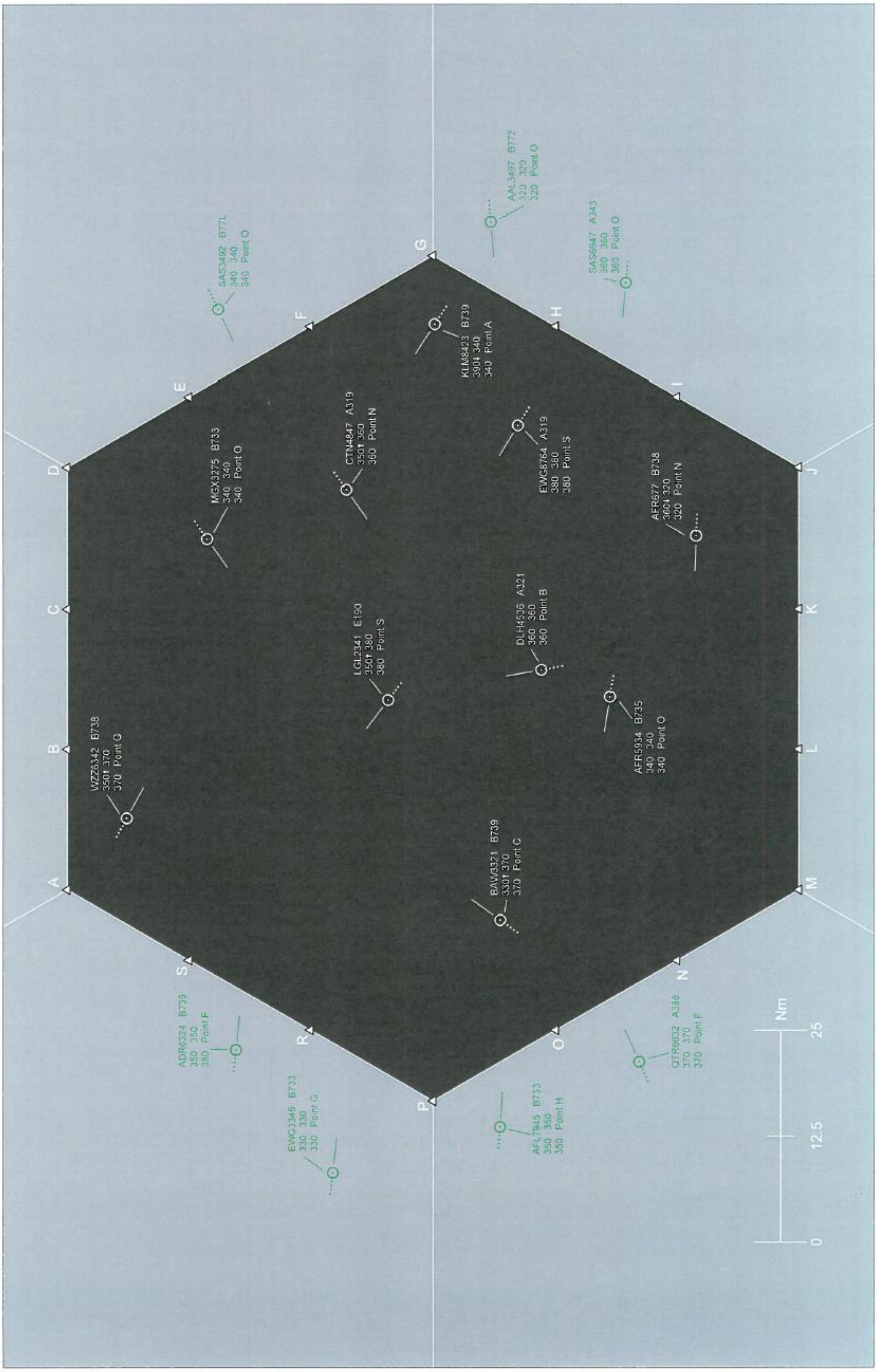


3194 T3 282 B19 371 Z72 787

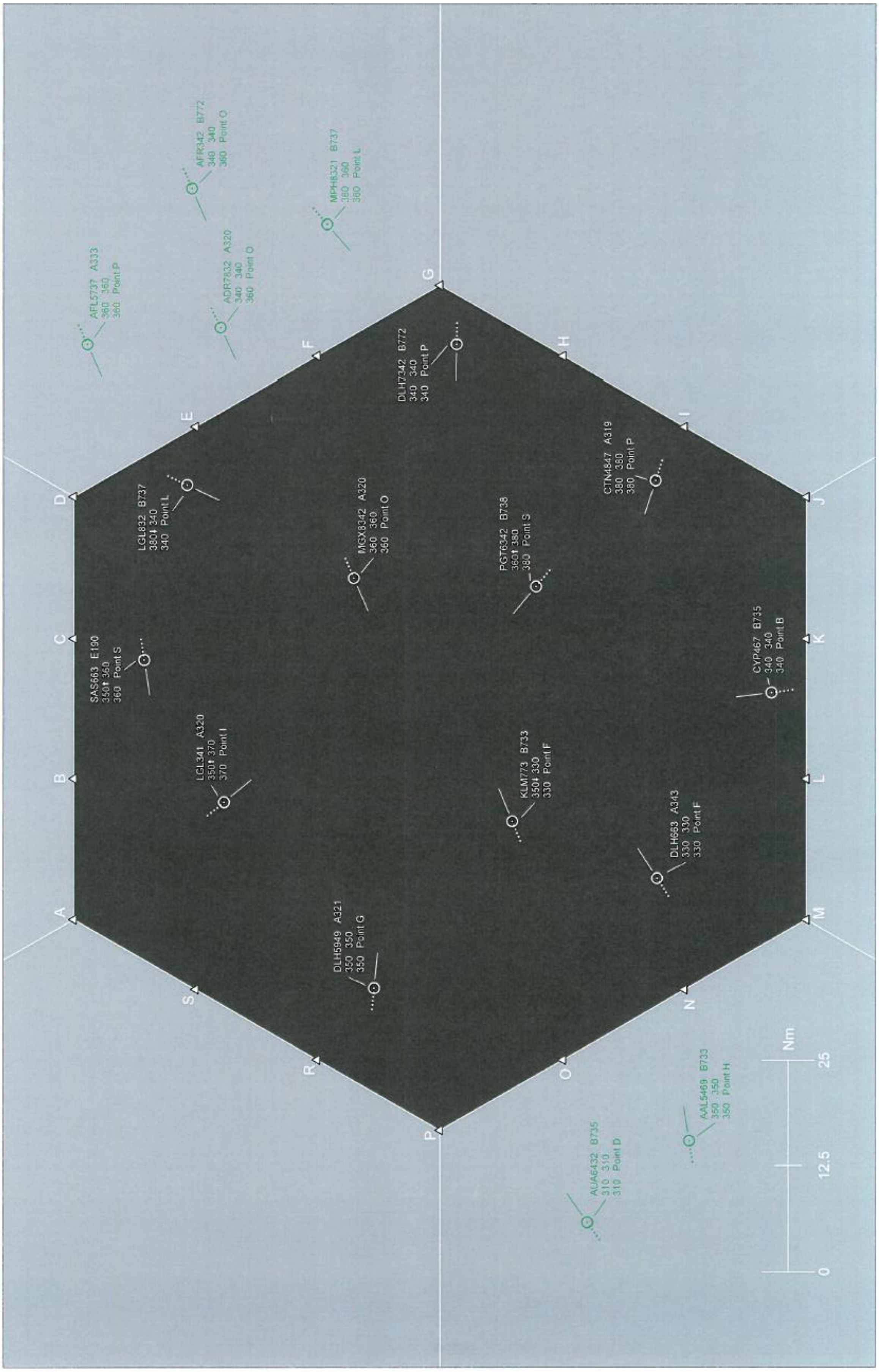
Traffic situation B19



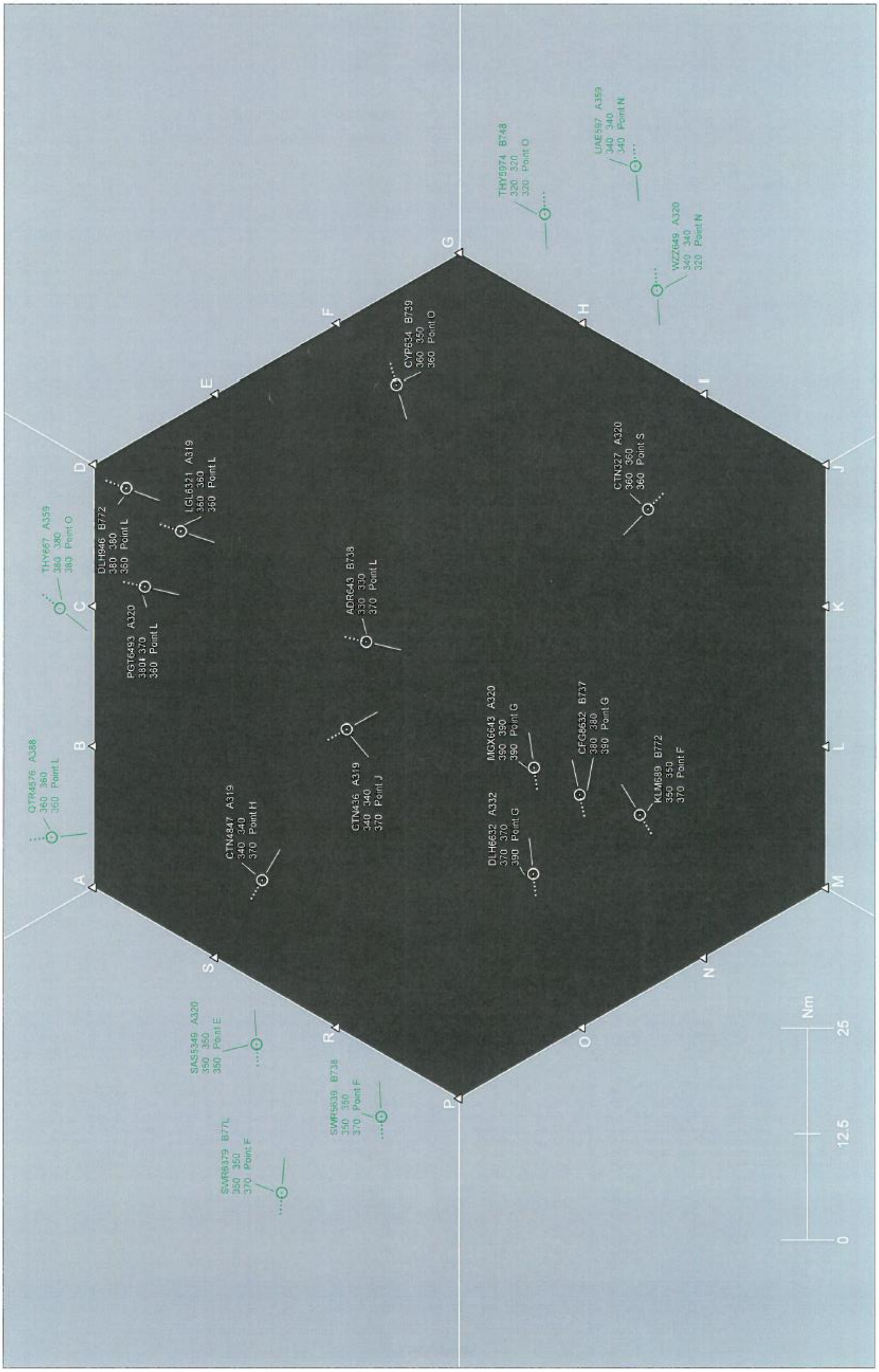
Traffic situation B20



Traffic situation B21

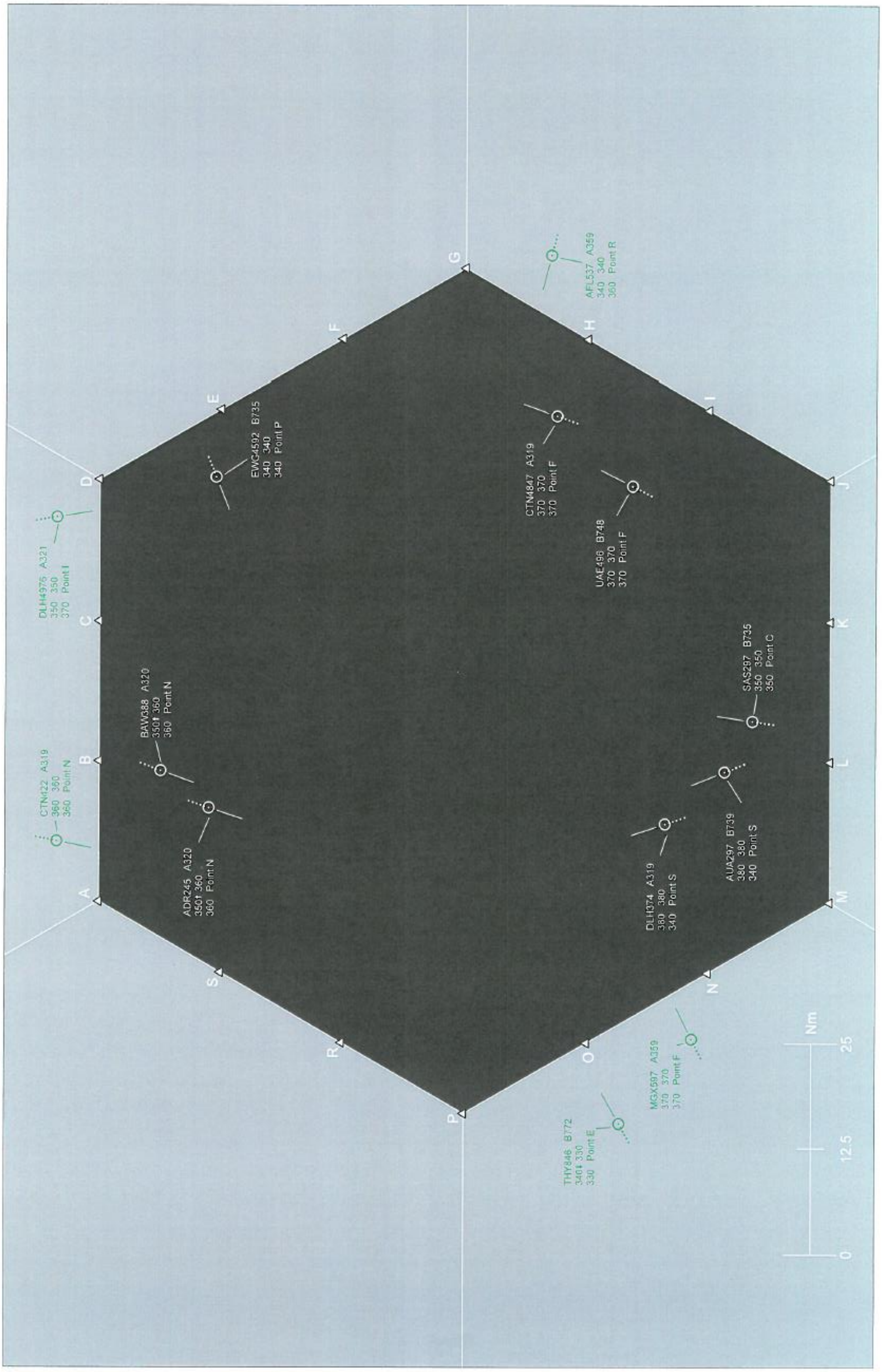


Traffic situation B22



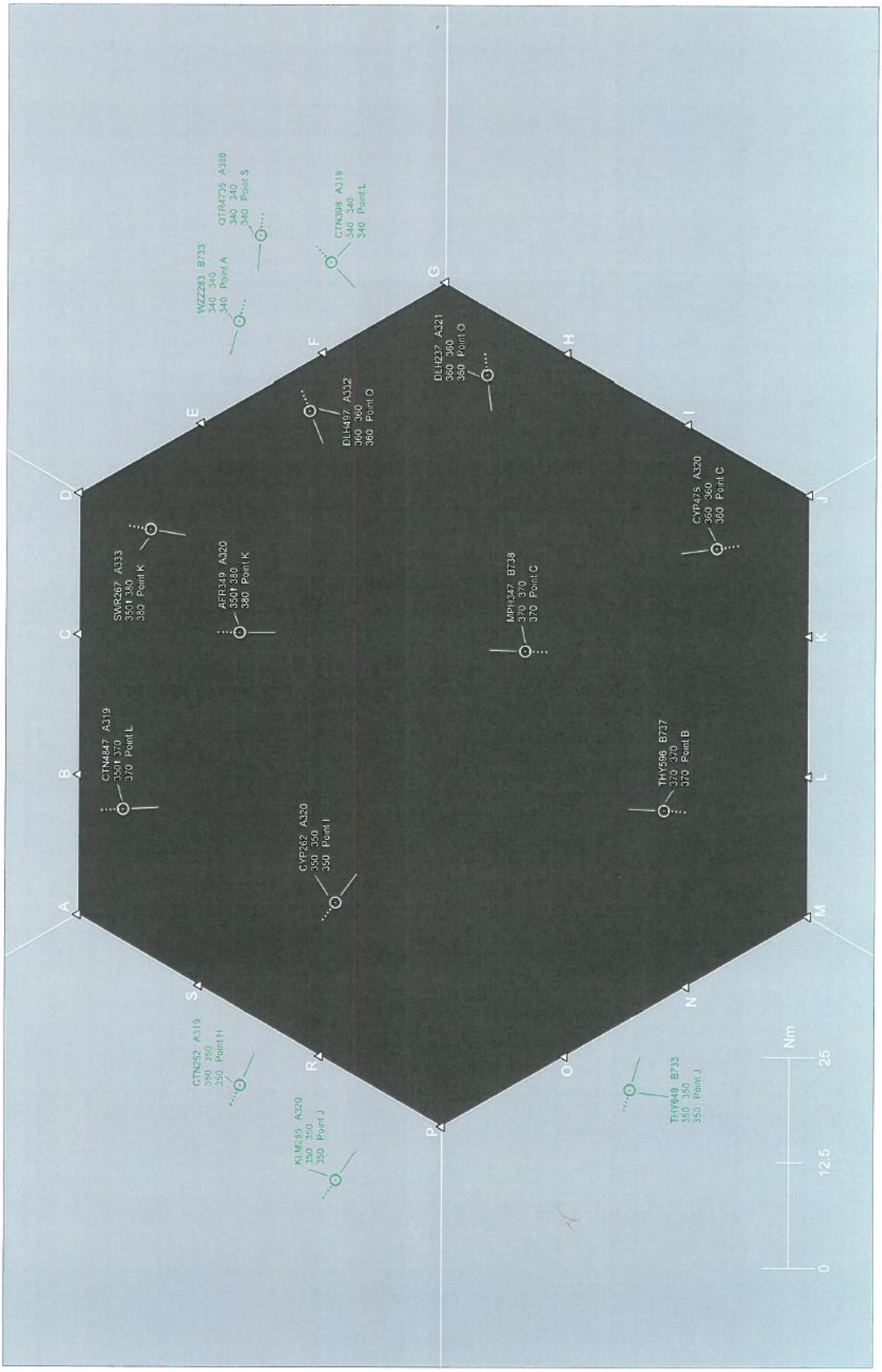
783 G7 681 B23 318 H97 824

Traffic situation B23

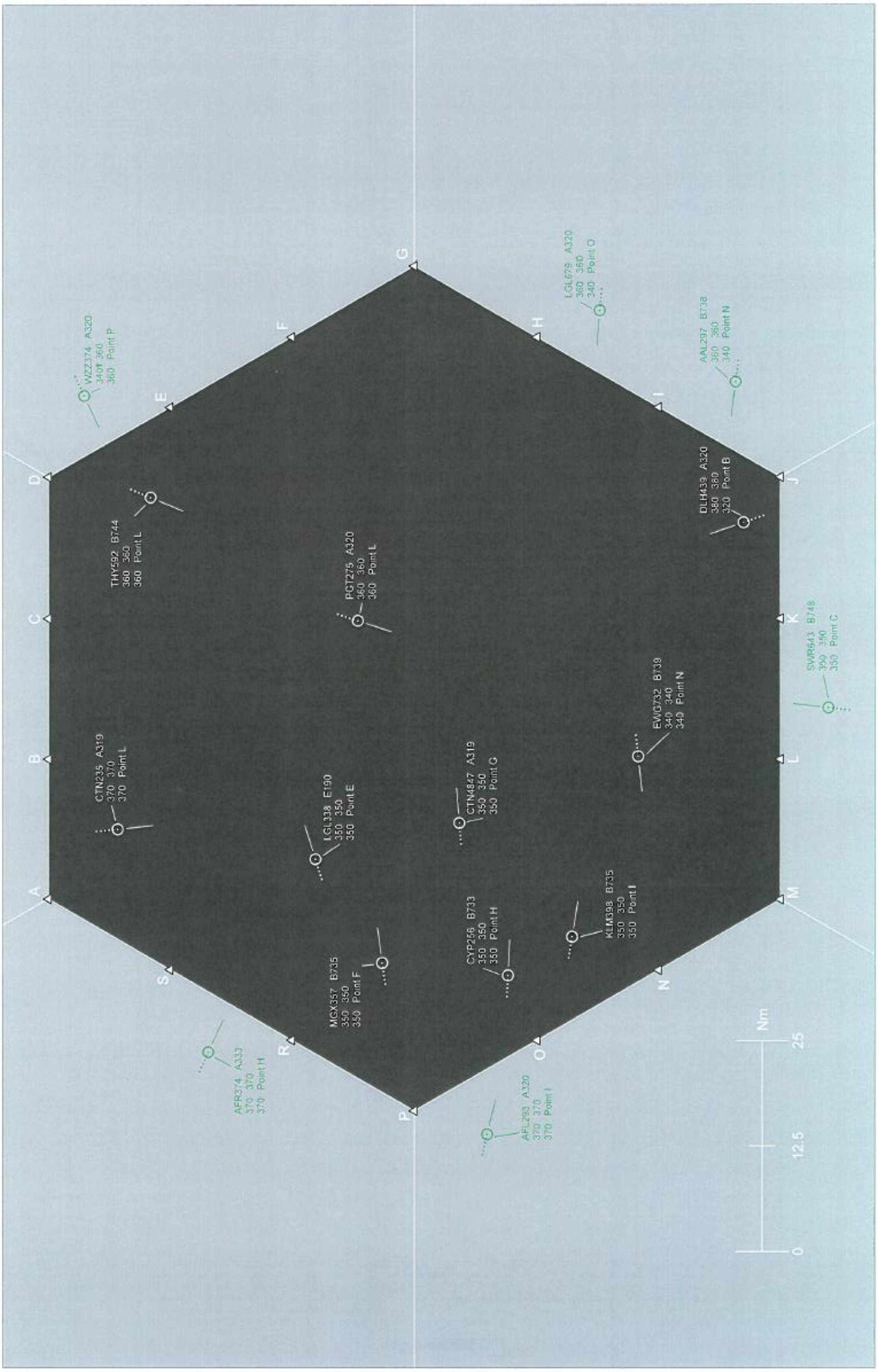


455 F5 464 B24 564 M55 545

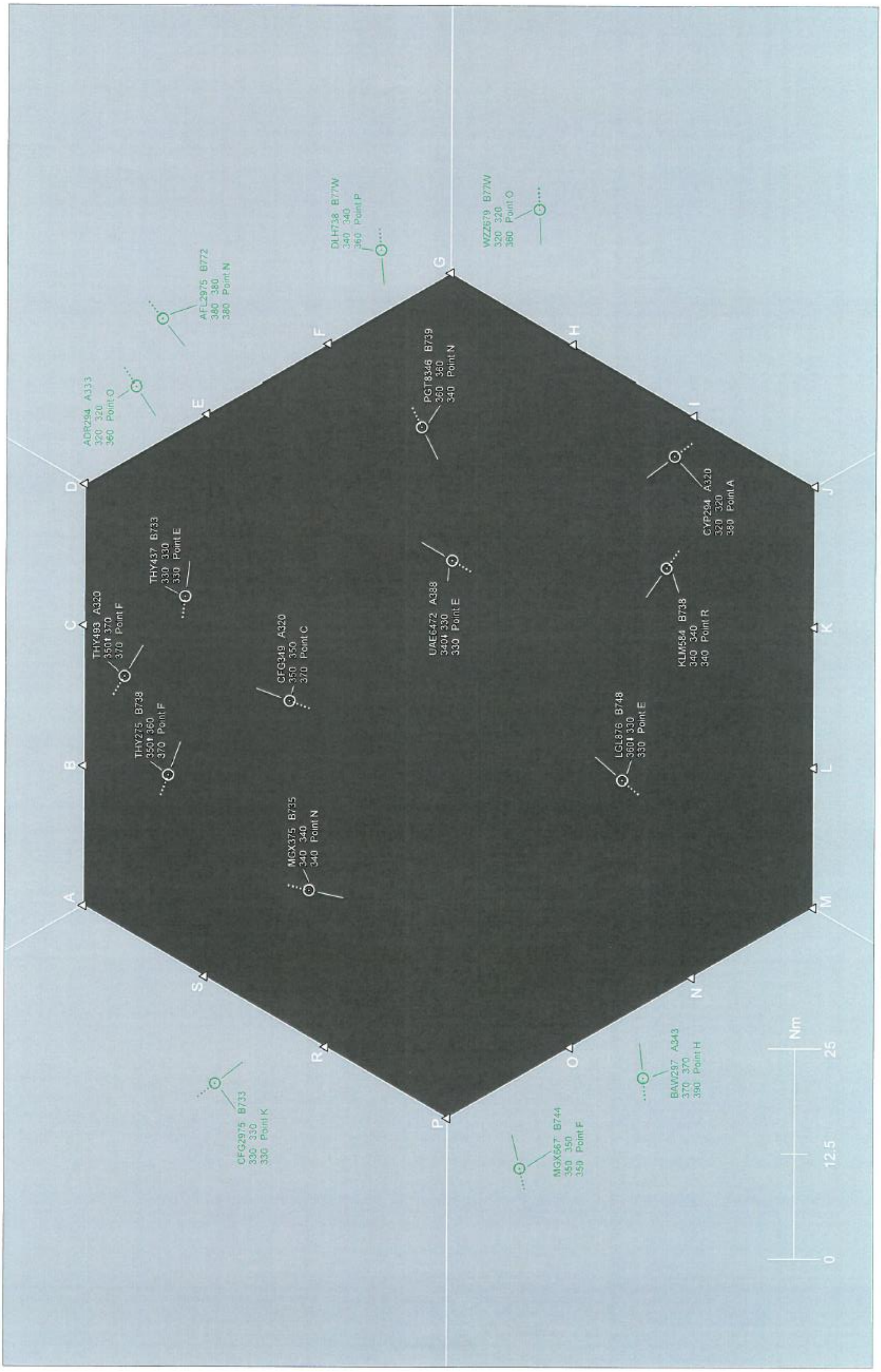
Traffic situation B24



Traffic situation B25

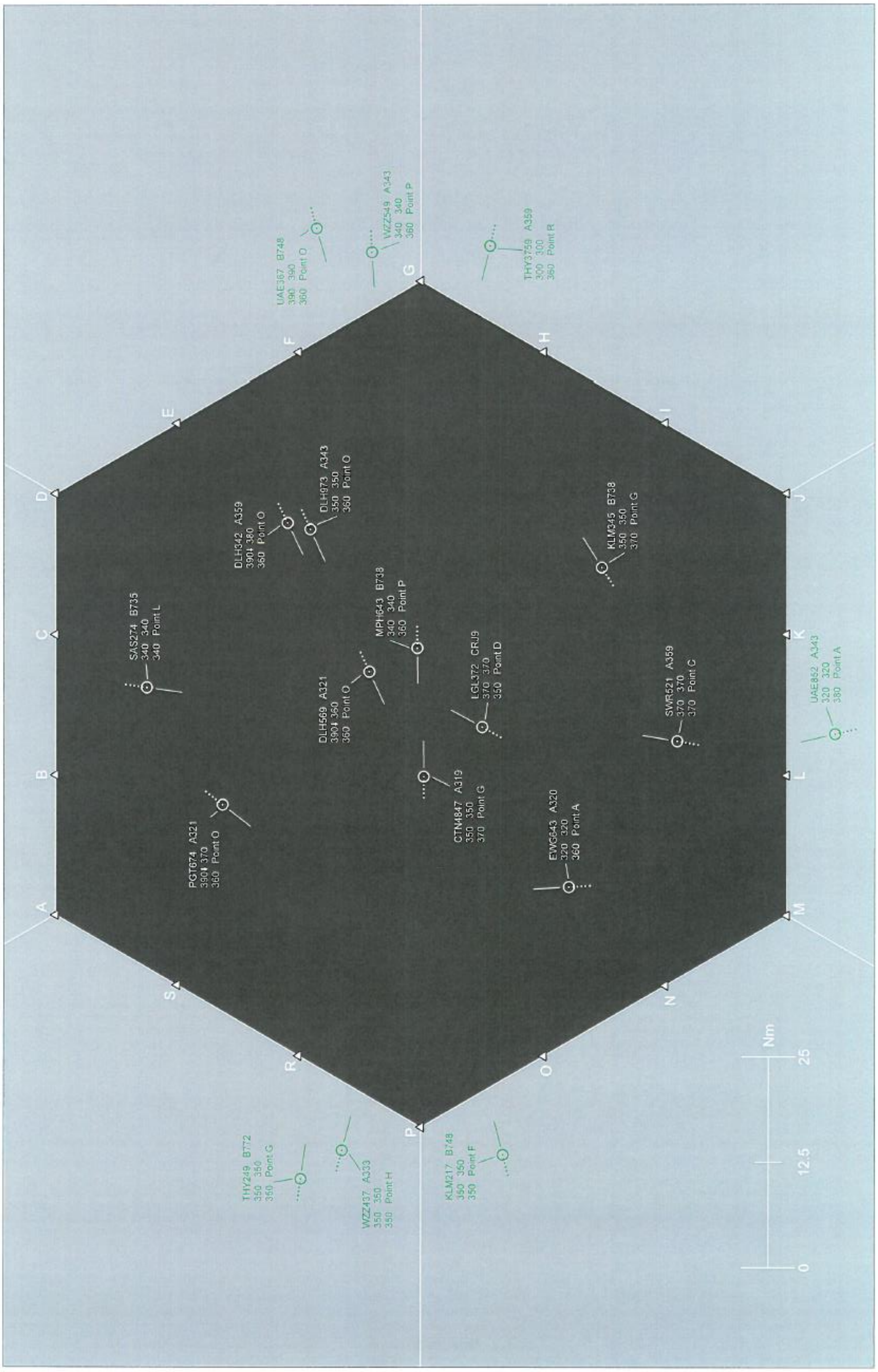


Traffic situation B26



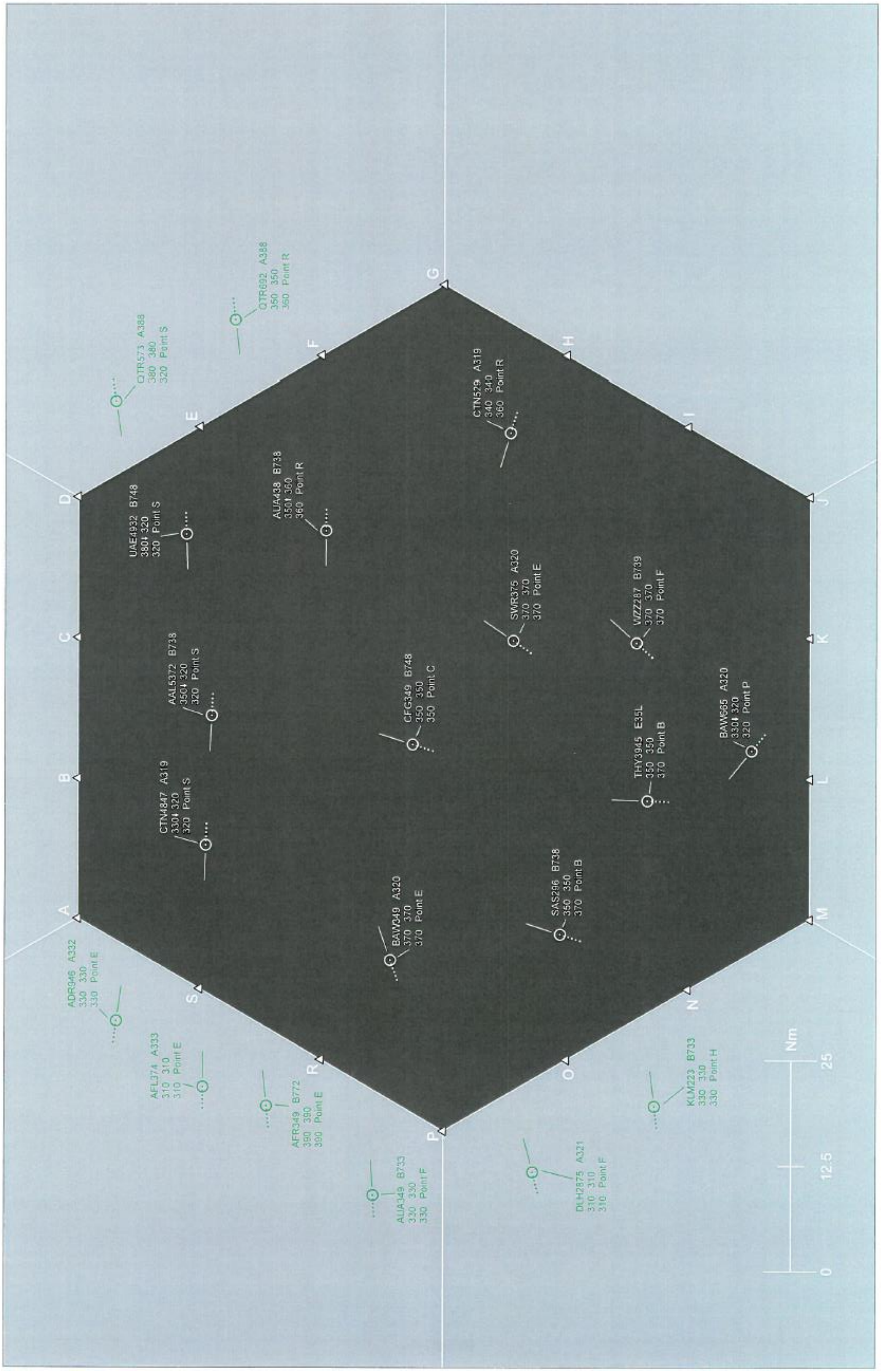
1973 H8 379 B27 194 J58 497

Traffic situation B27



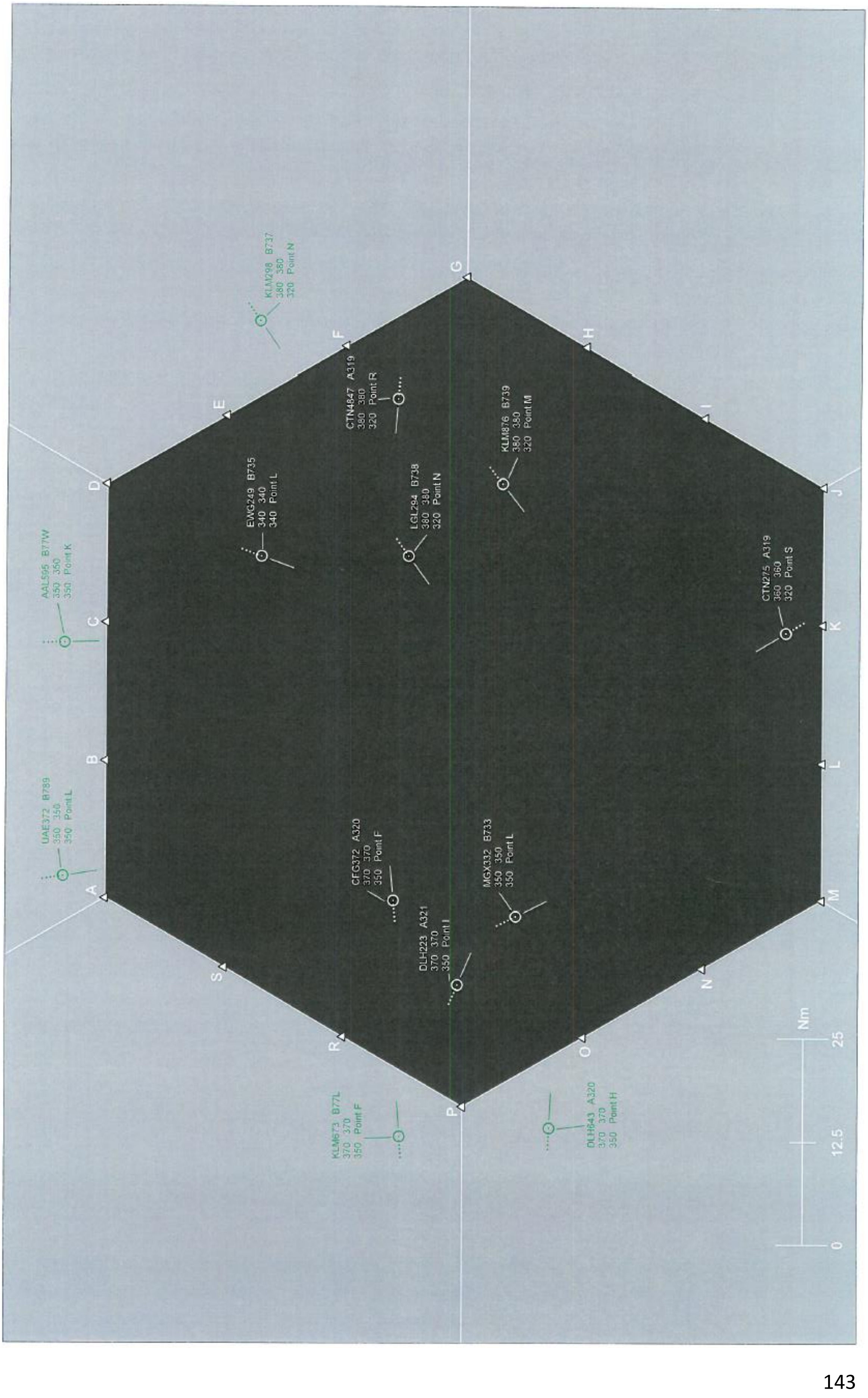
3194 T3 282 B28 371 272 787

Traffic situation B28

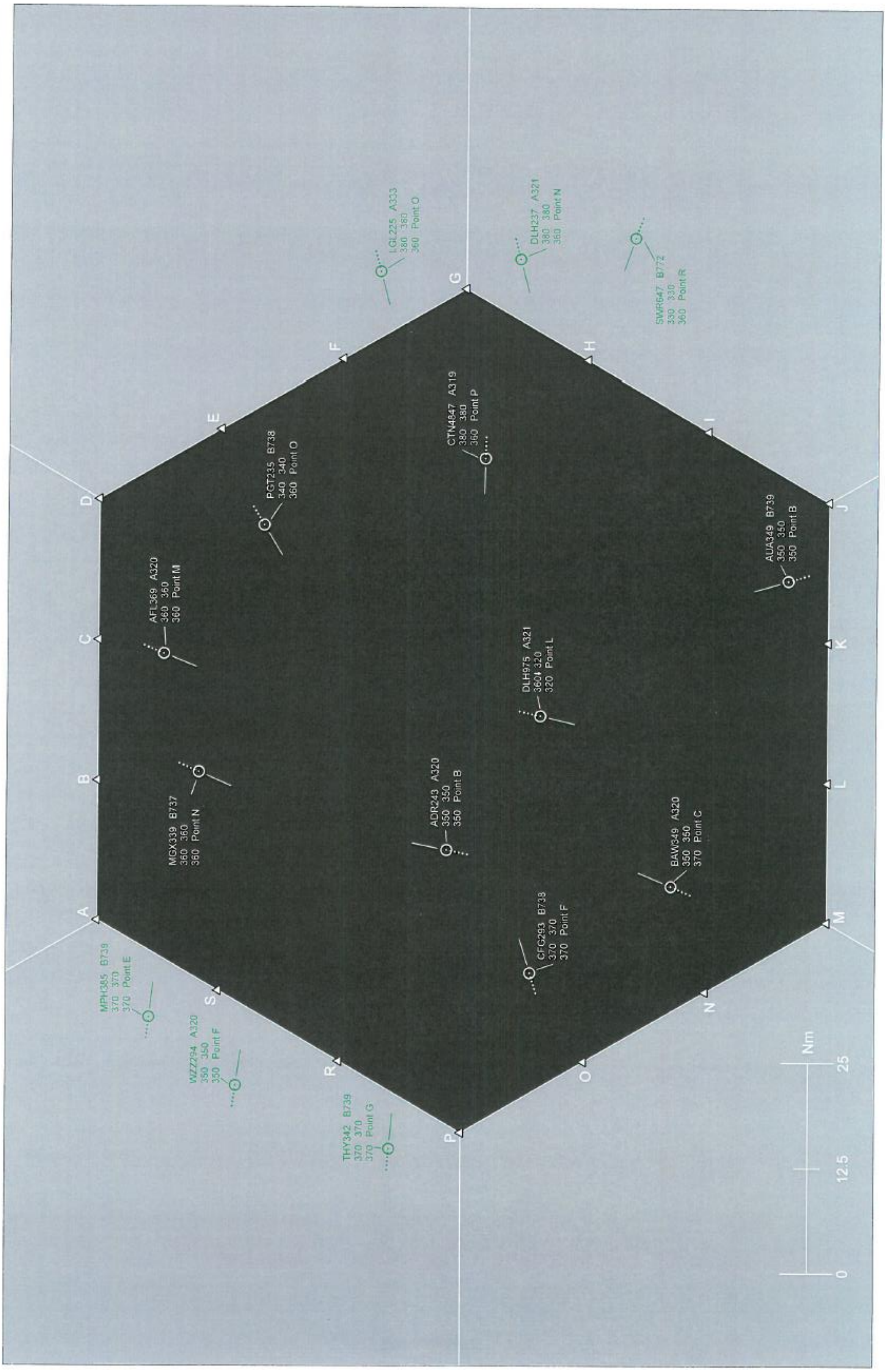


Traffic situation B29

678 V8 656 B29 373 P26 364

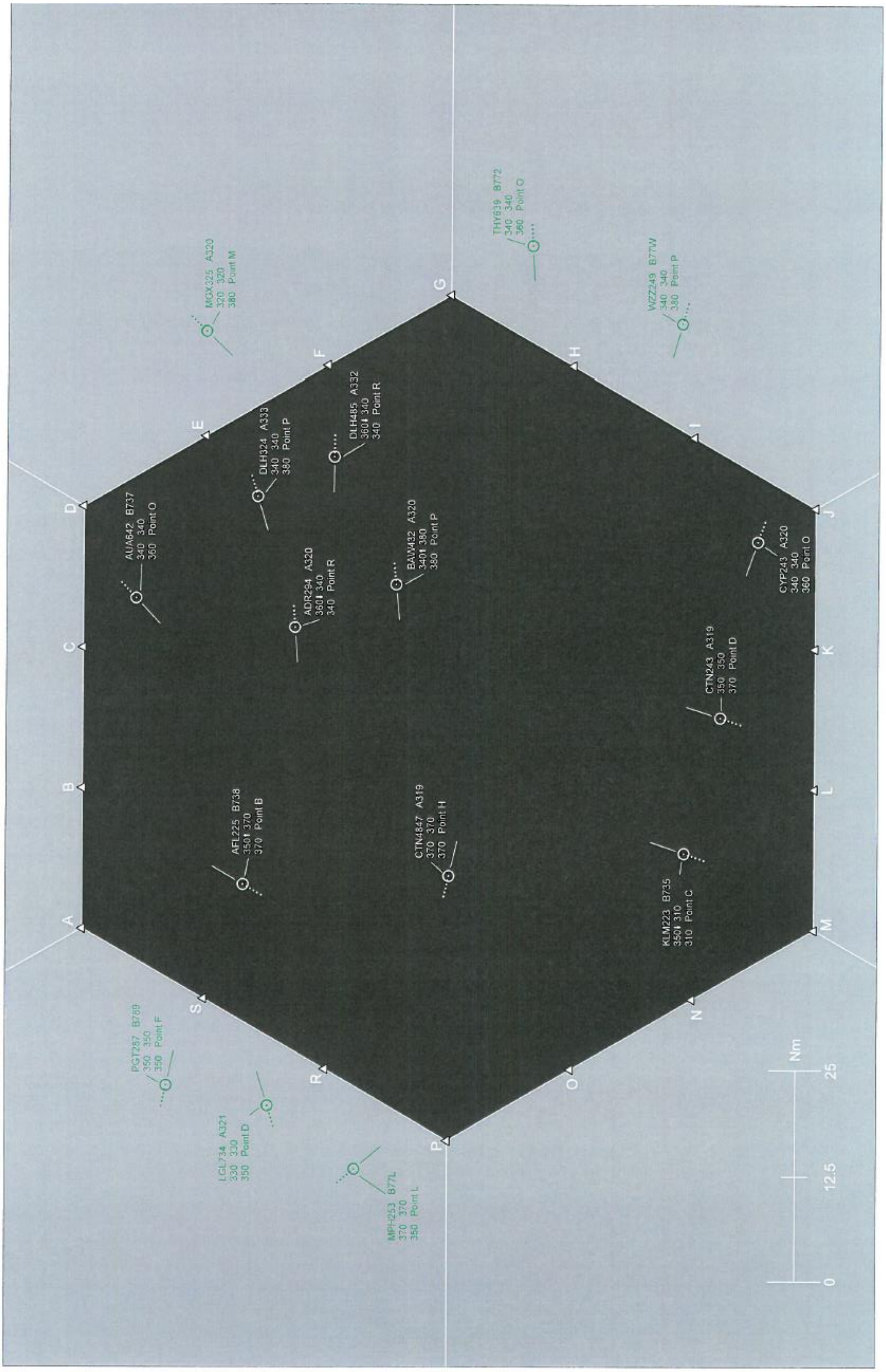


Traffic situation B30

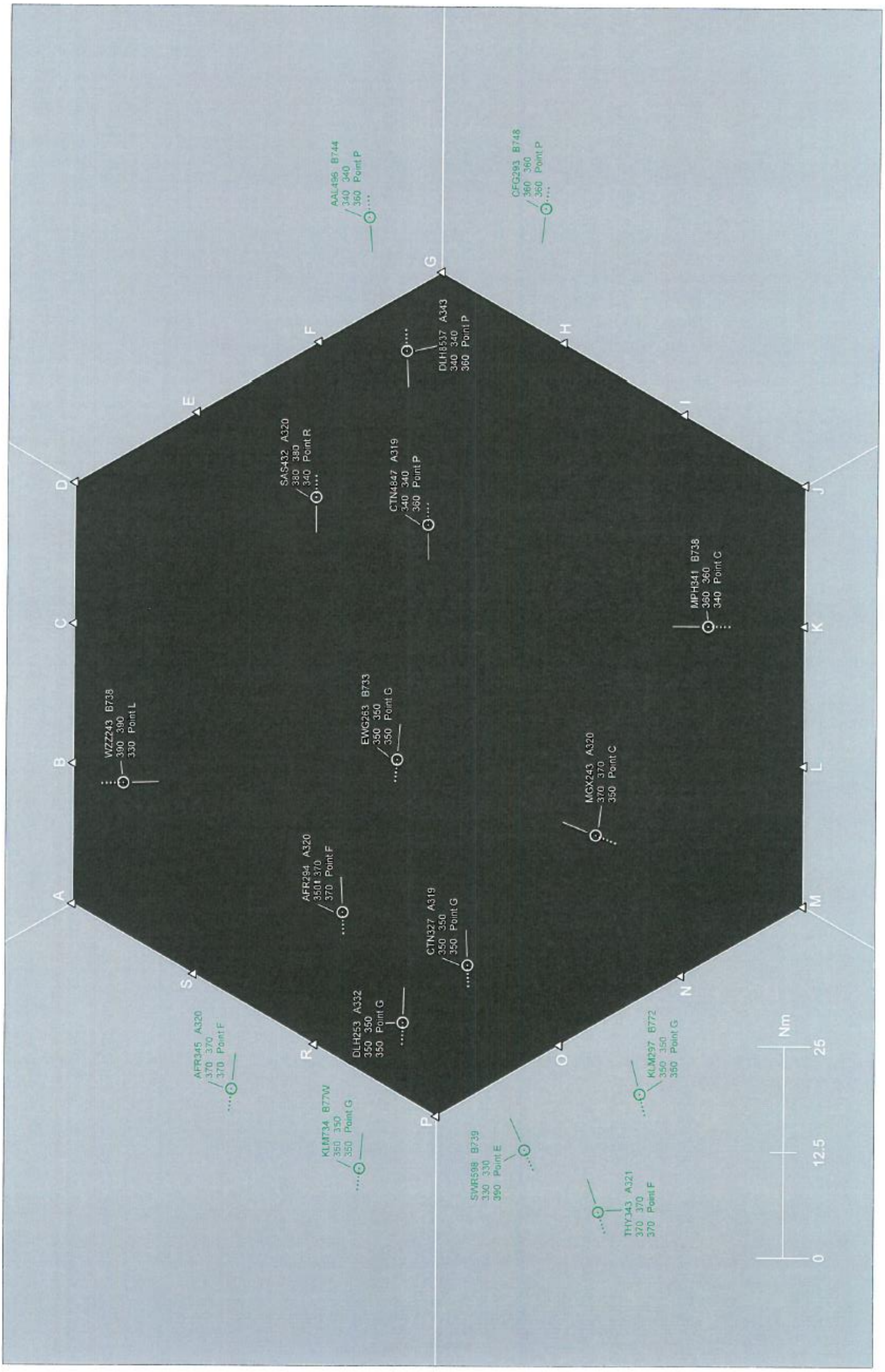


Traffic situation B31

3194 T3 282 B31 371 272 787

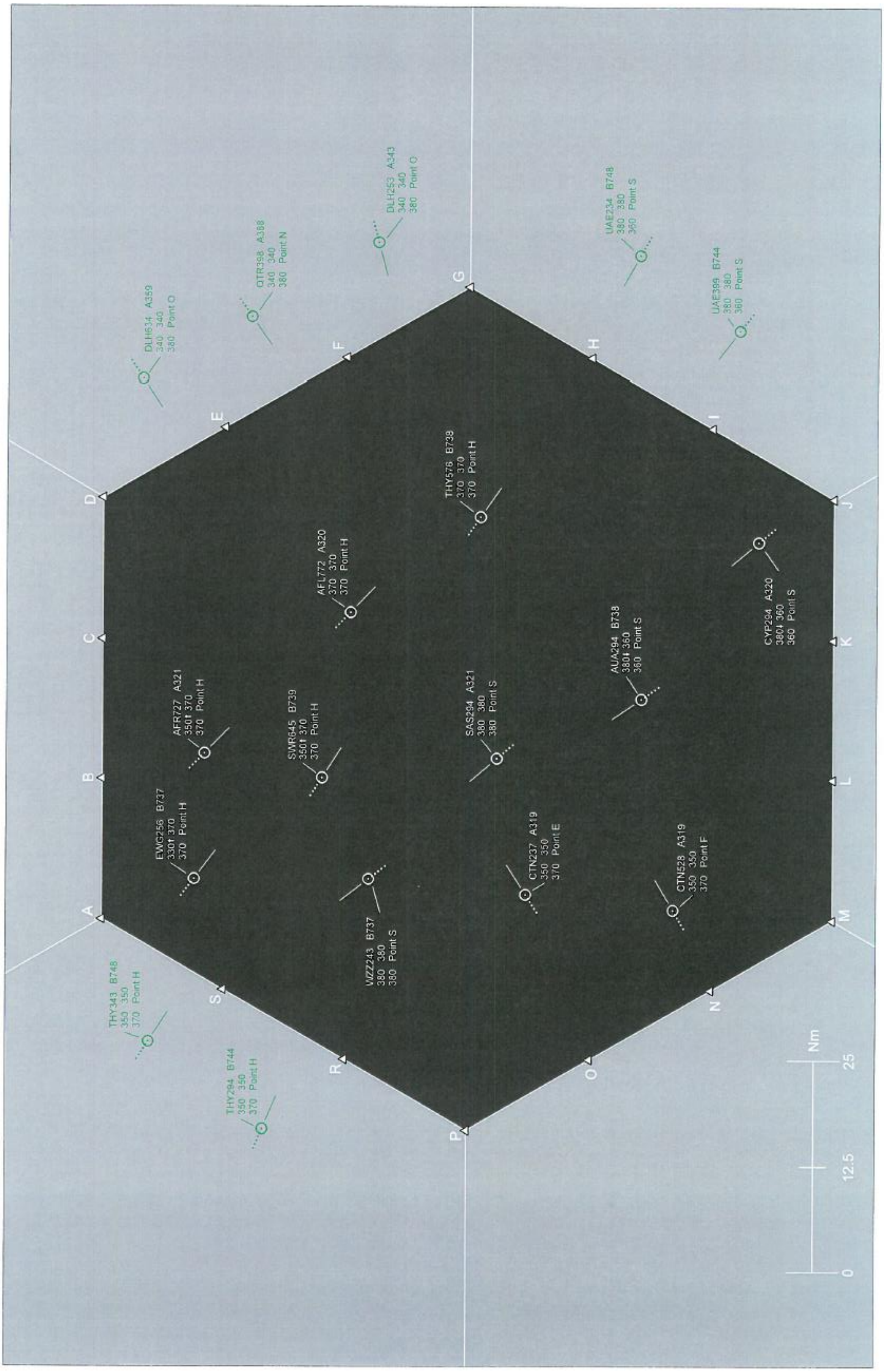


Traffic situation B32



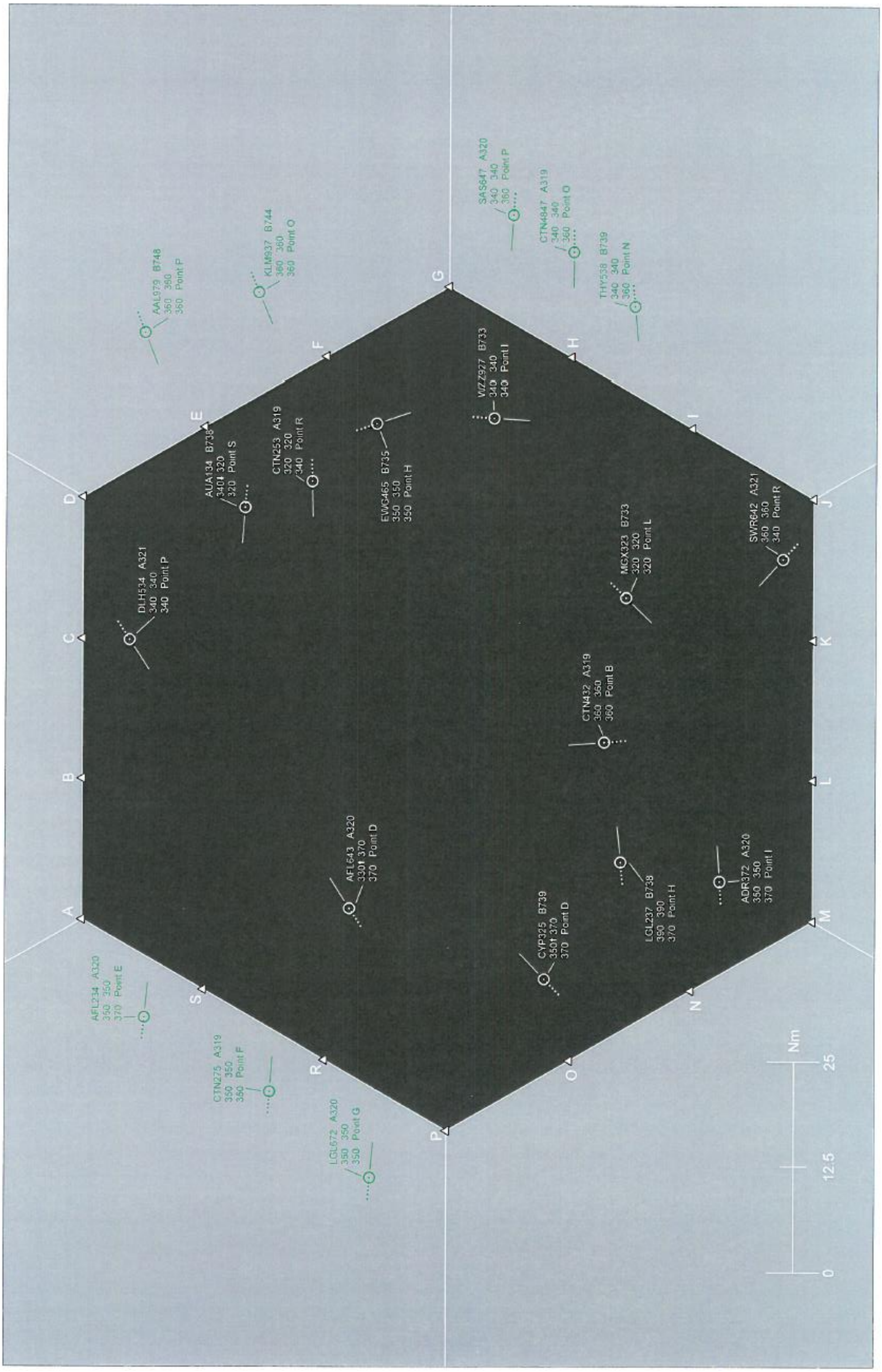
671 G3 696 B33 191 F78 197

Traffic situation B33



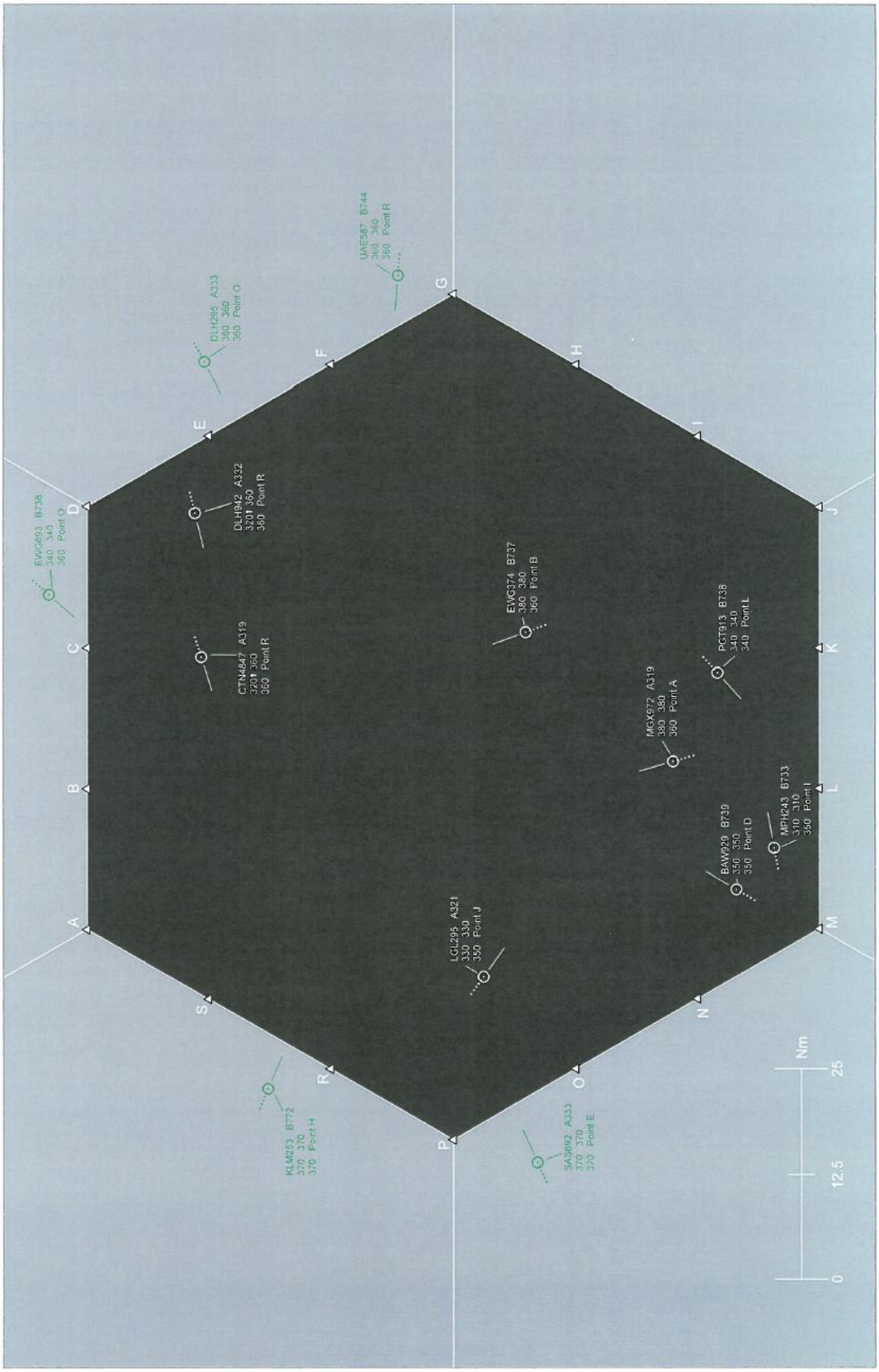
Traffic situation B34

2094 V8 161 B34 436 529 197

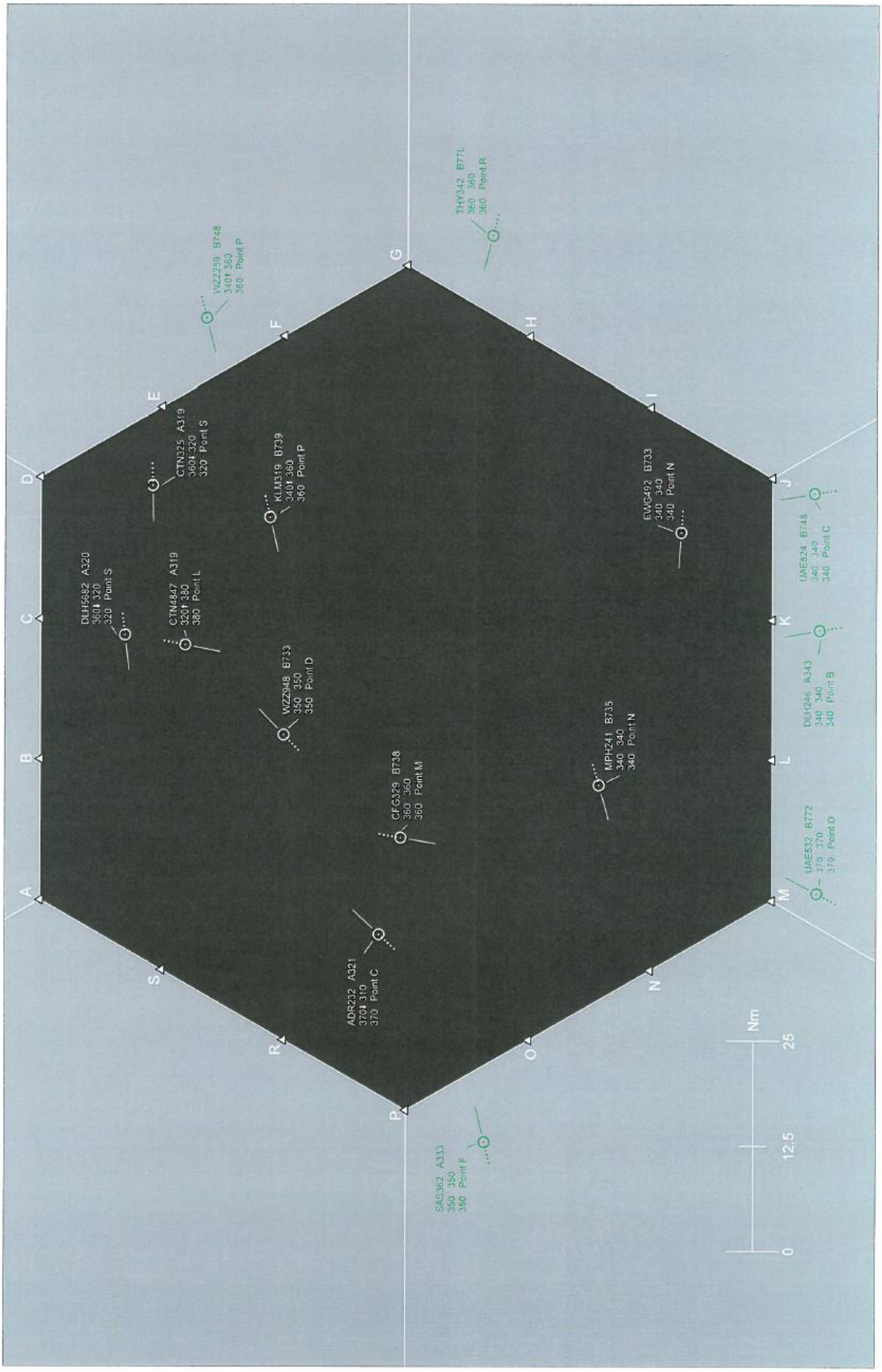


Traffic situation B35

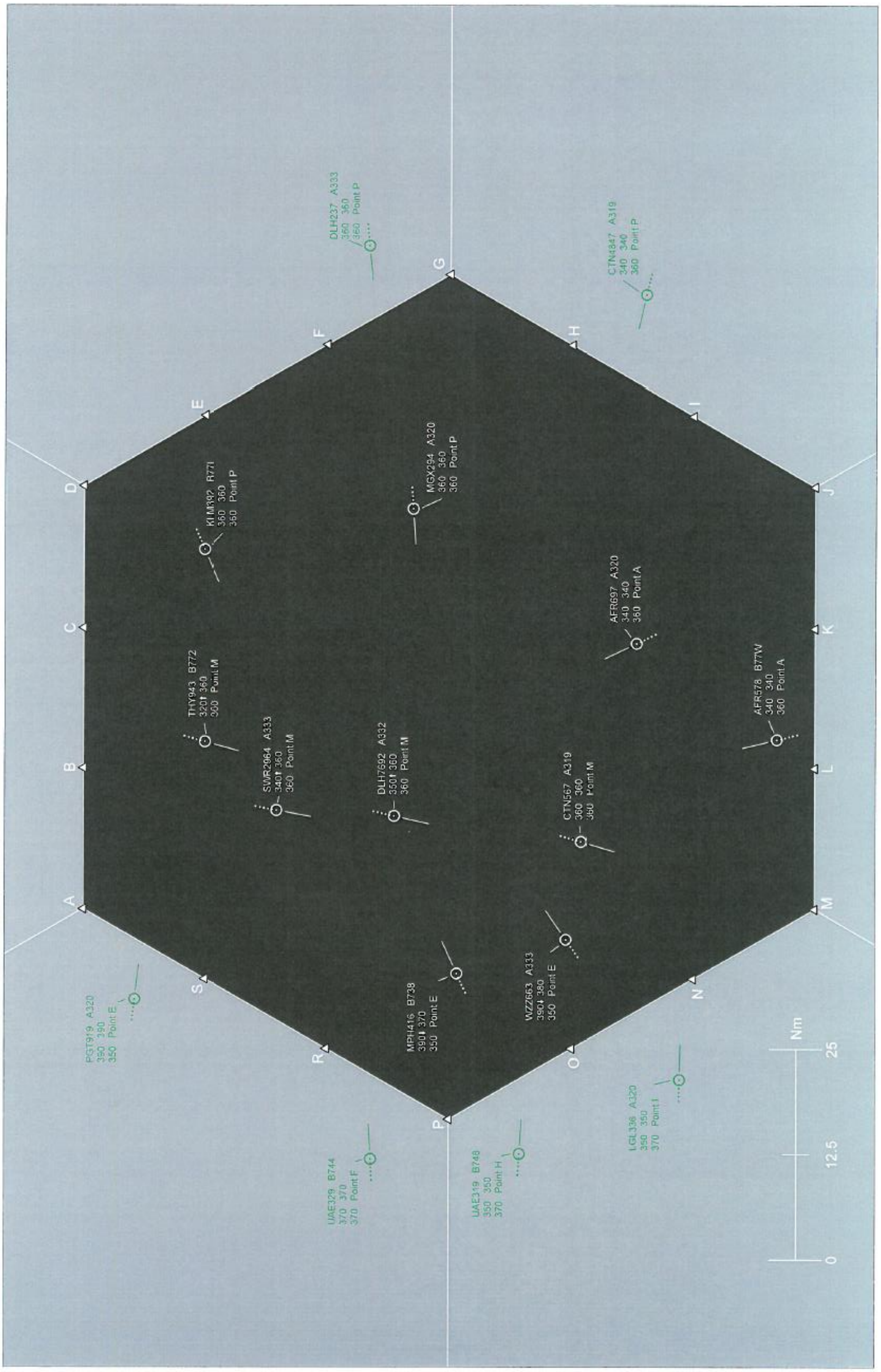
8296 I7 271 B35 371 H97 352



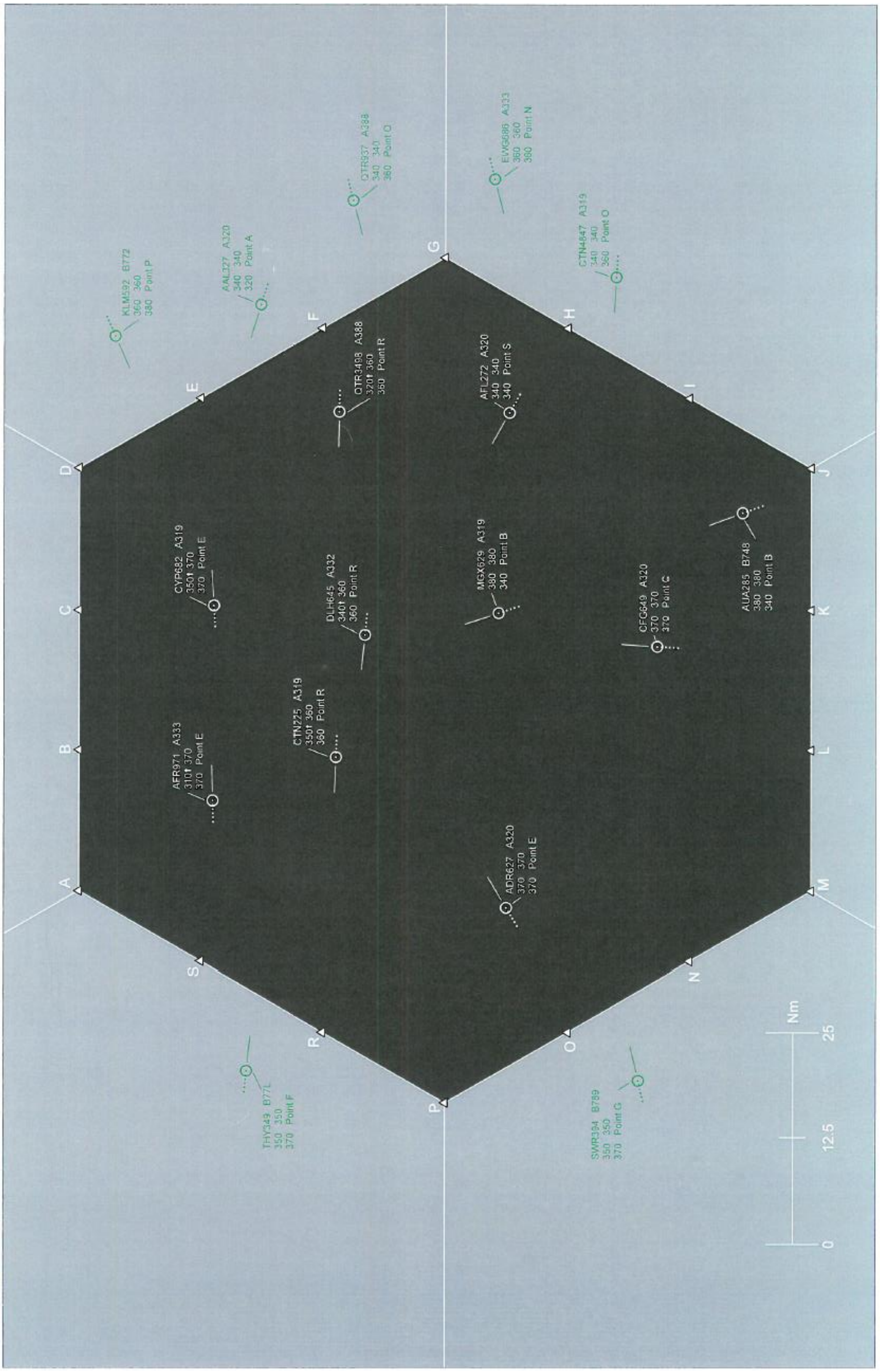
Traffic situation B36



Traffic situation B37

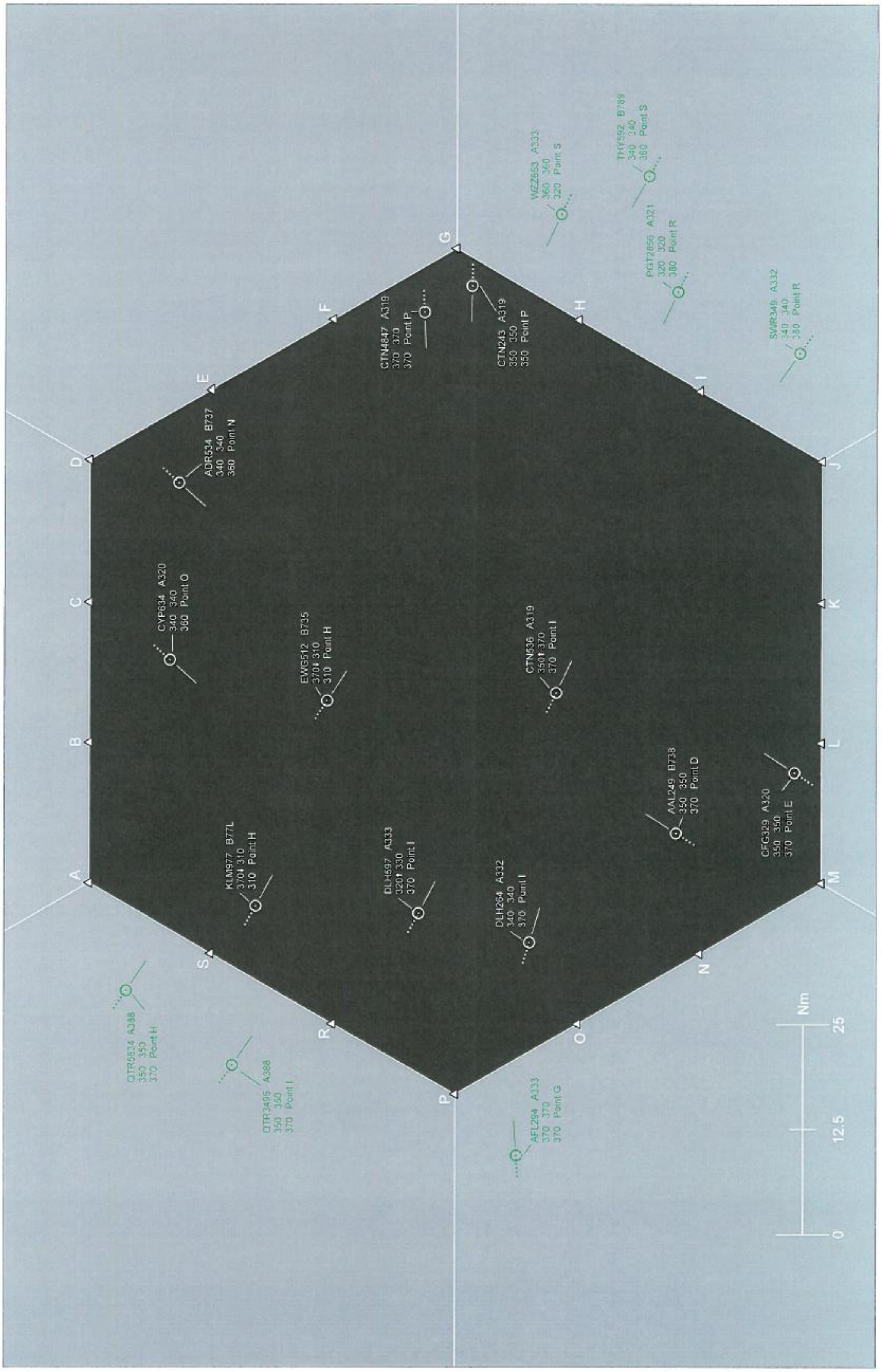


Traffic situation B38

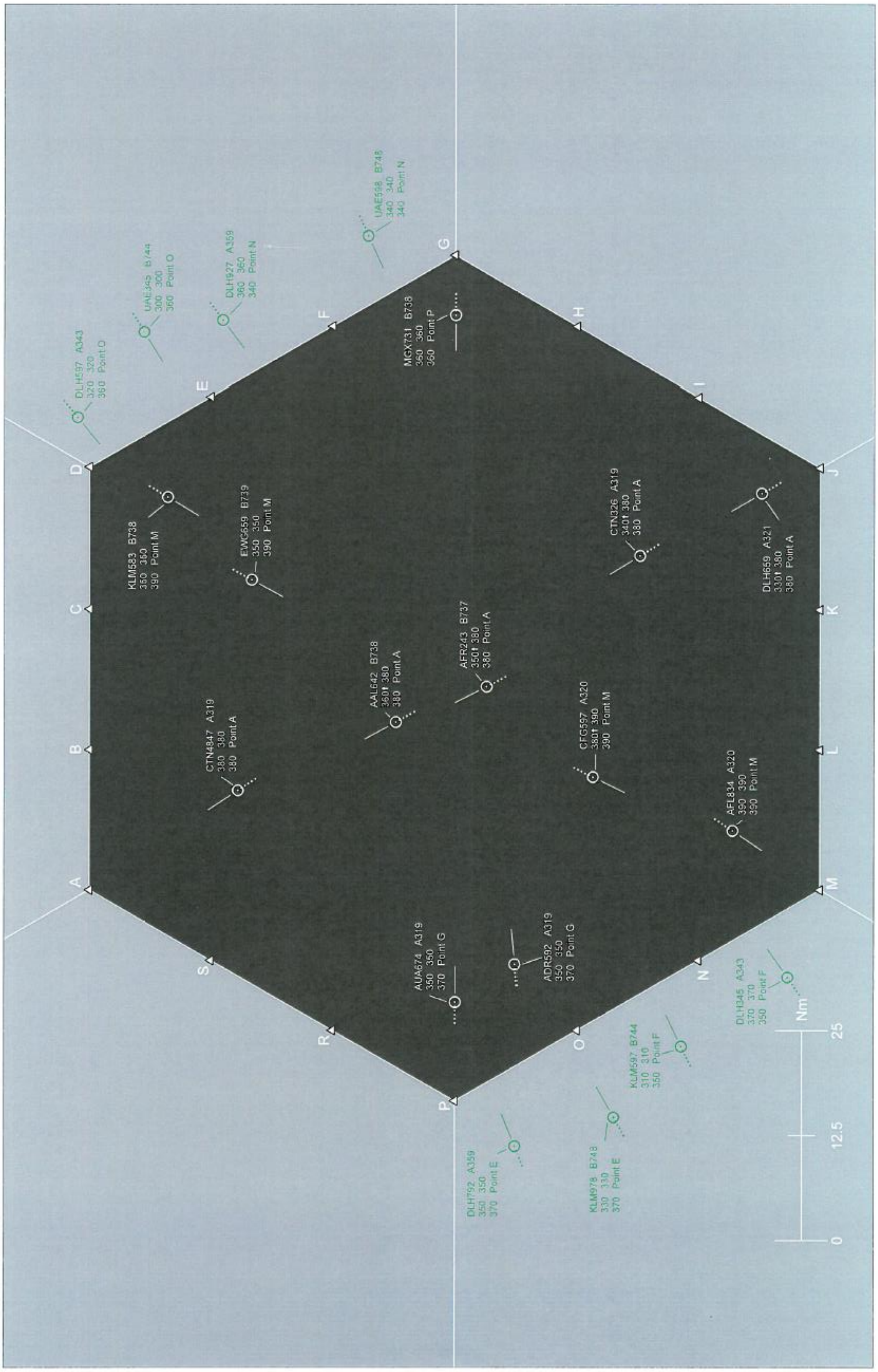


781 V8 656 B39 373 P26 364

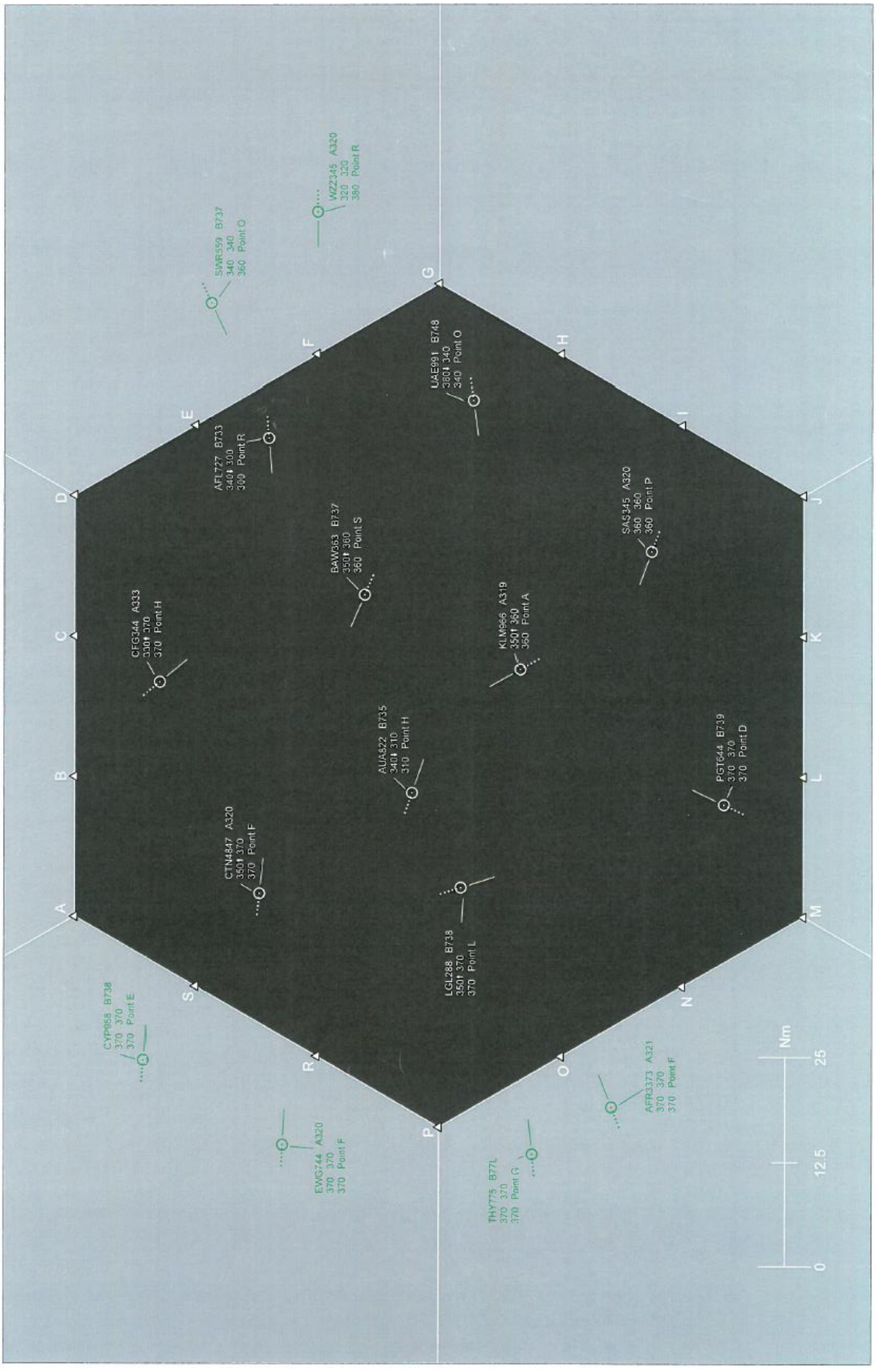
Traffic situation B39



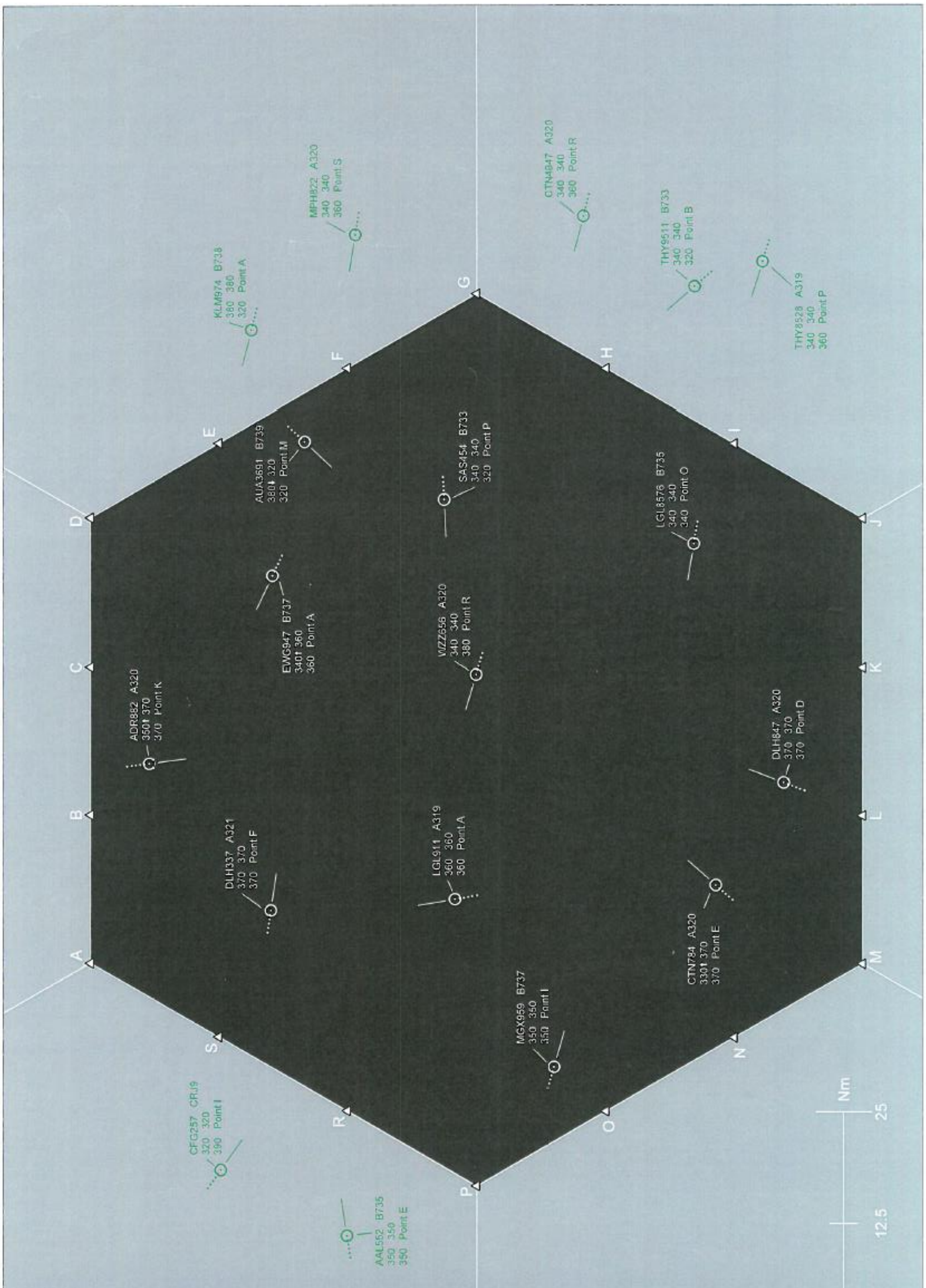
Traffic situation B40



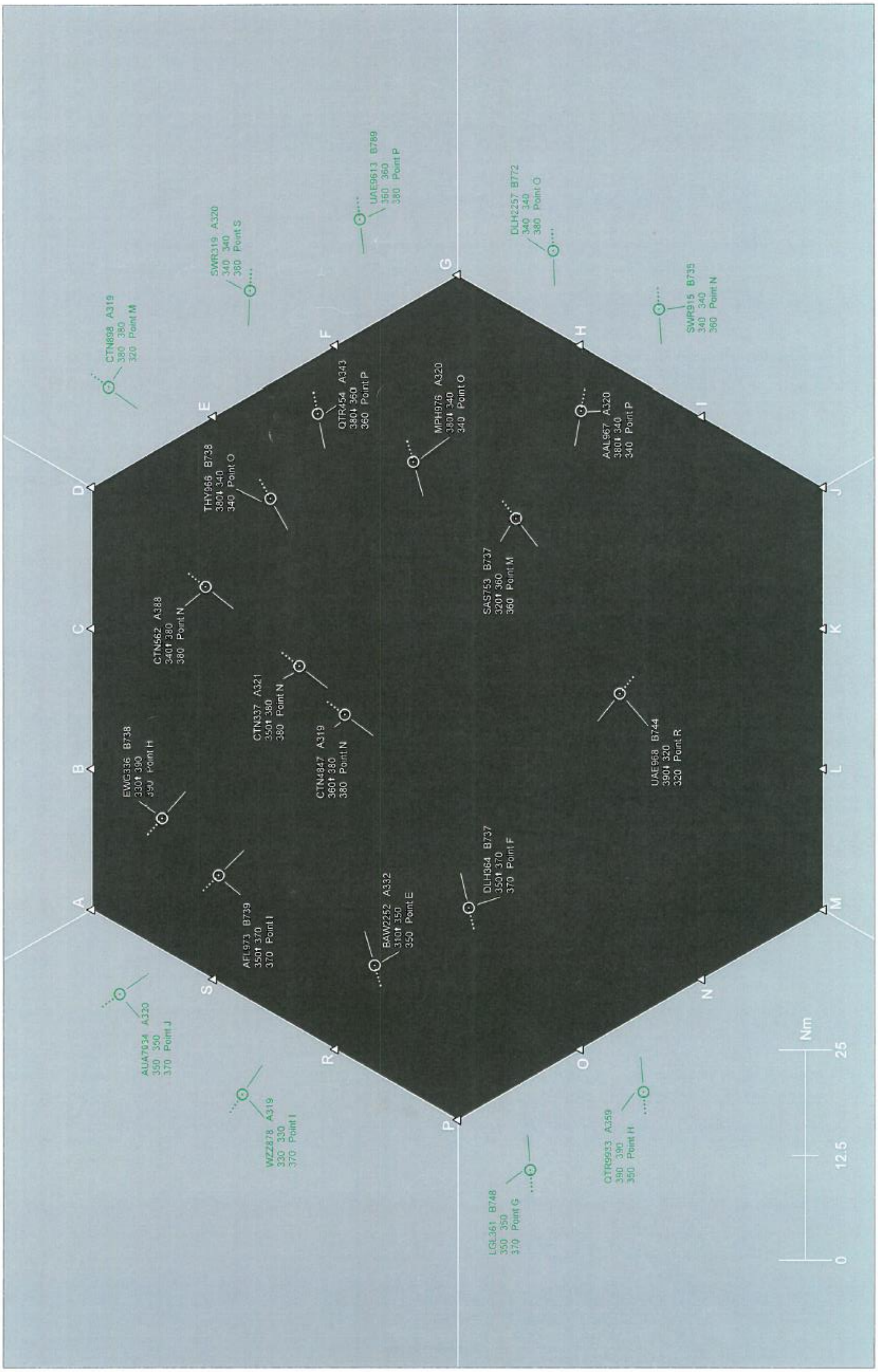
Traffic situation C1



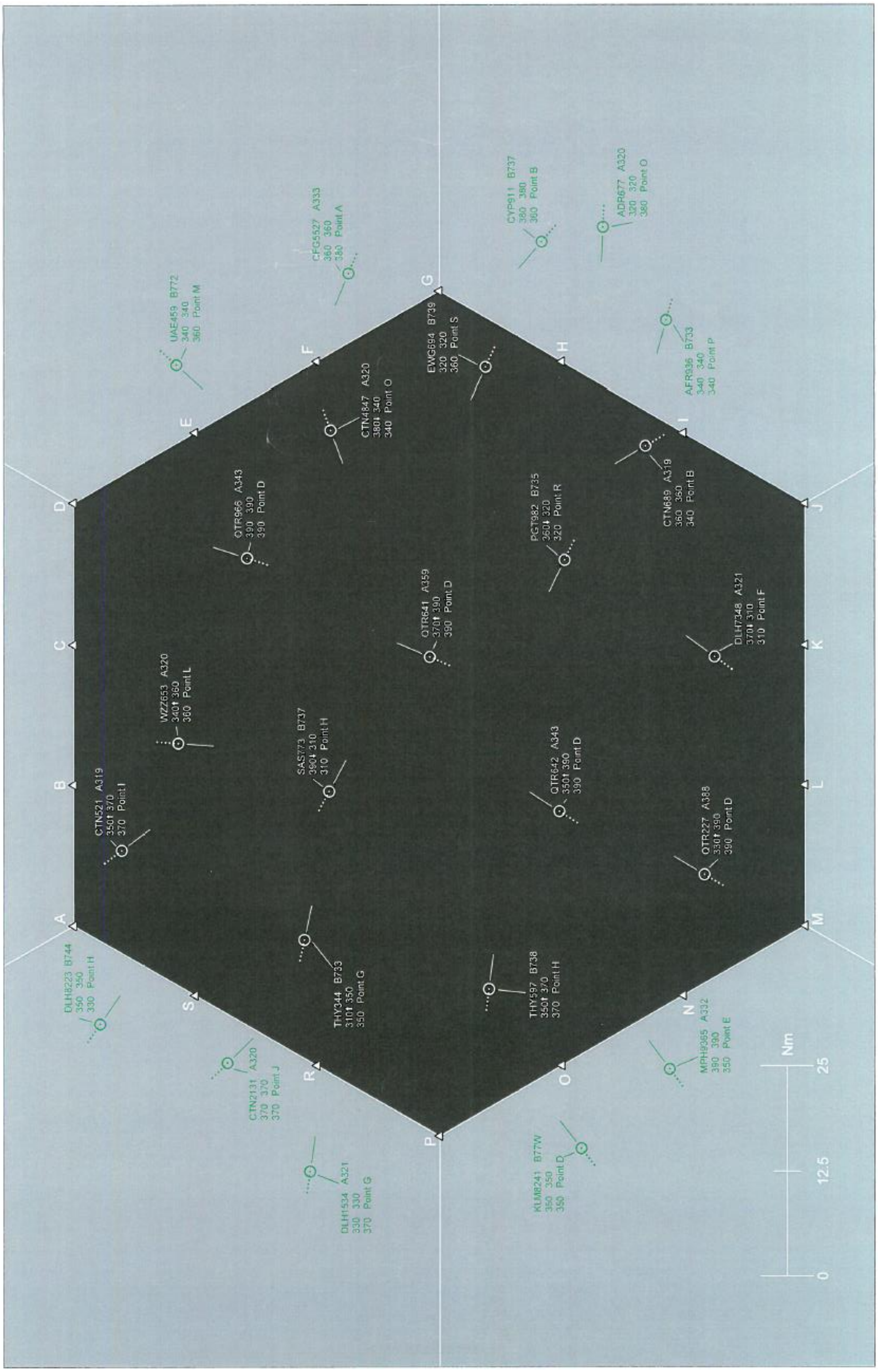
Traffic situation C2



Traffic situation C3

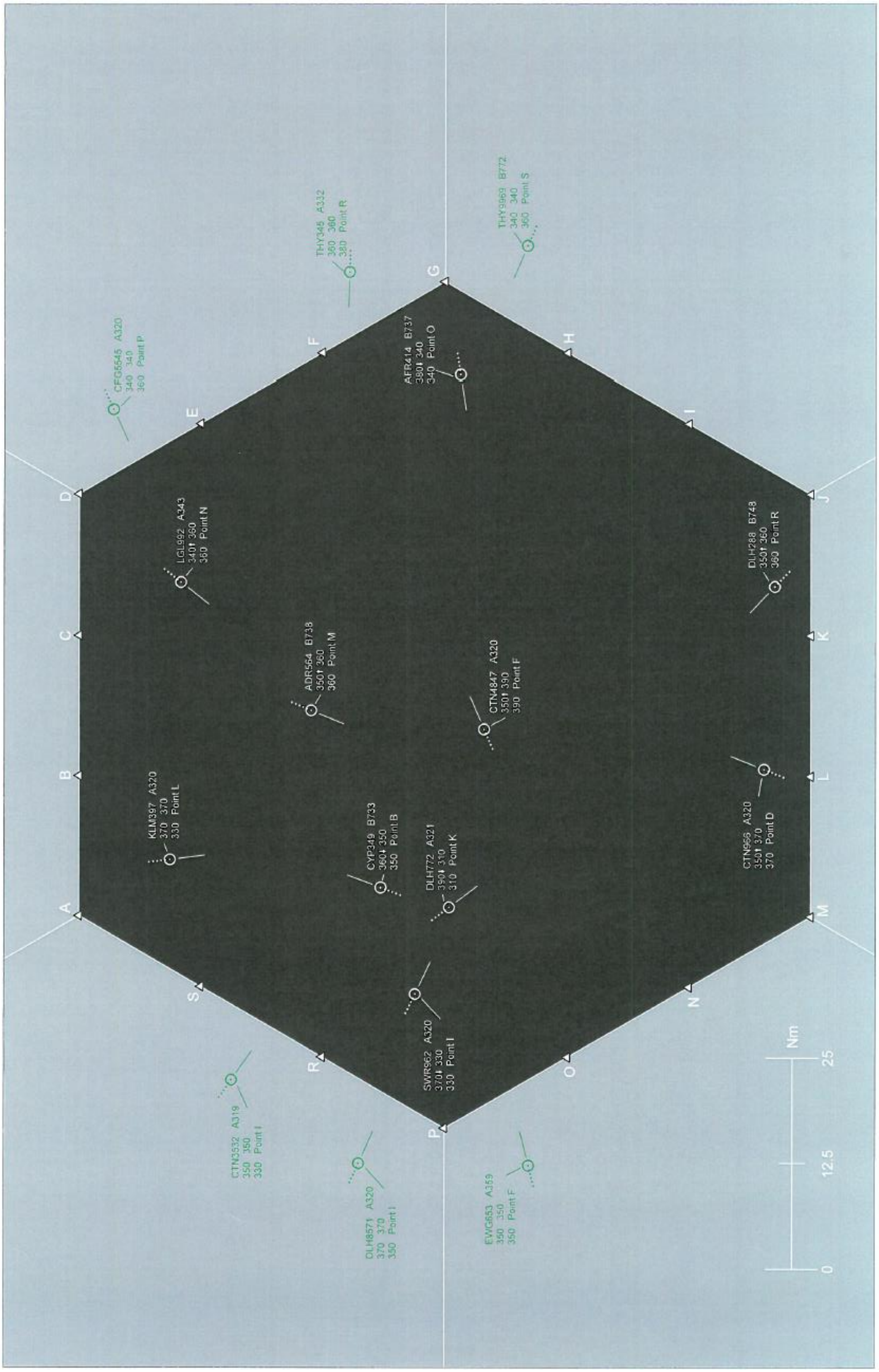


Traffic situation C4



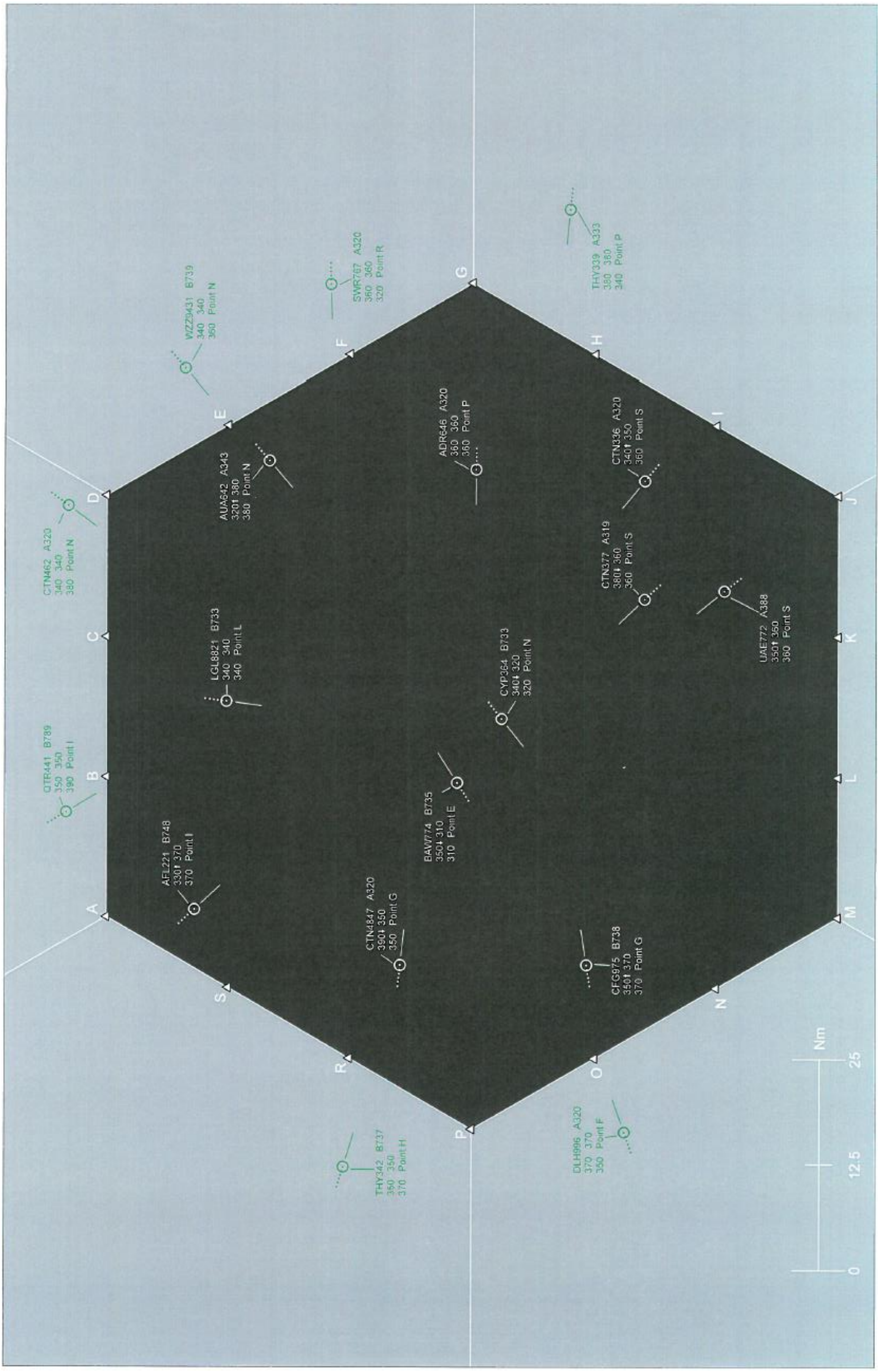
9647 K9 271 C5 334 L67 344

Traffic situation C5



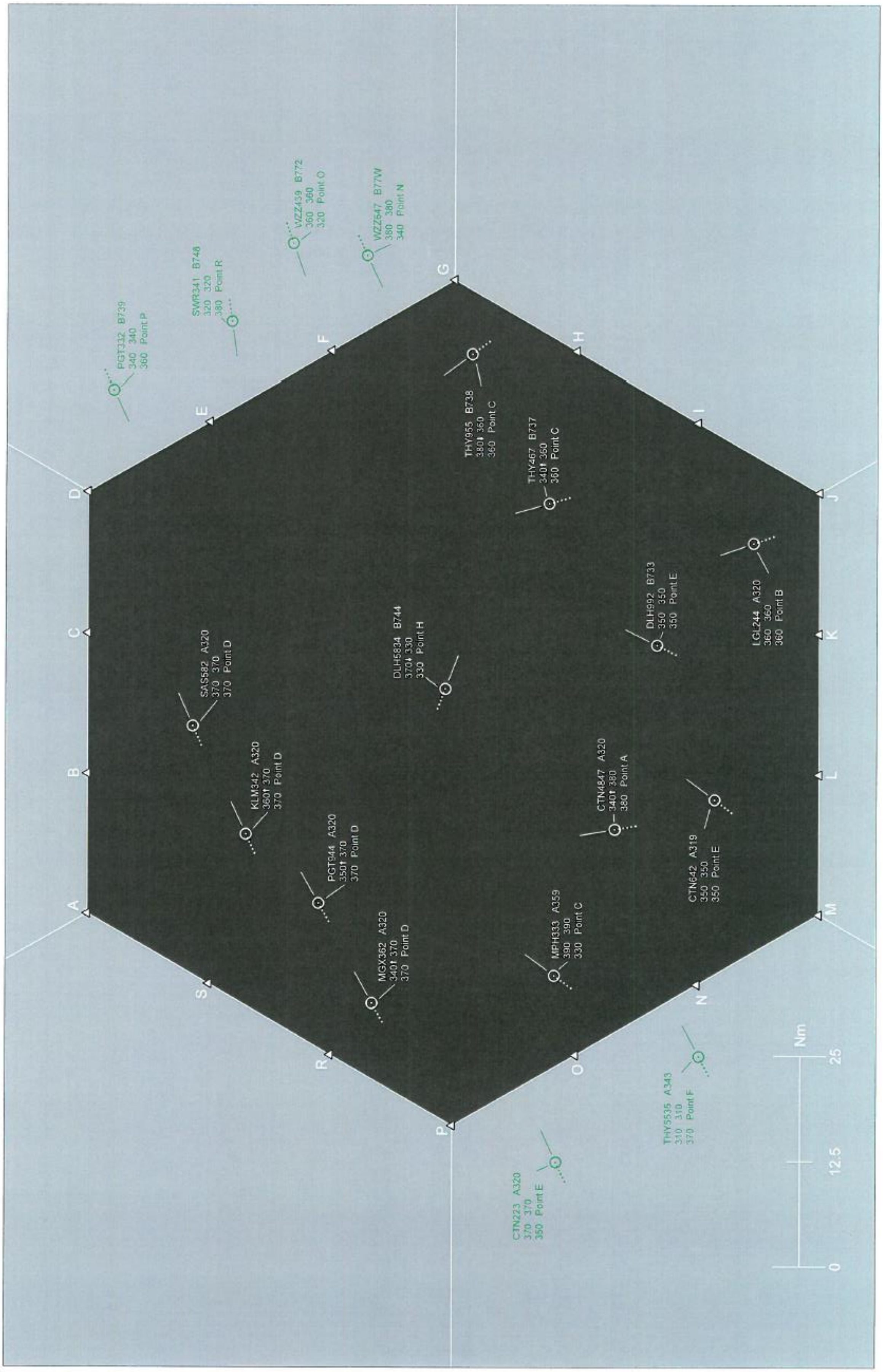
9986 F5 589 C6 401 M55 151

Traffic situation C6

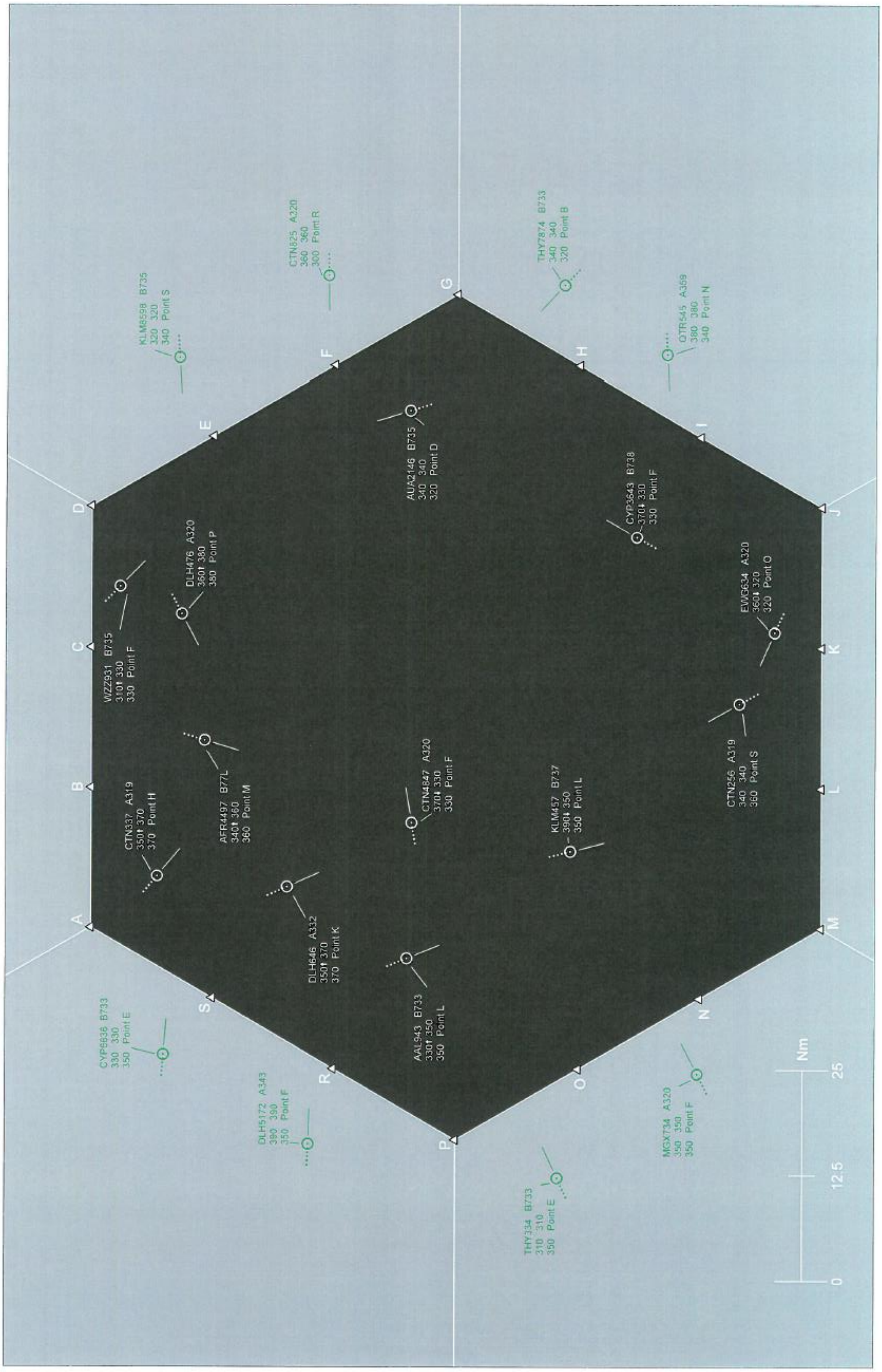


7831 D1 372 C7 971 V38 747

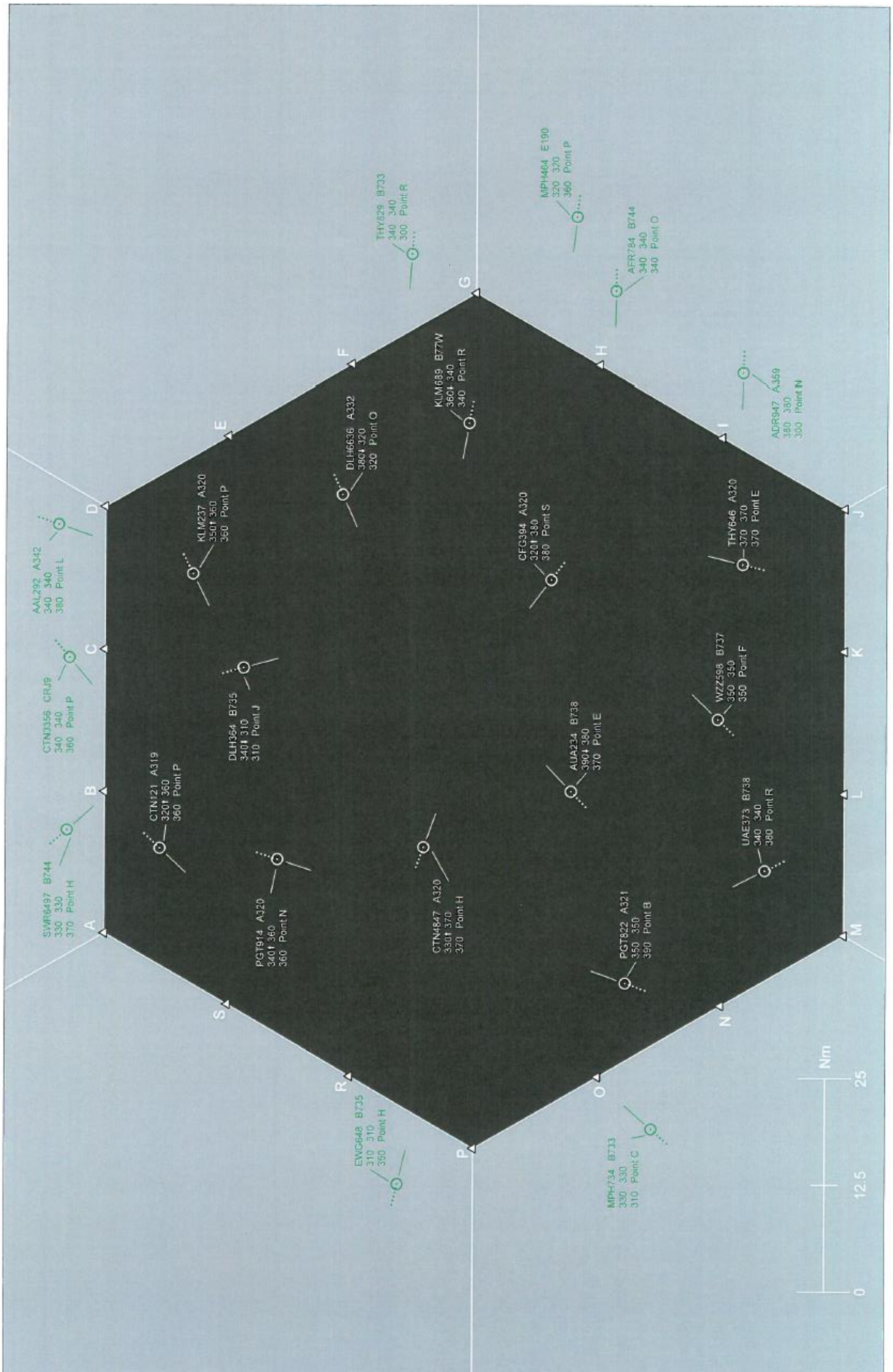
Traffic situation C7



Traffic situation C8

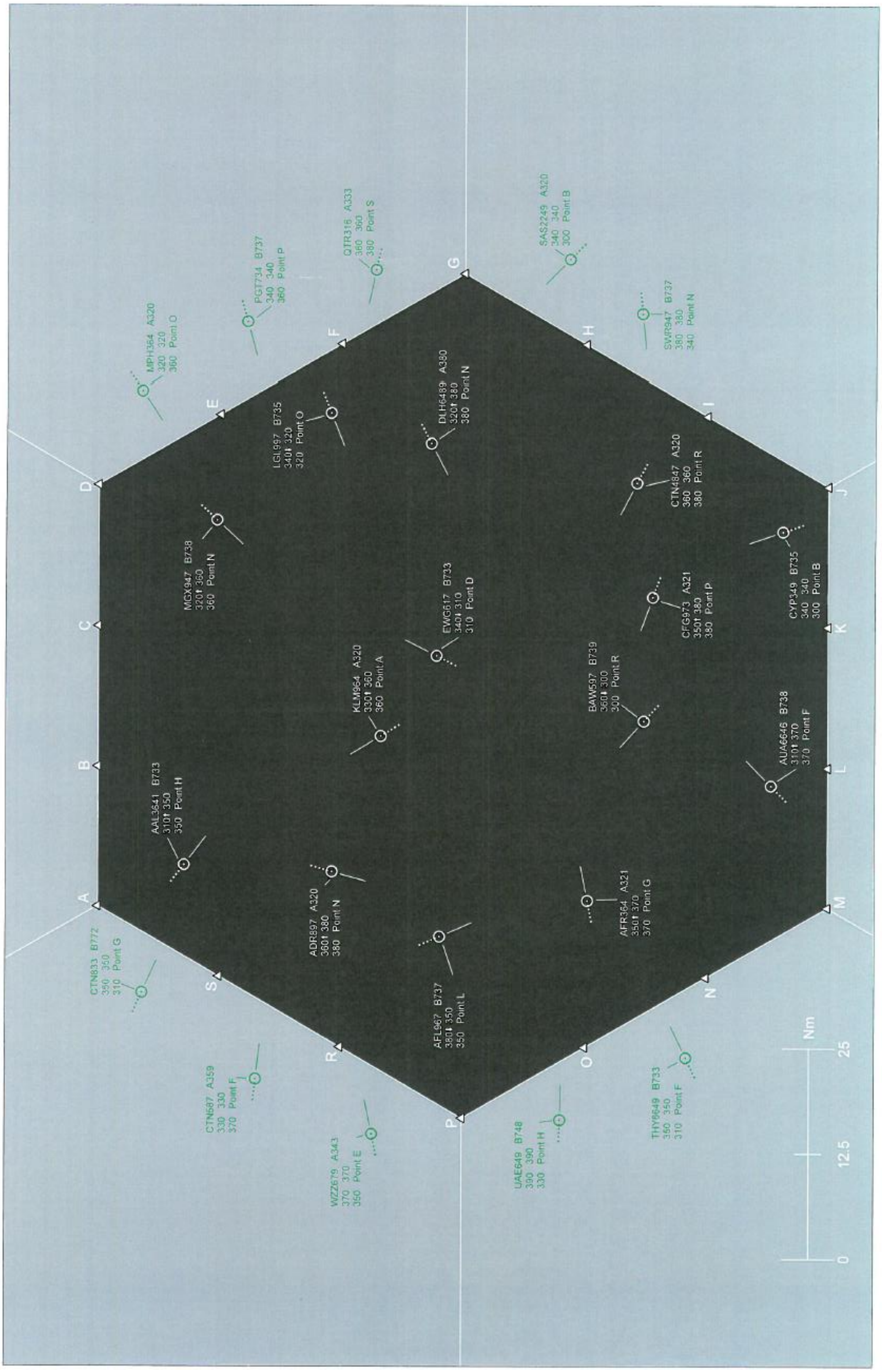


Traffic situation C9

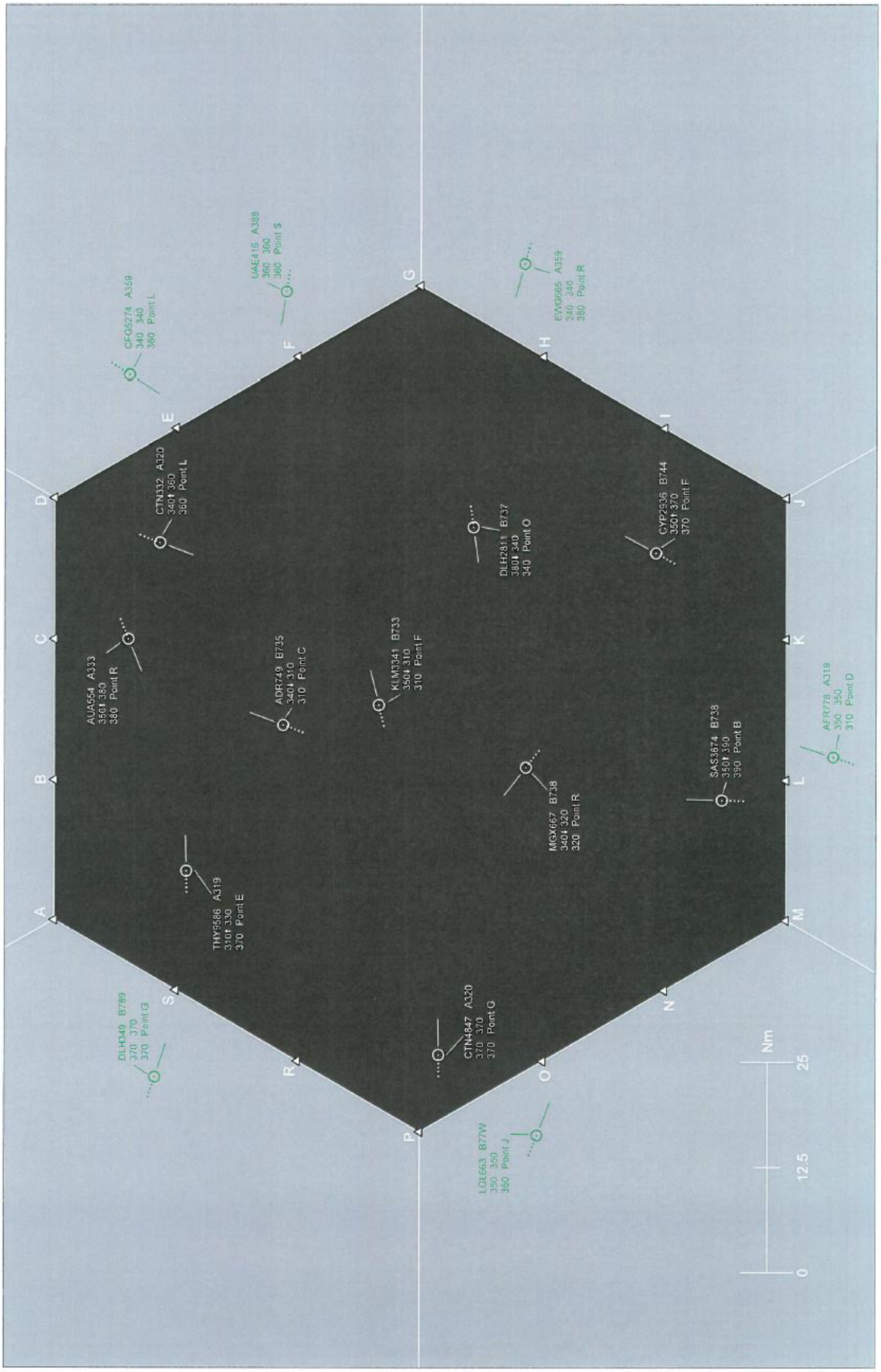


Traffic situation C10

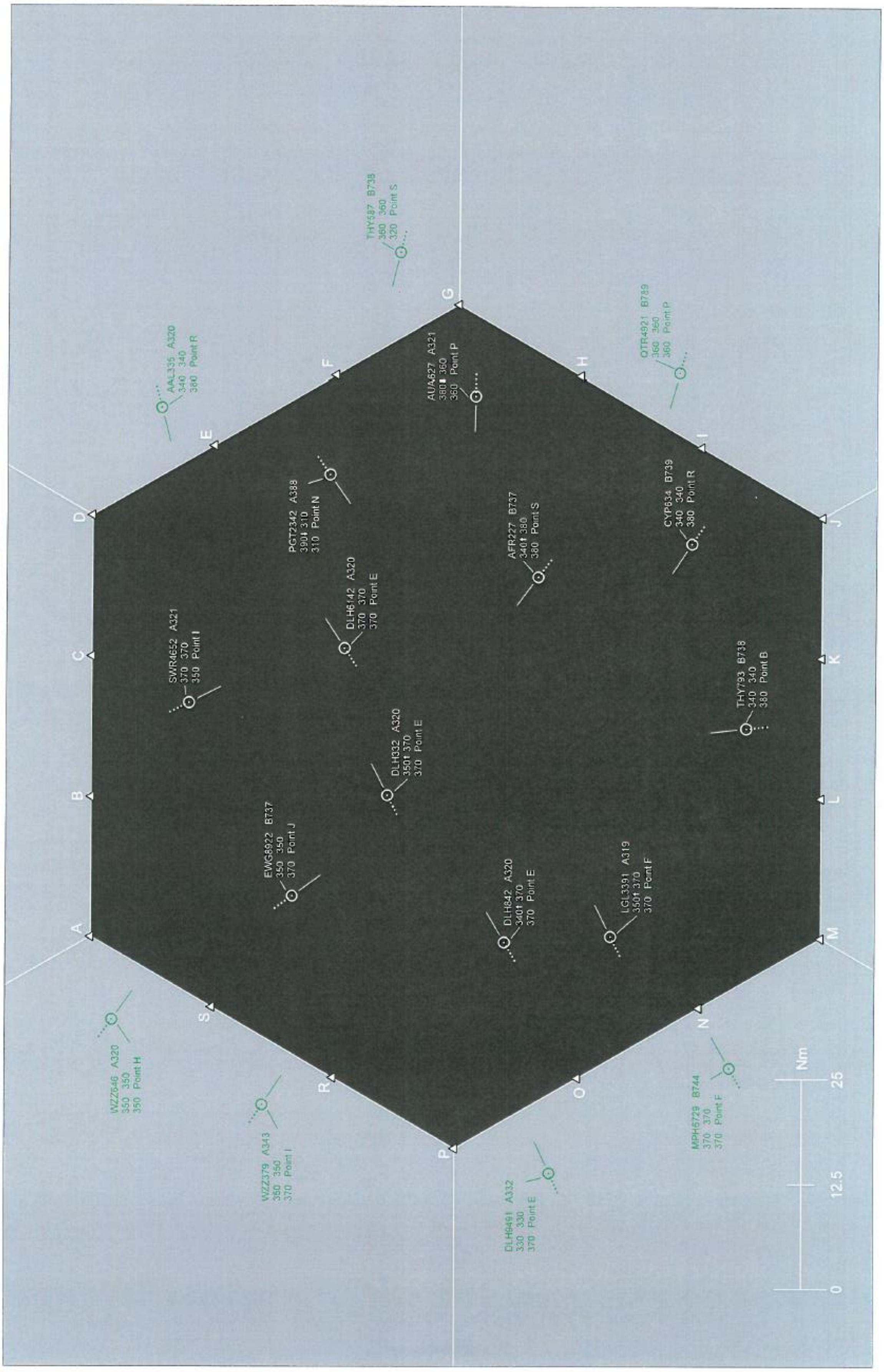
9647 W3 101 C10 311 J6 517



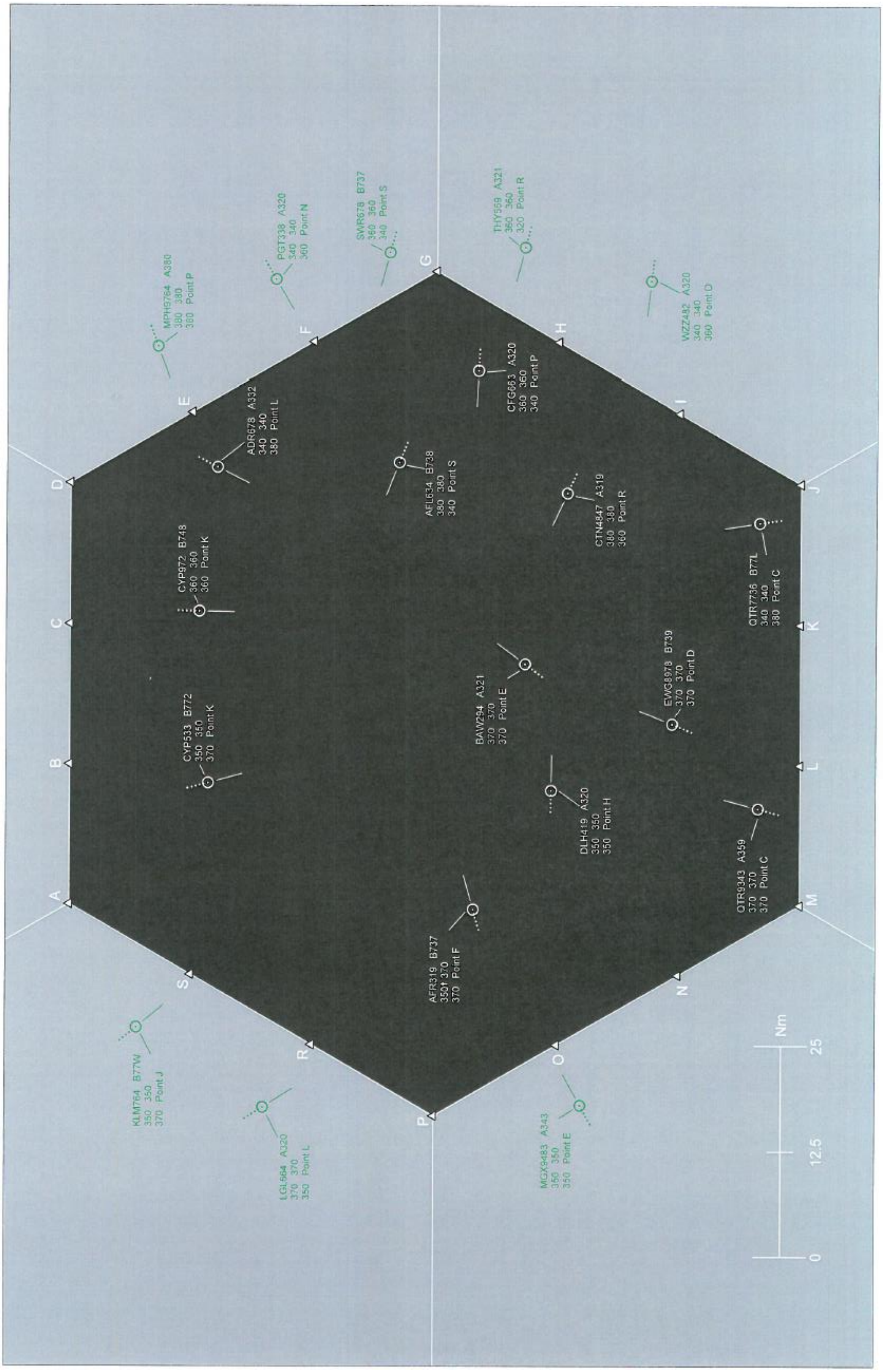
Traffic situation C11



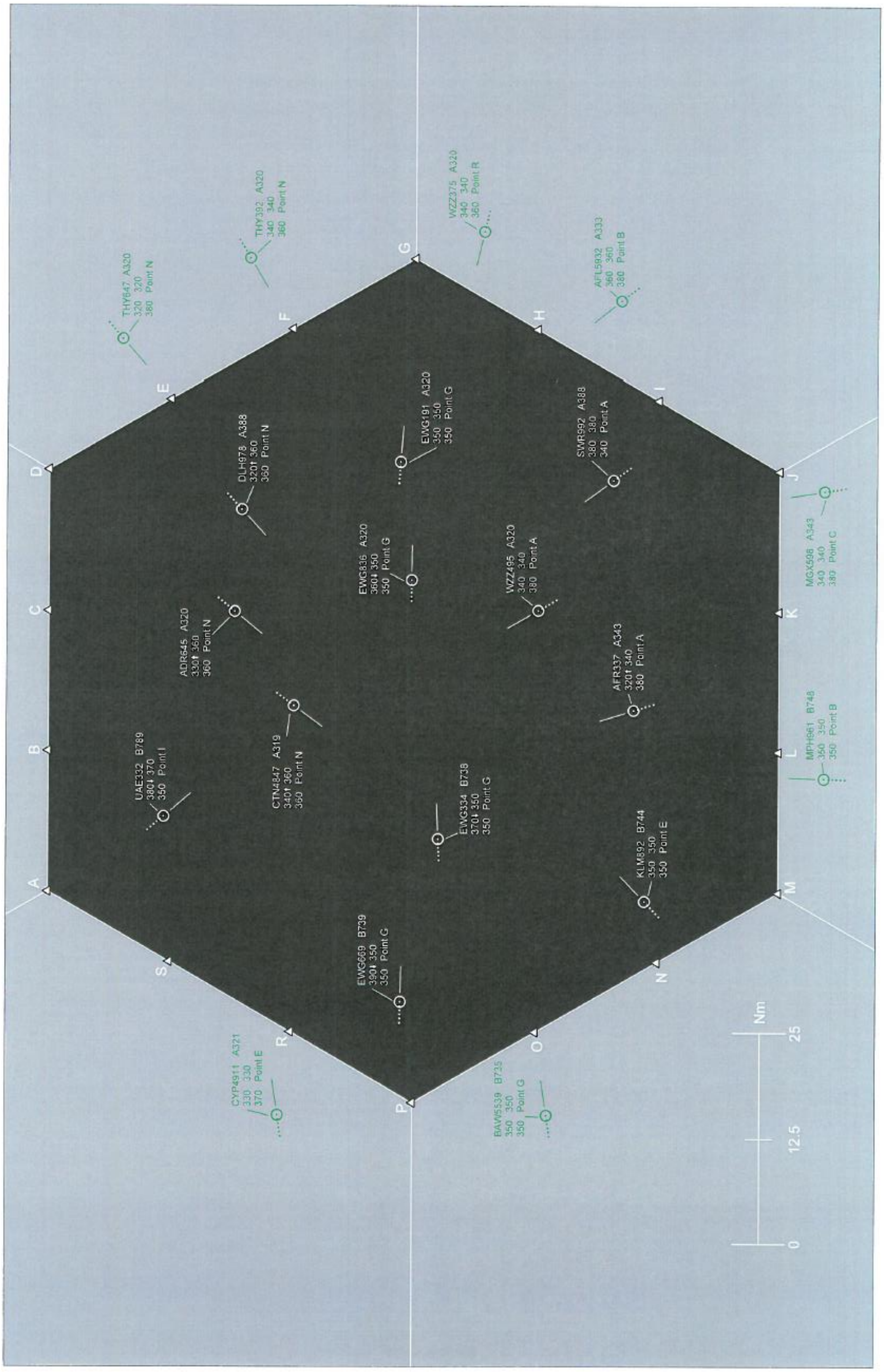
Traffic situation C12



Traffic situation C13

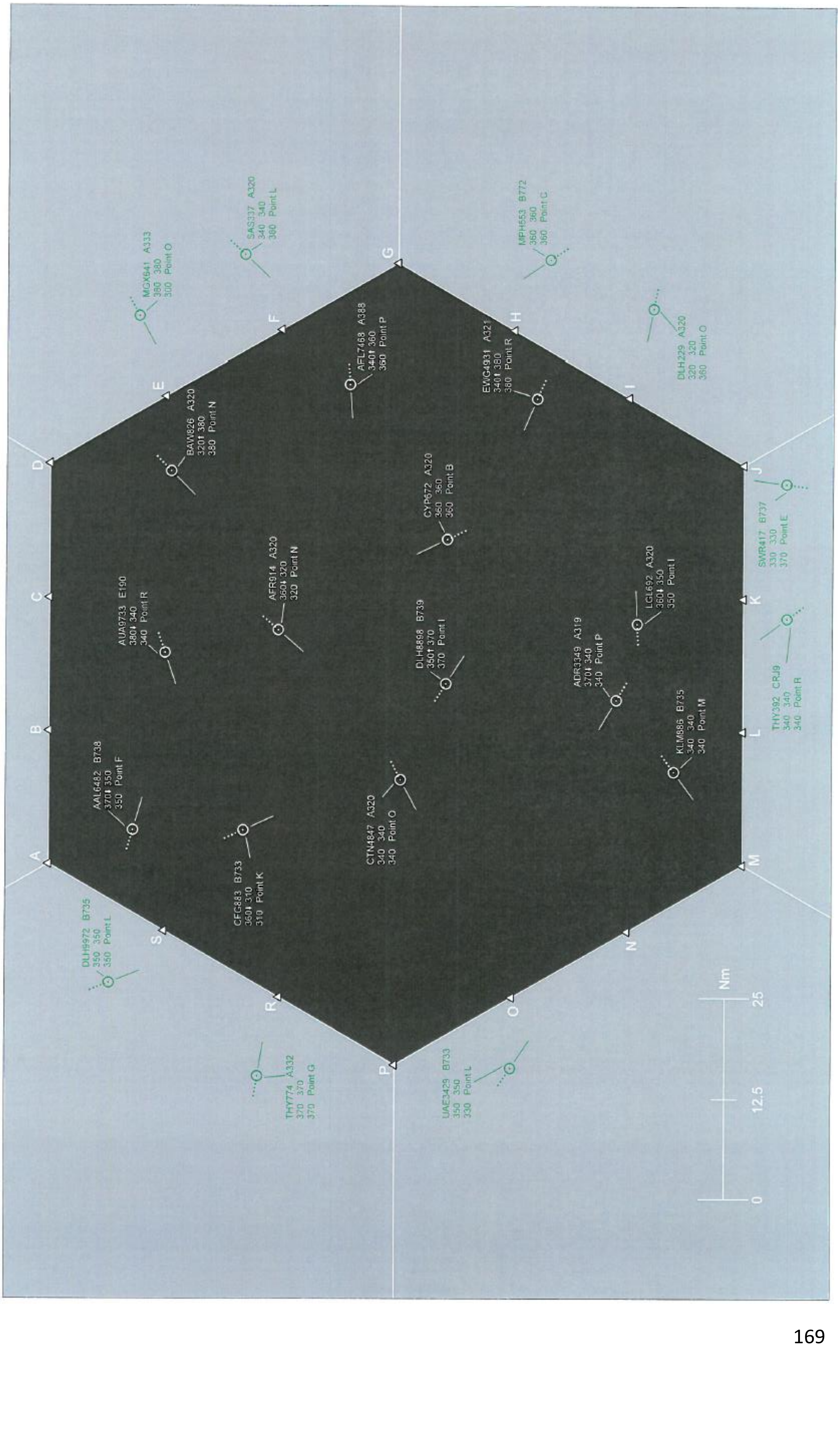


Traffic situation C14



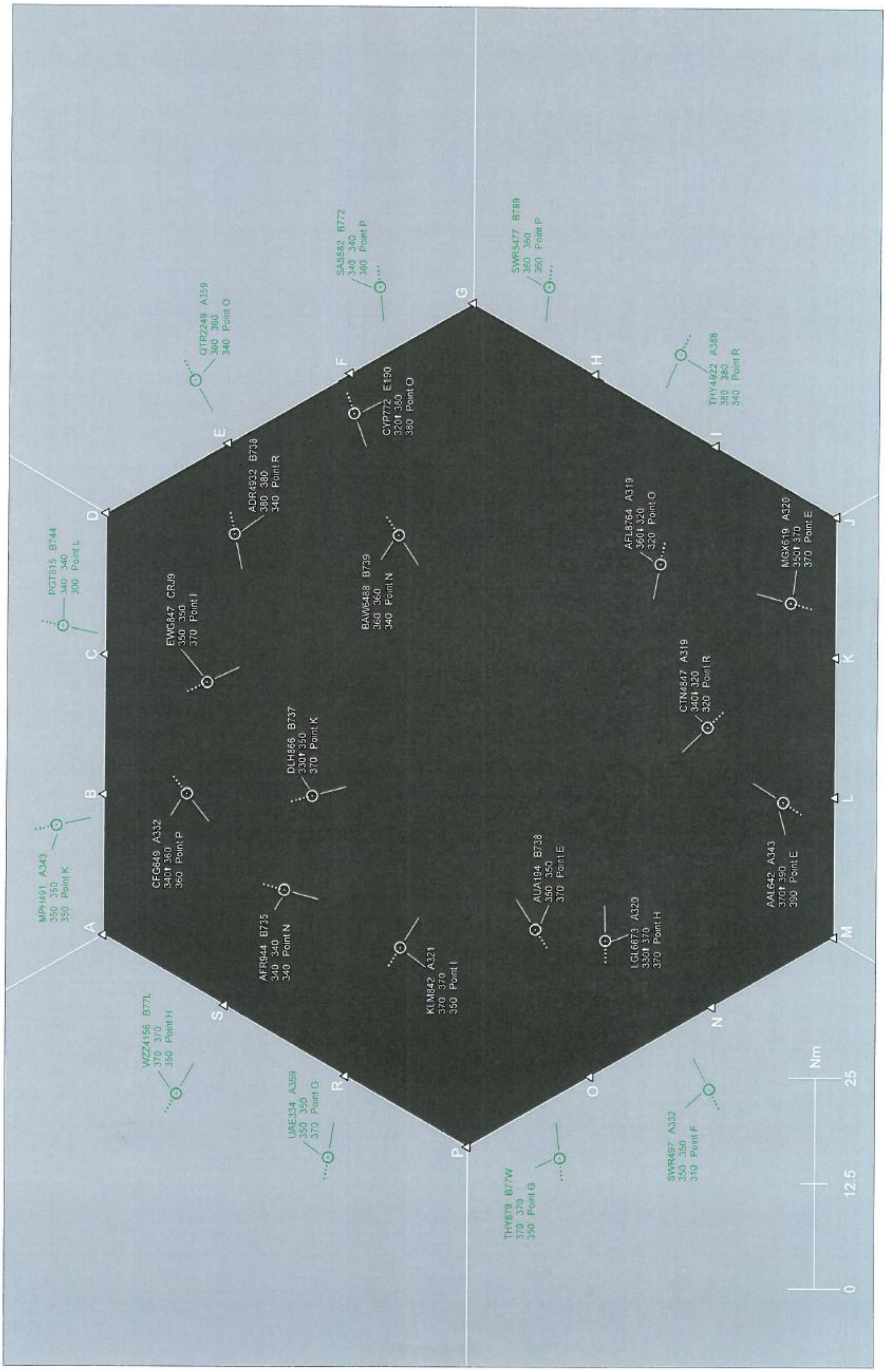
781 G7 681 C15 318 H97 824

Traffic situation C15



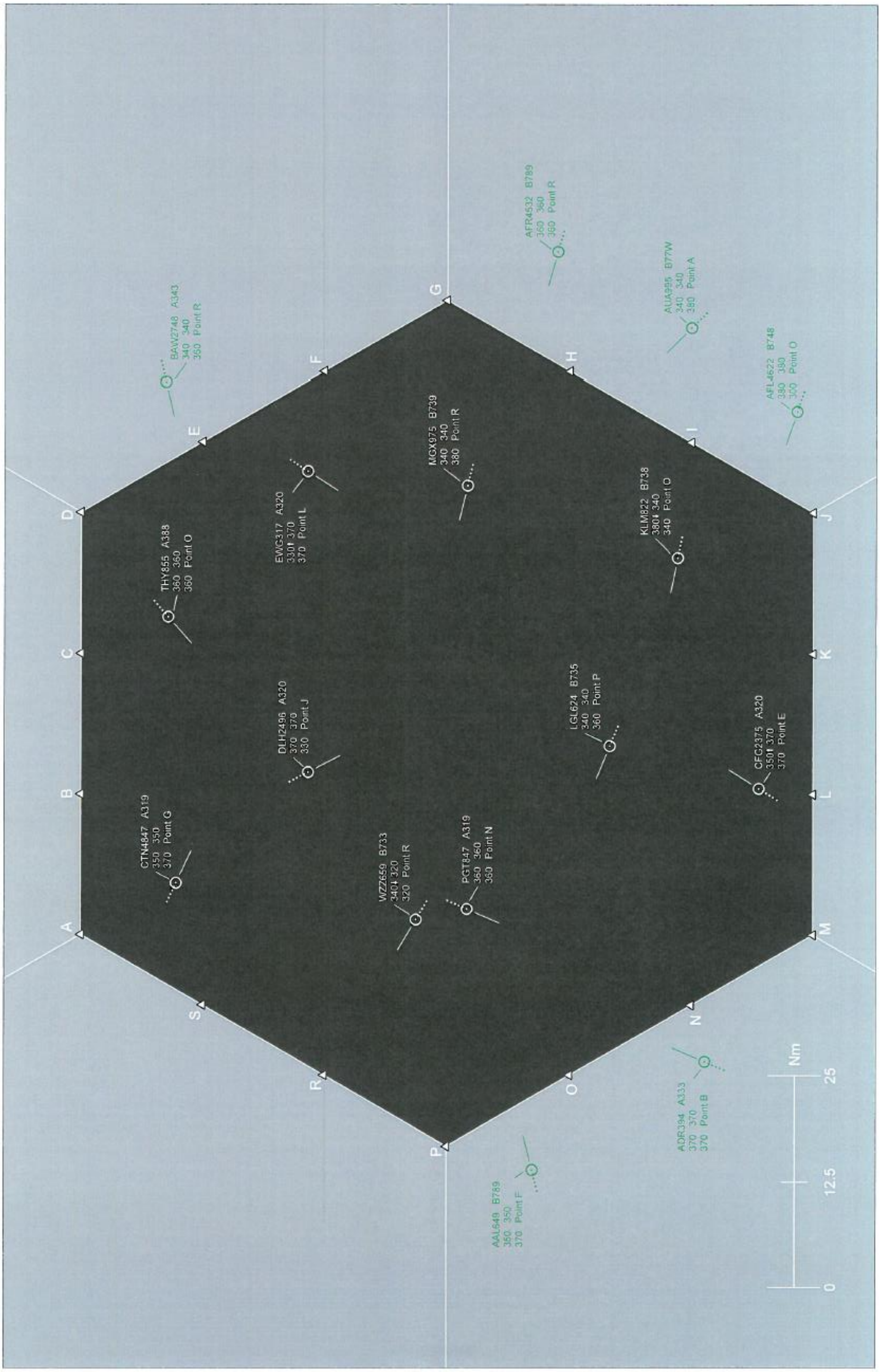
455 F5 464 C16 564 M55 545

Traffic situation C16



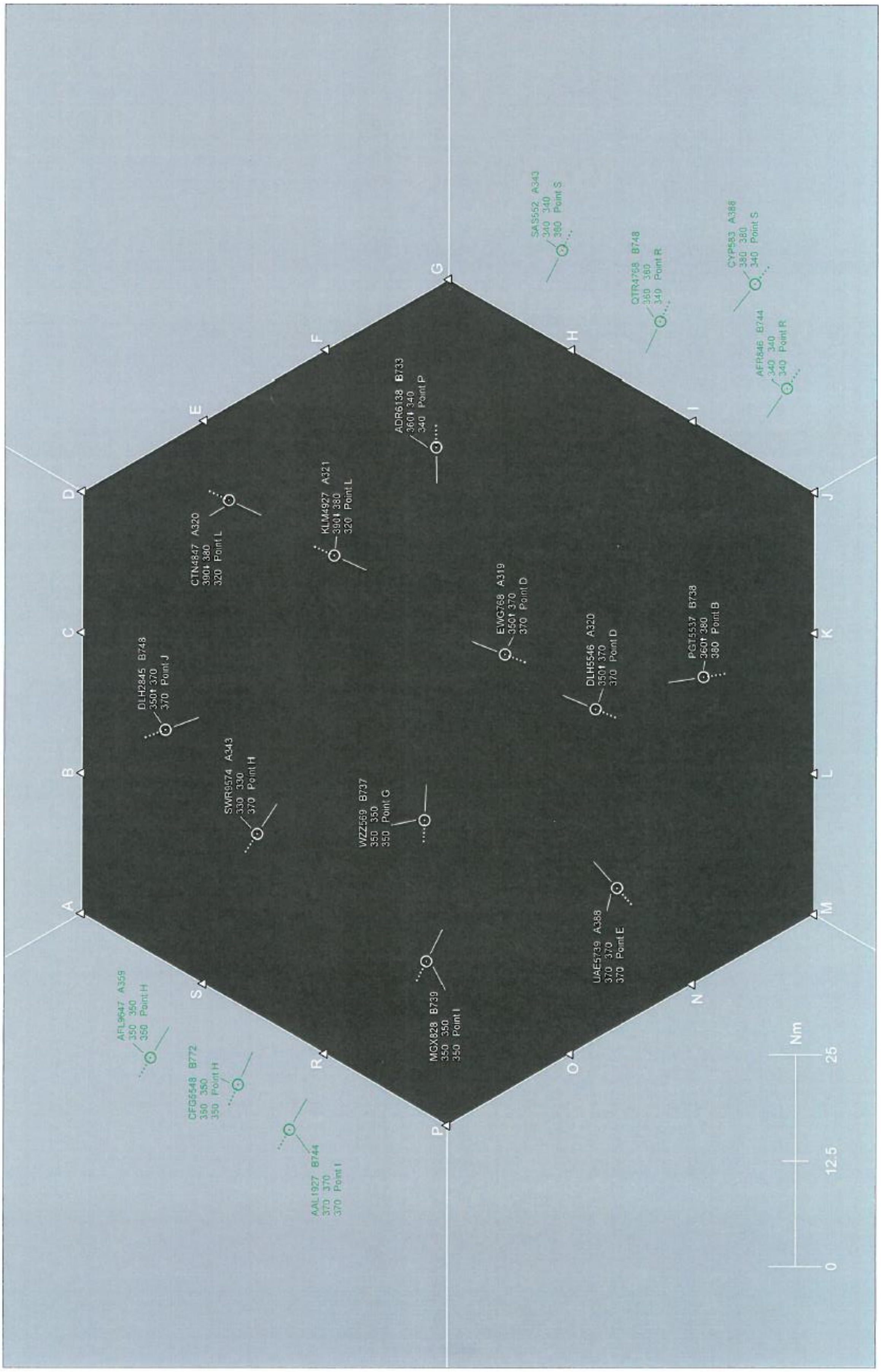
2933 X7 365 C17 970 E36 669

Traffic situation C17



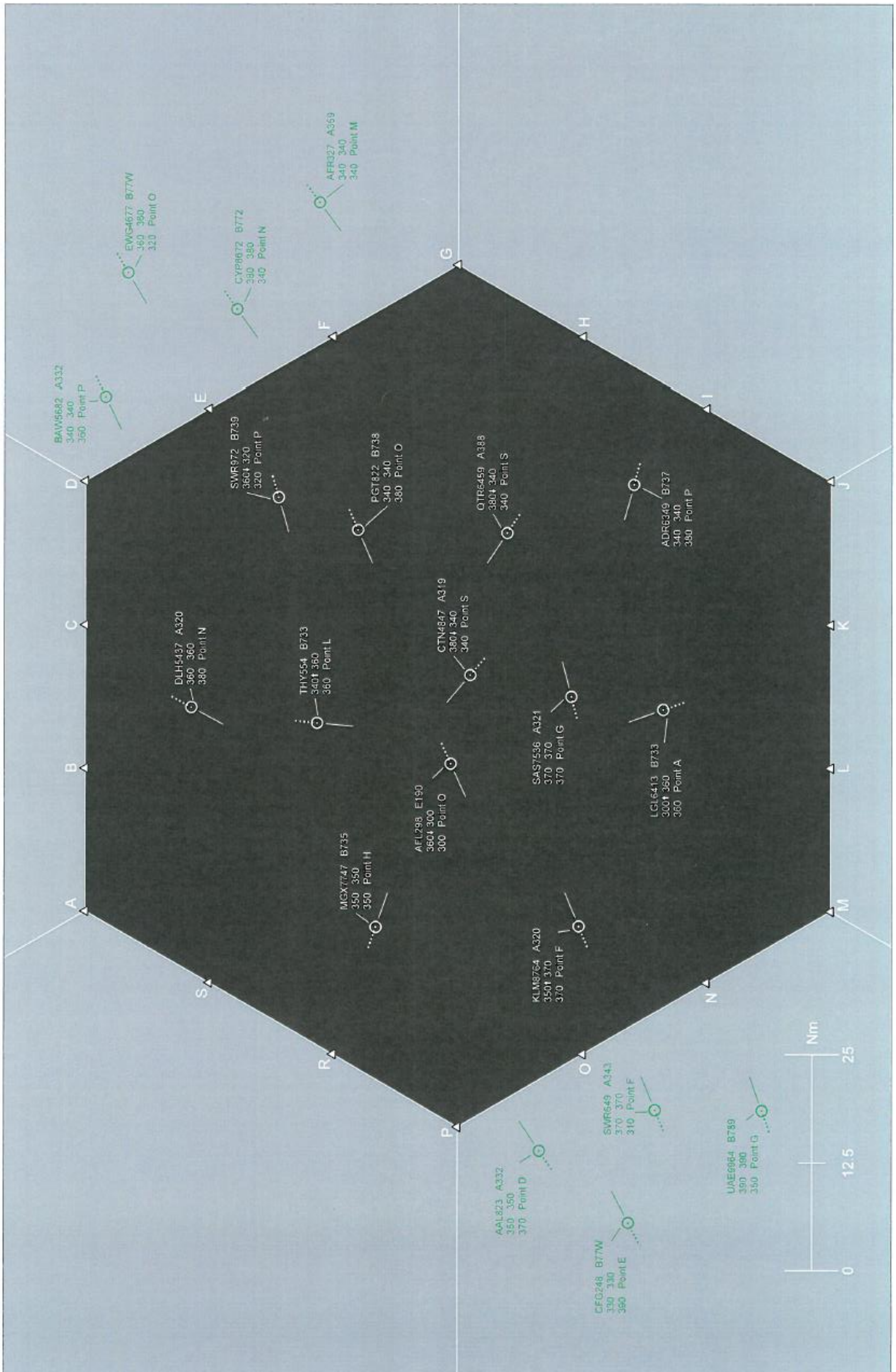
Traffic situation C18

3363 H3 823 C18 494 L64 993

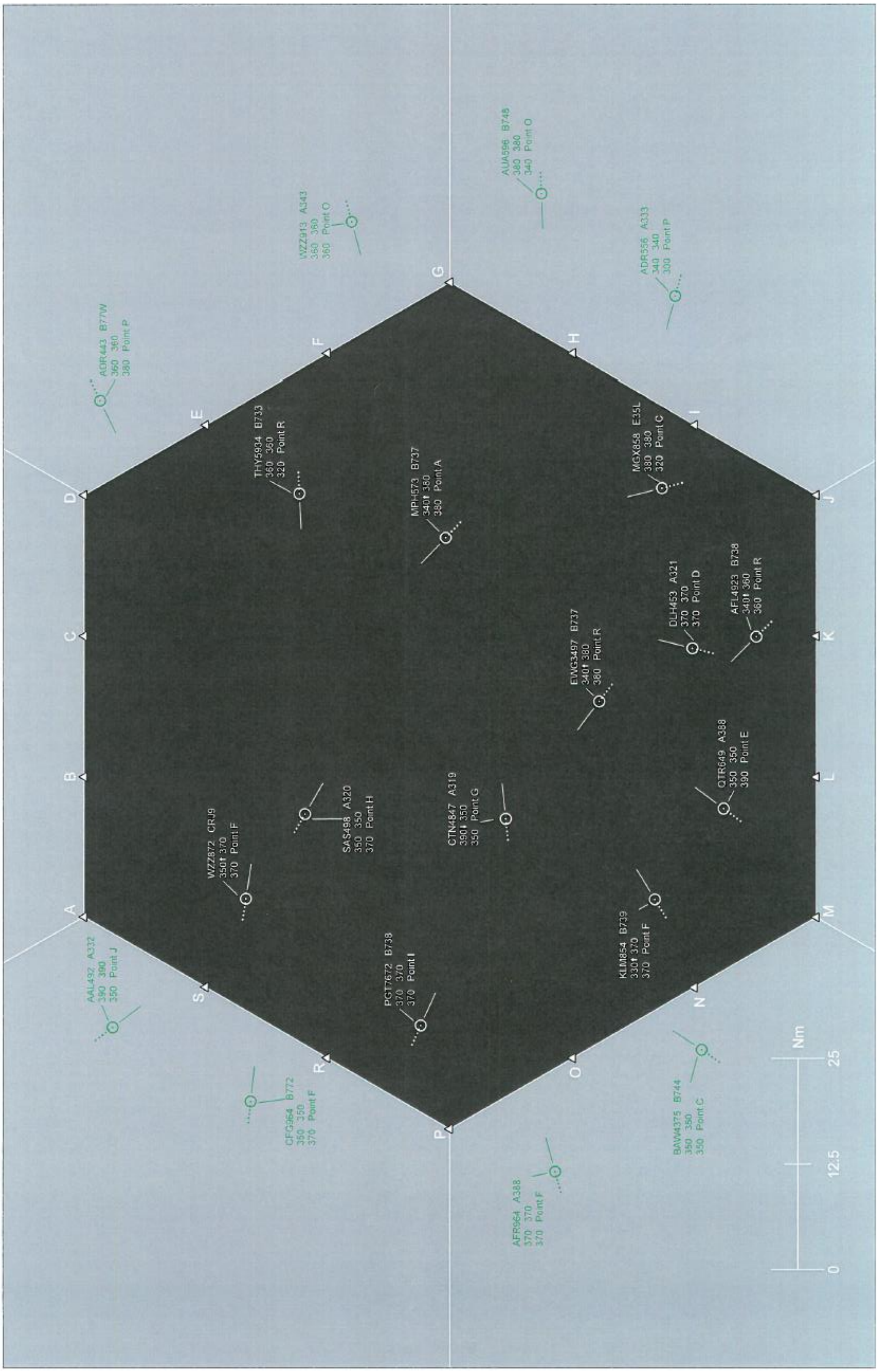


Traffic situation C19

3744 53 373 C19 686 N78 589

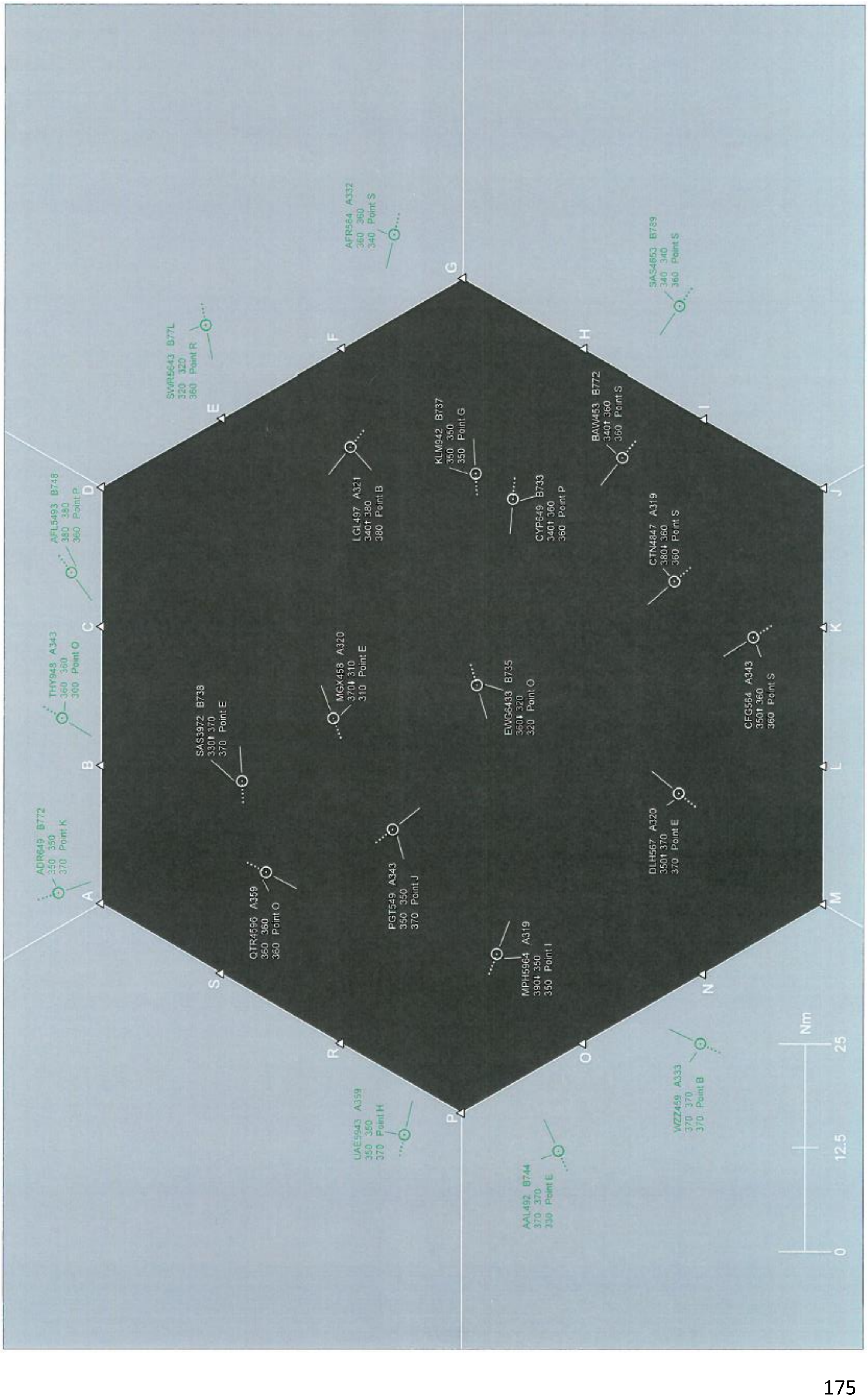


Traffic situation C20



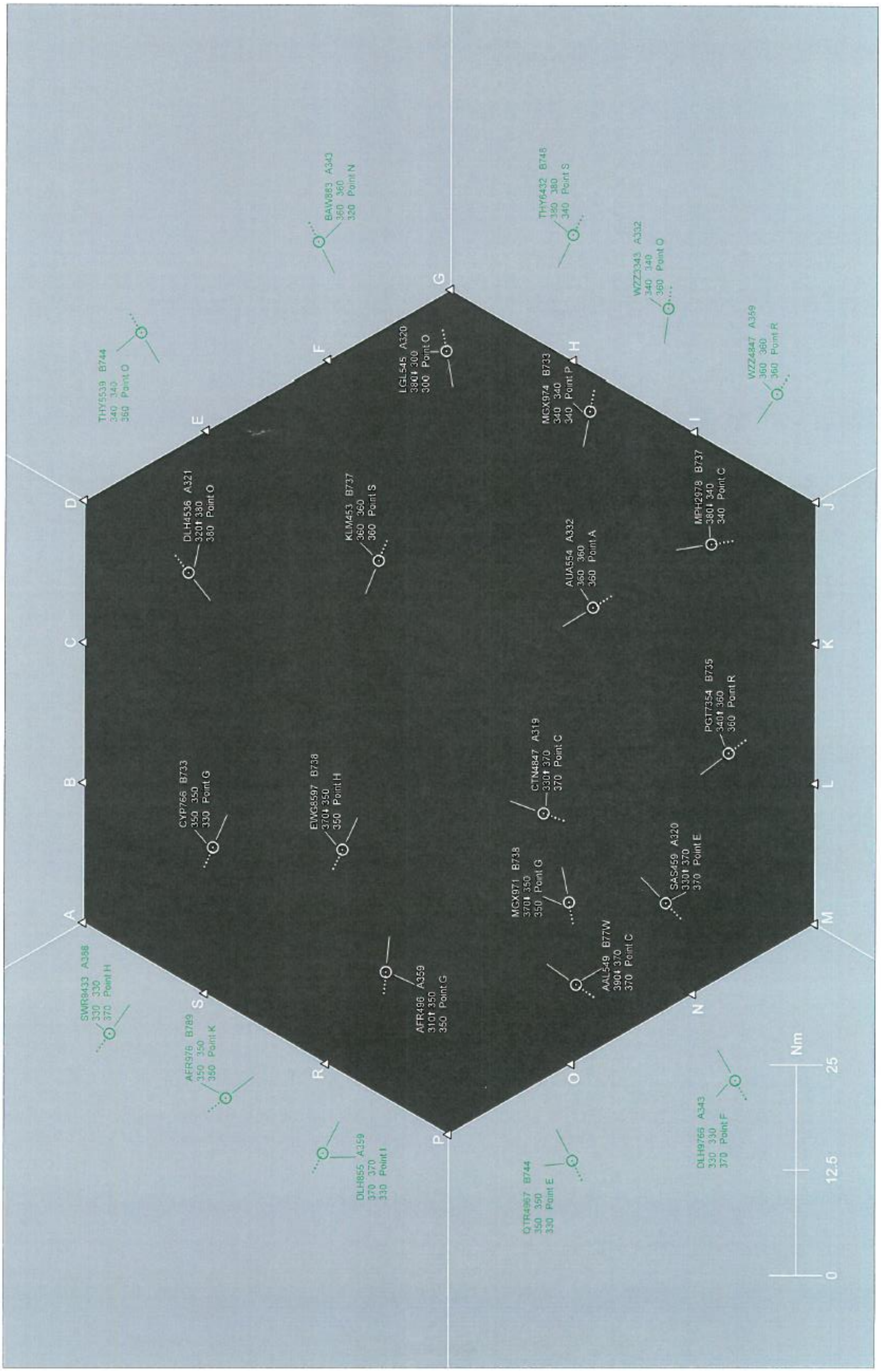
9647 K9 271 C21 334 L67 344

Traffic situation C21

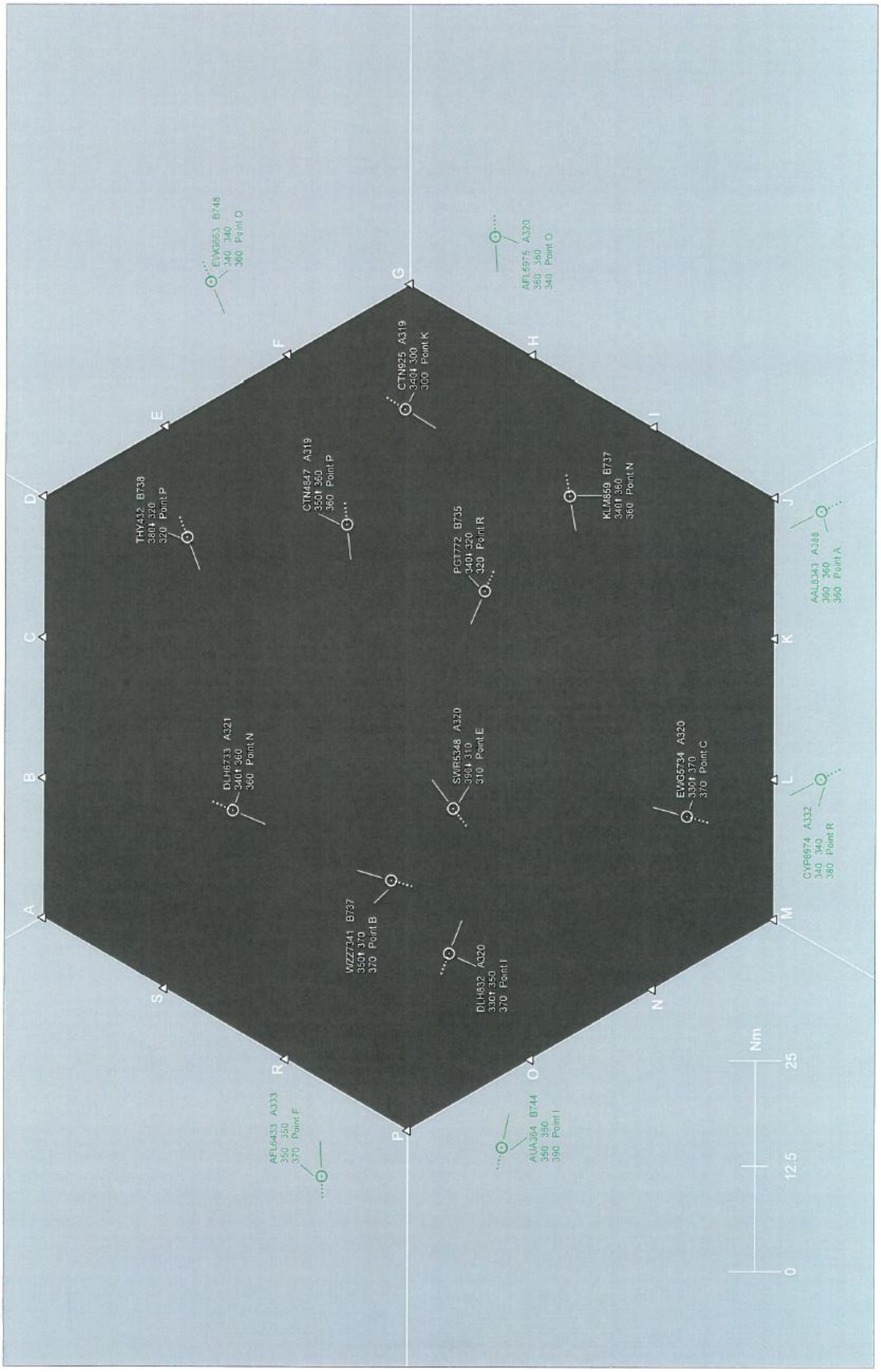


9986 F5 589 C22 401 M55 151

Traffic situation C22

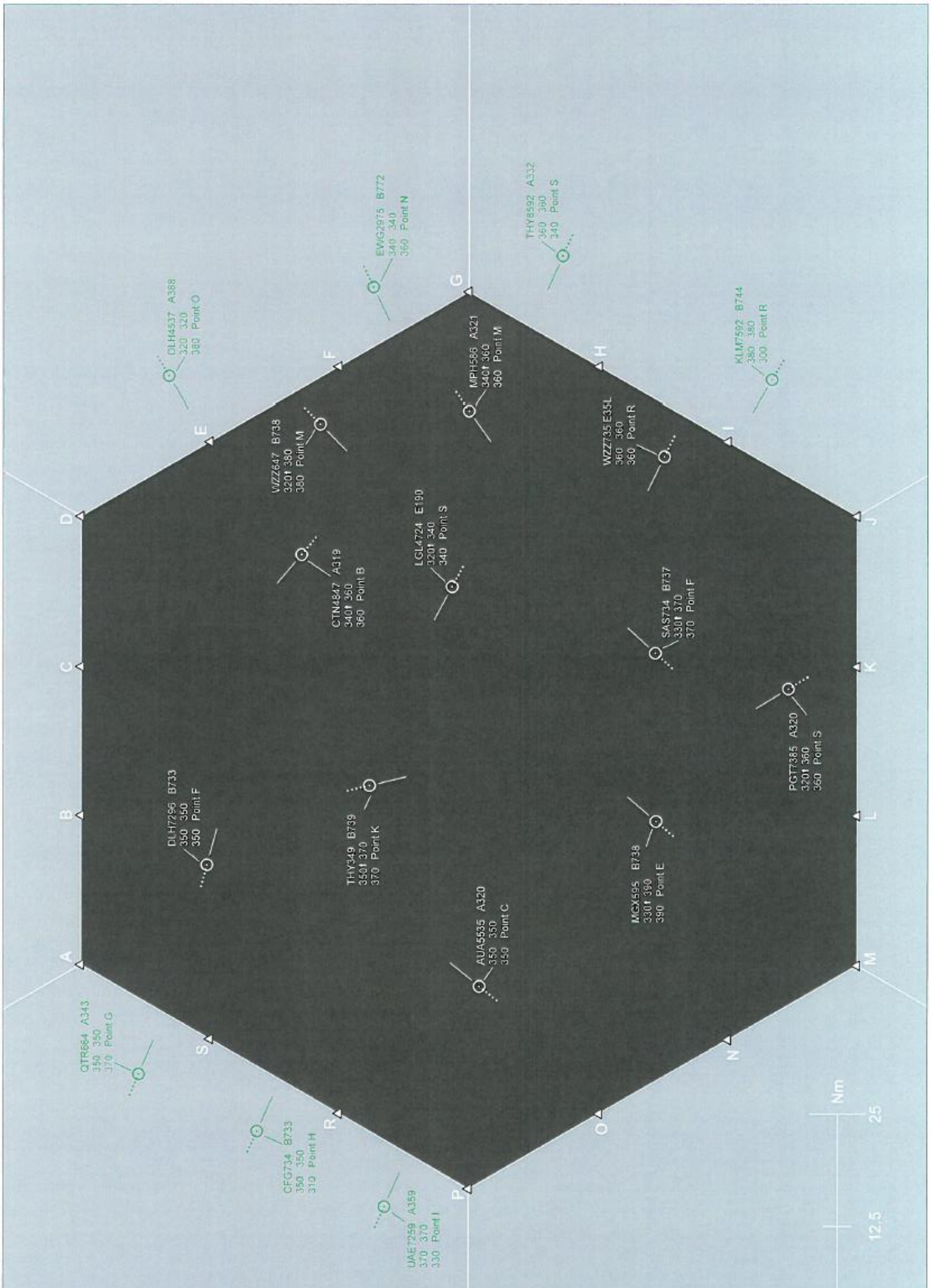


Traffic situation C23

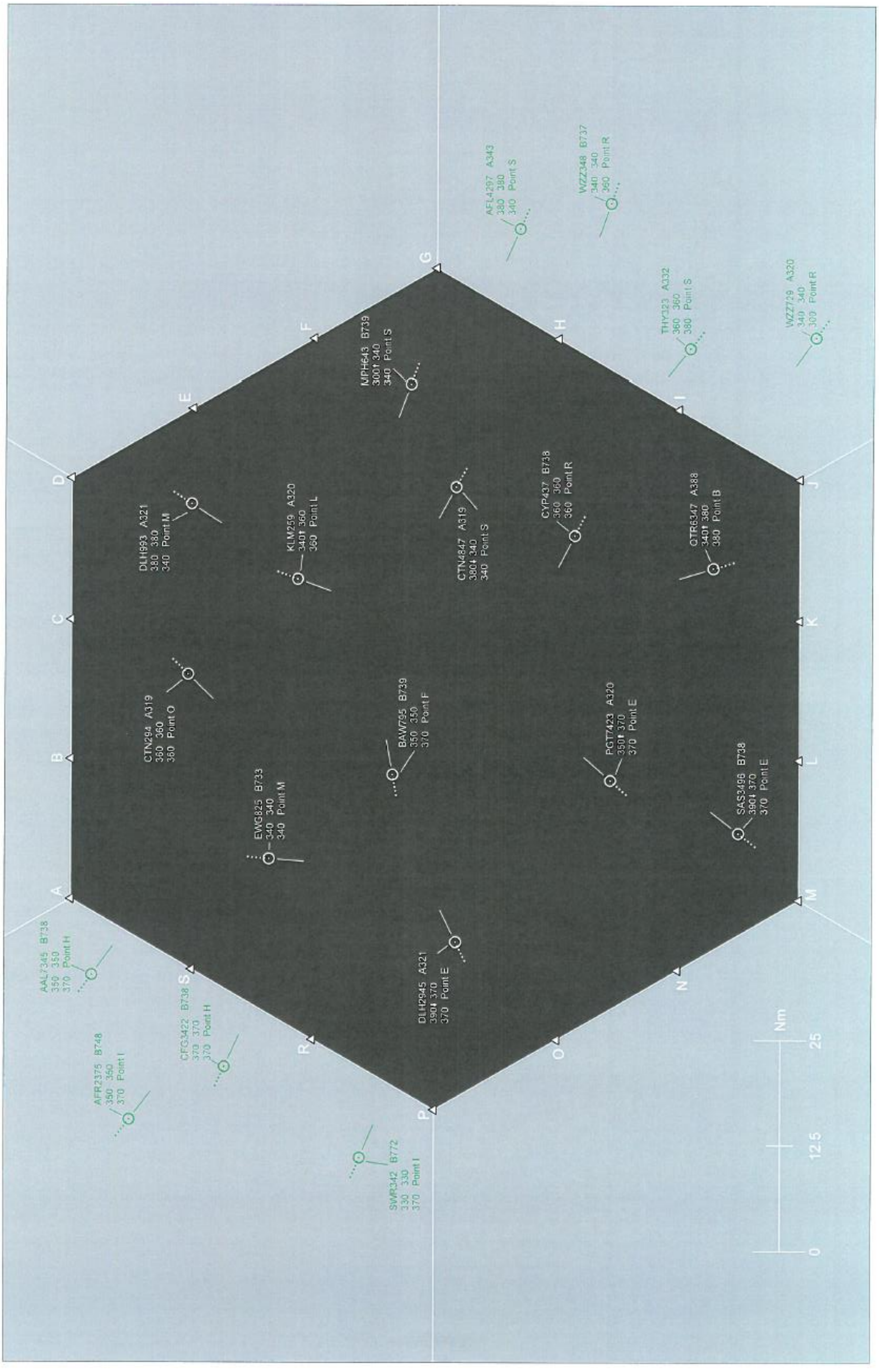


Traffic situation C24

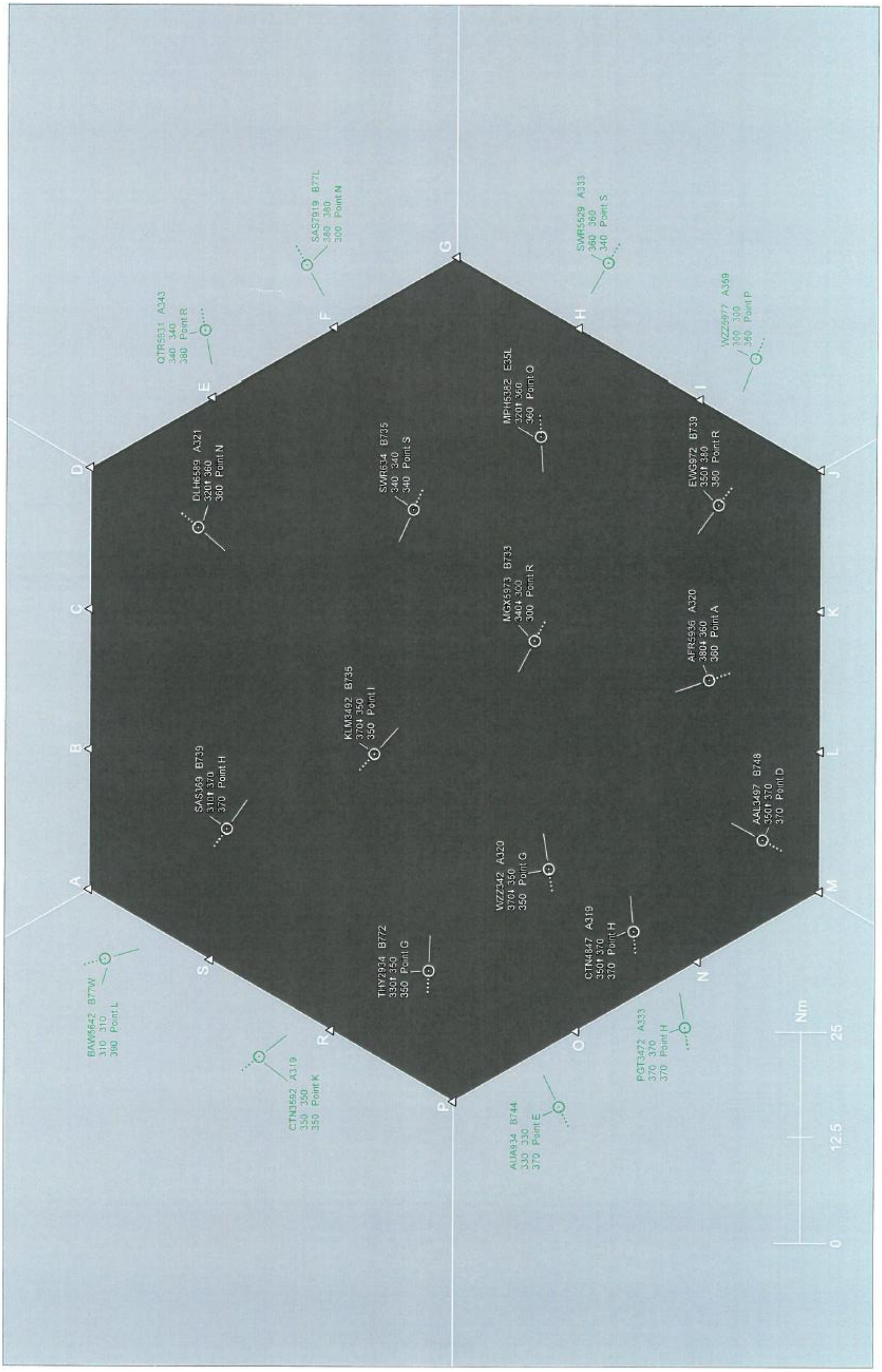
678 V8 656 C24 373 P26 364



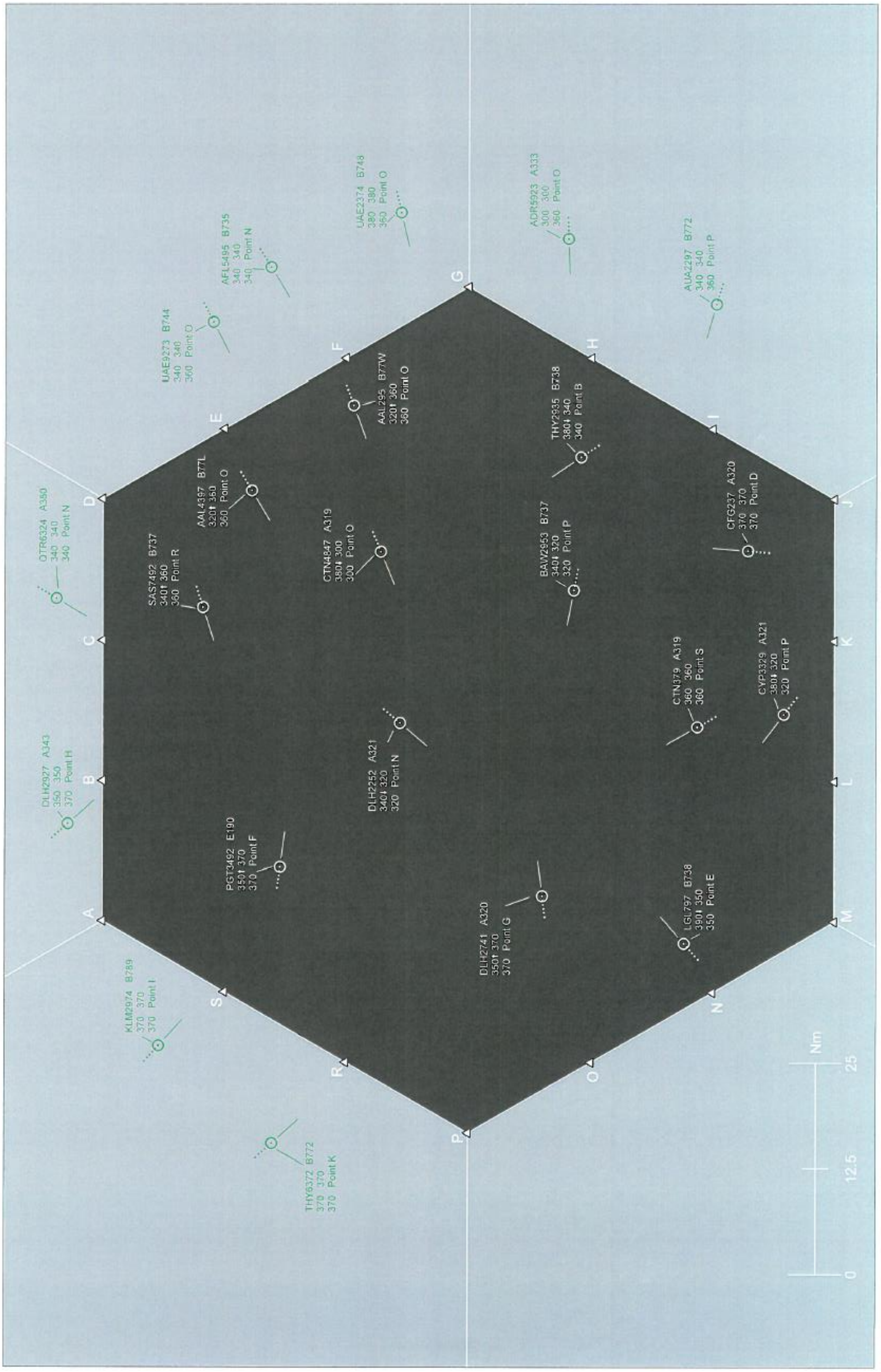
Traffic situation C25



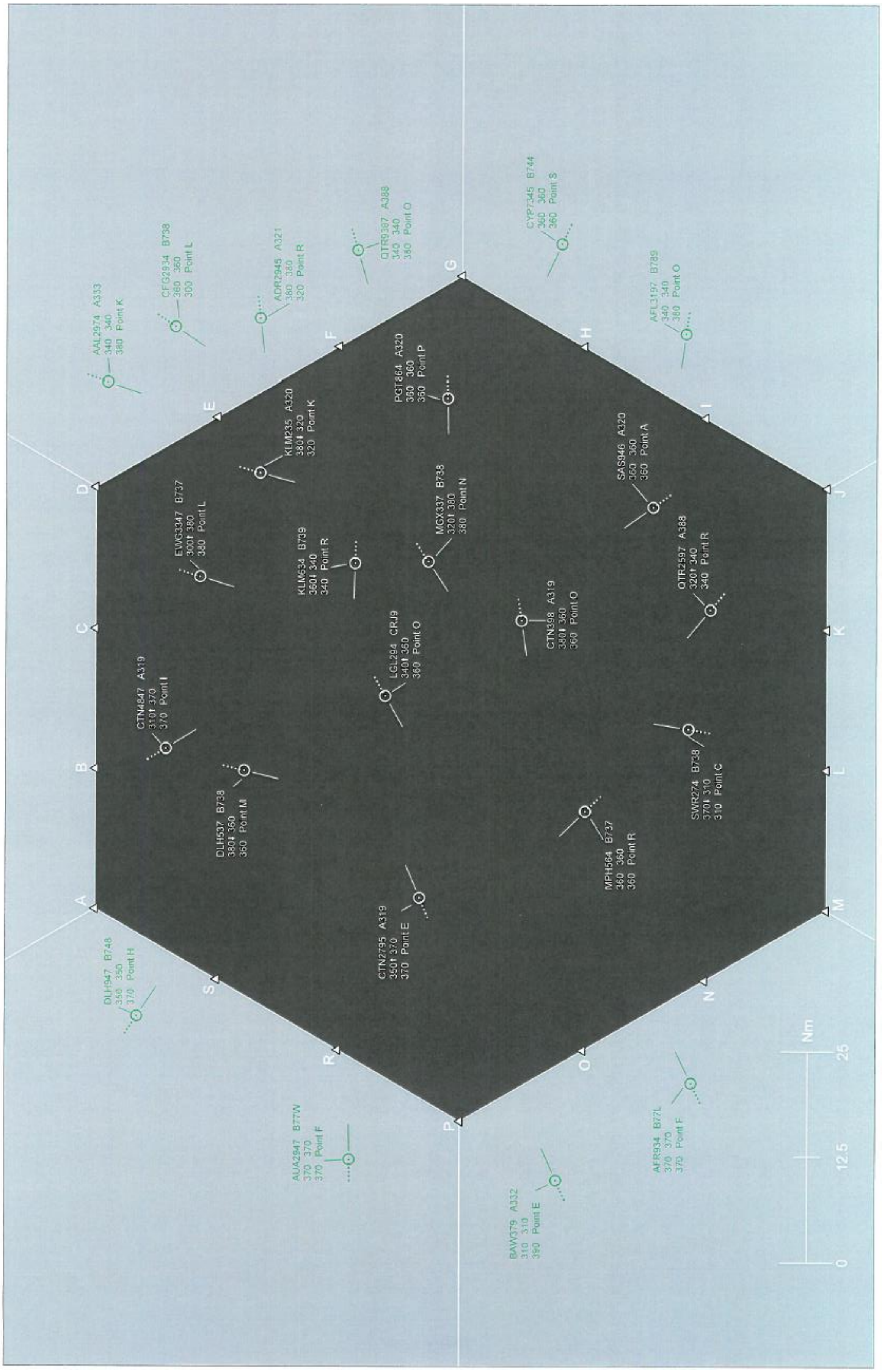
Traffic situation C26



Traffic situation C27

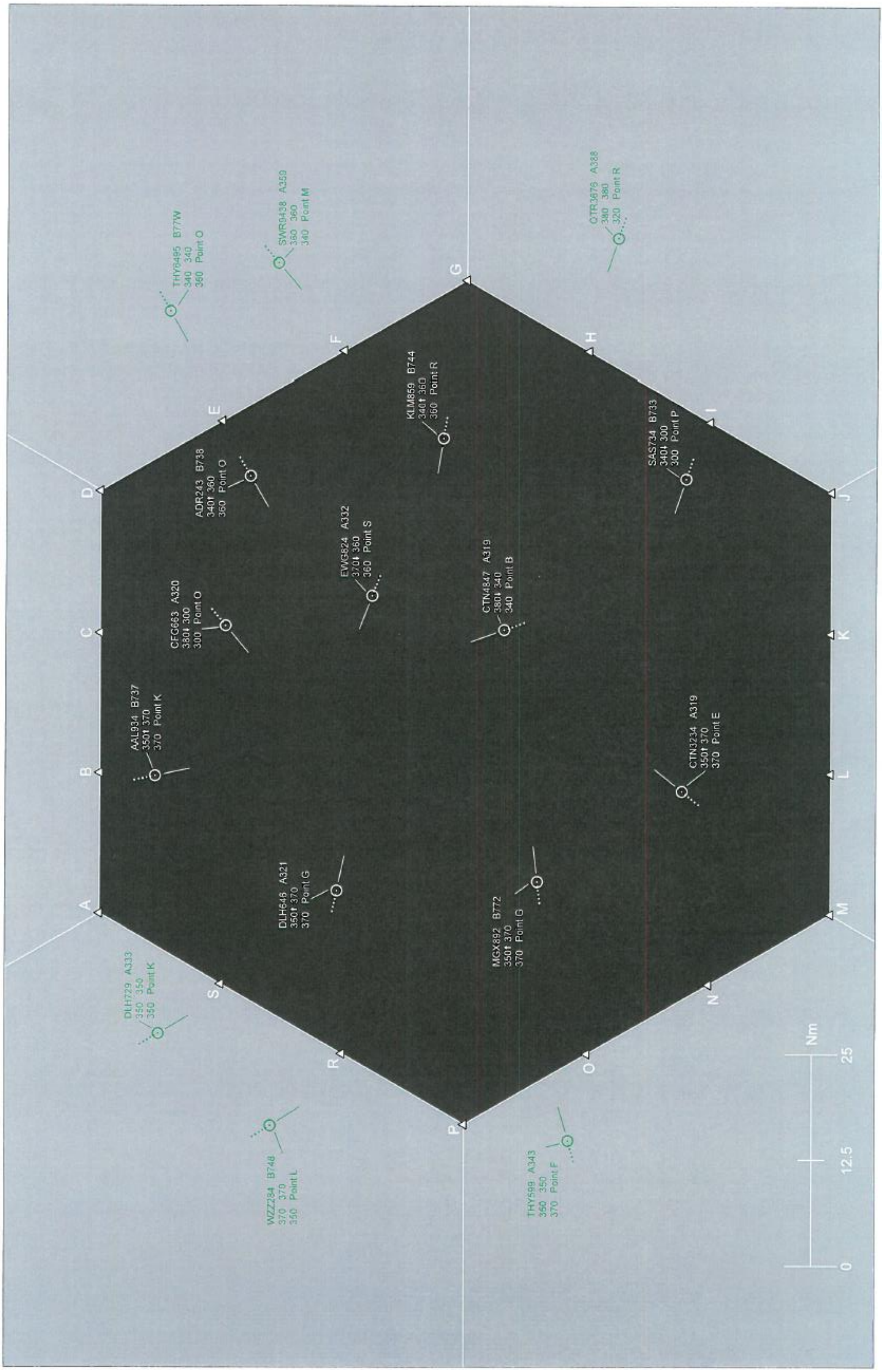


Traffic situation C28

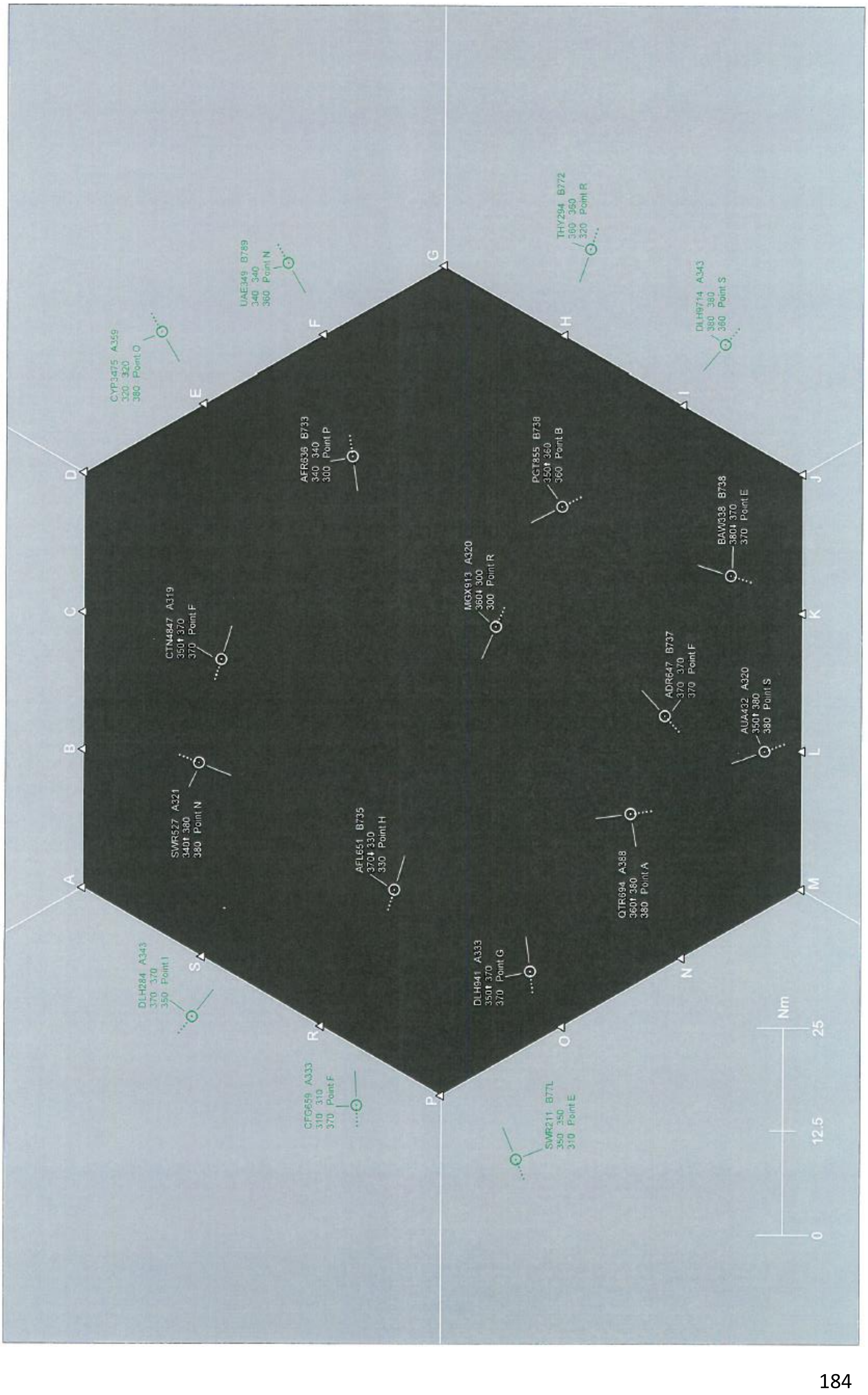


Traffic situation C29

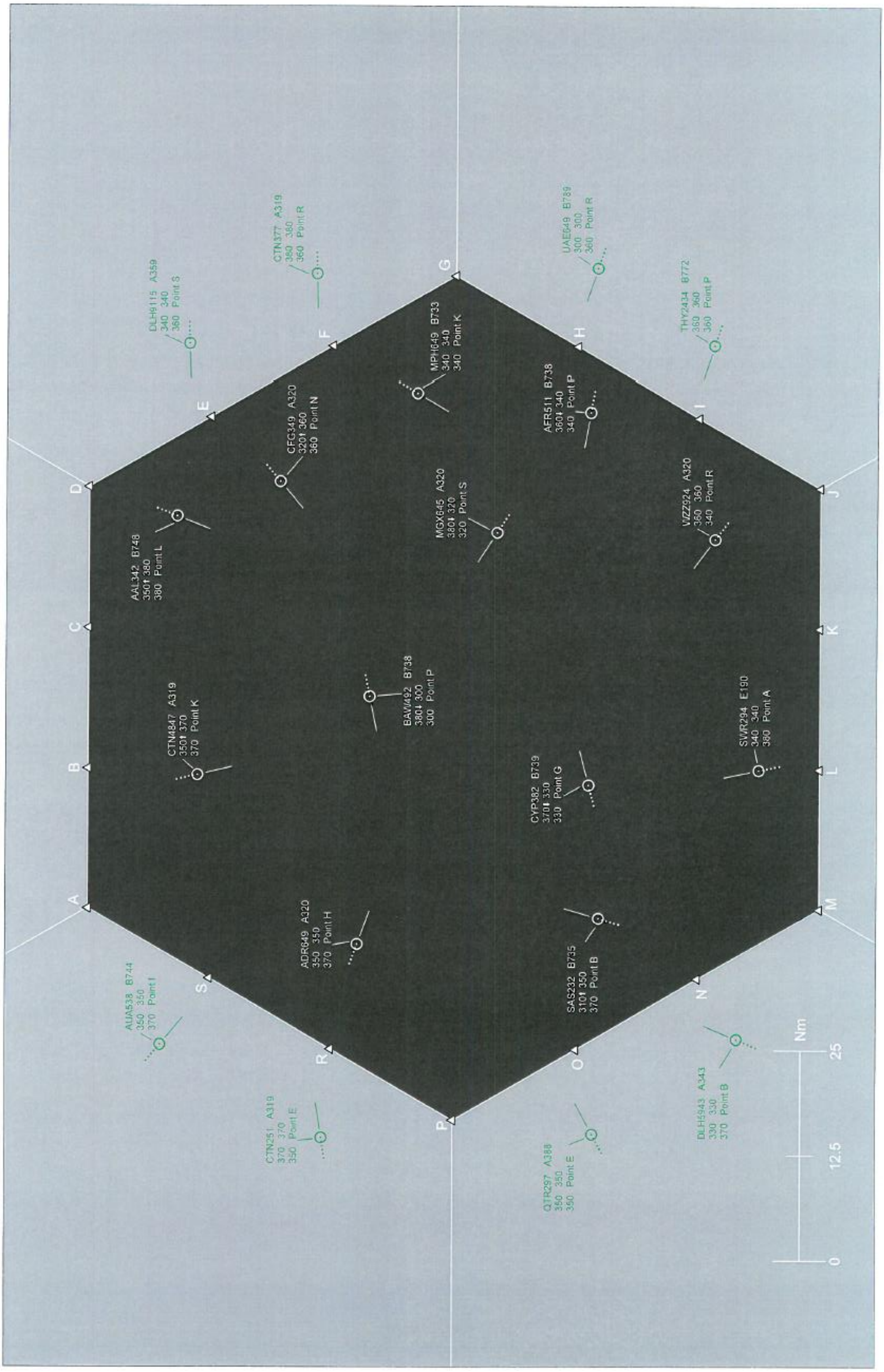
2933 X7 365 C29 970 E36 669



Traffic situation C30

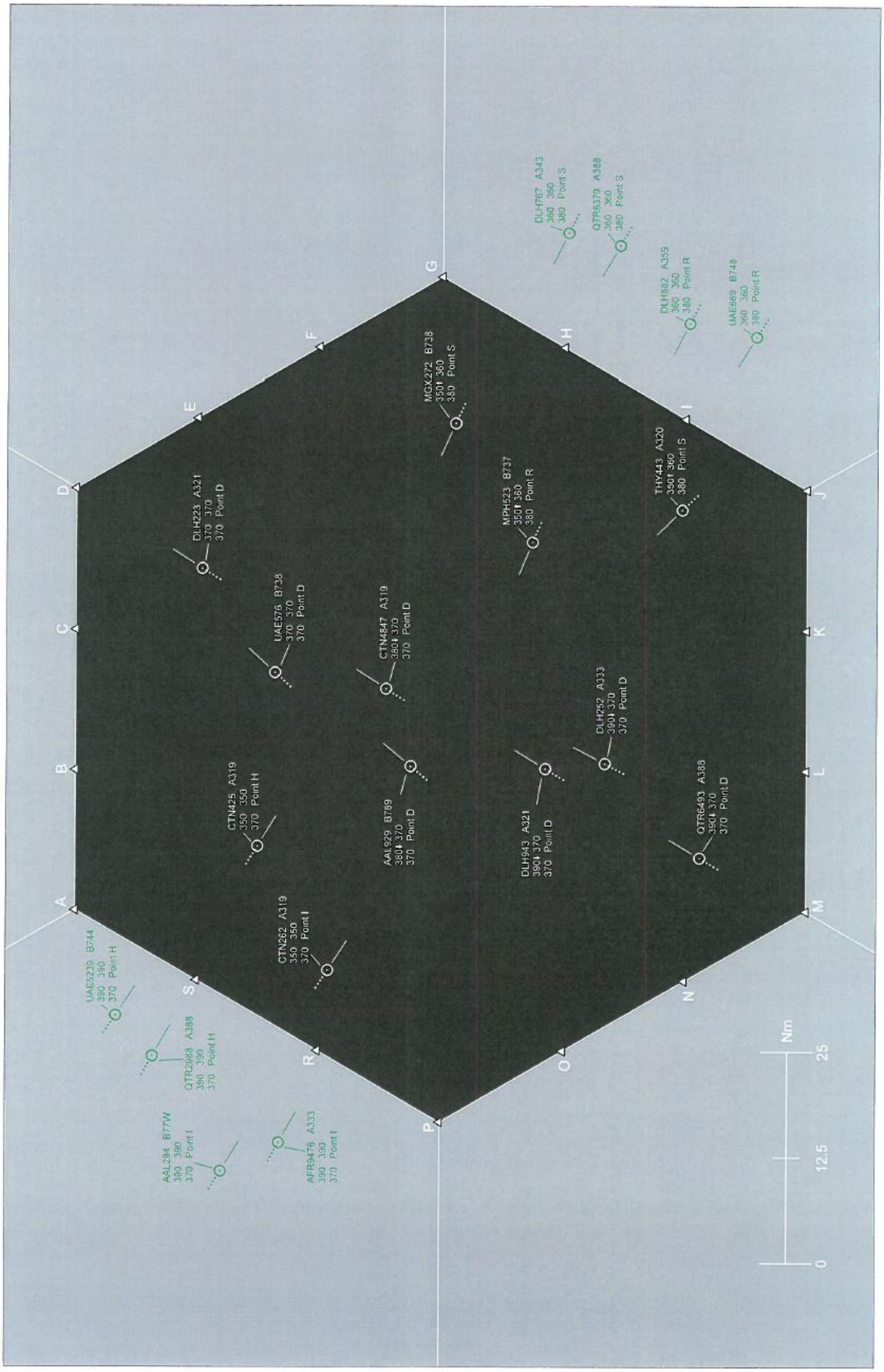


Traffic situation C31

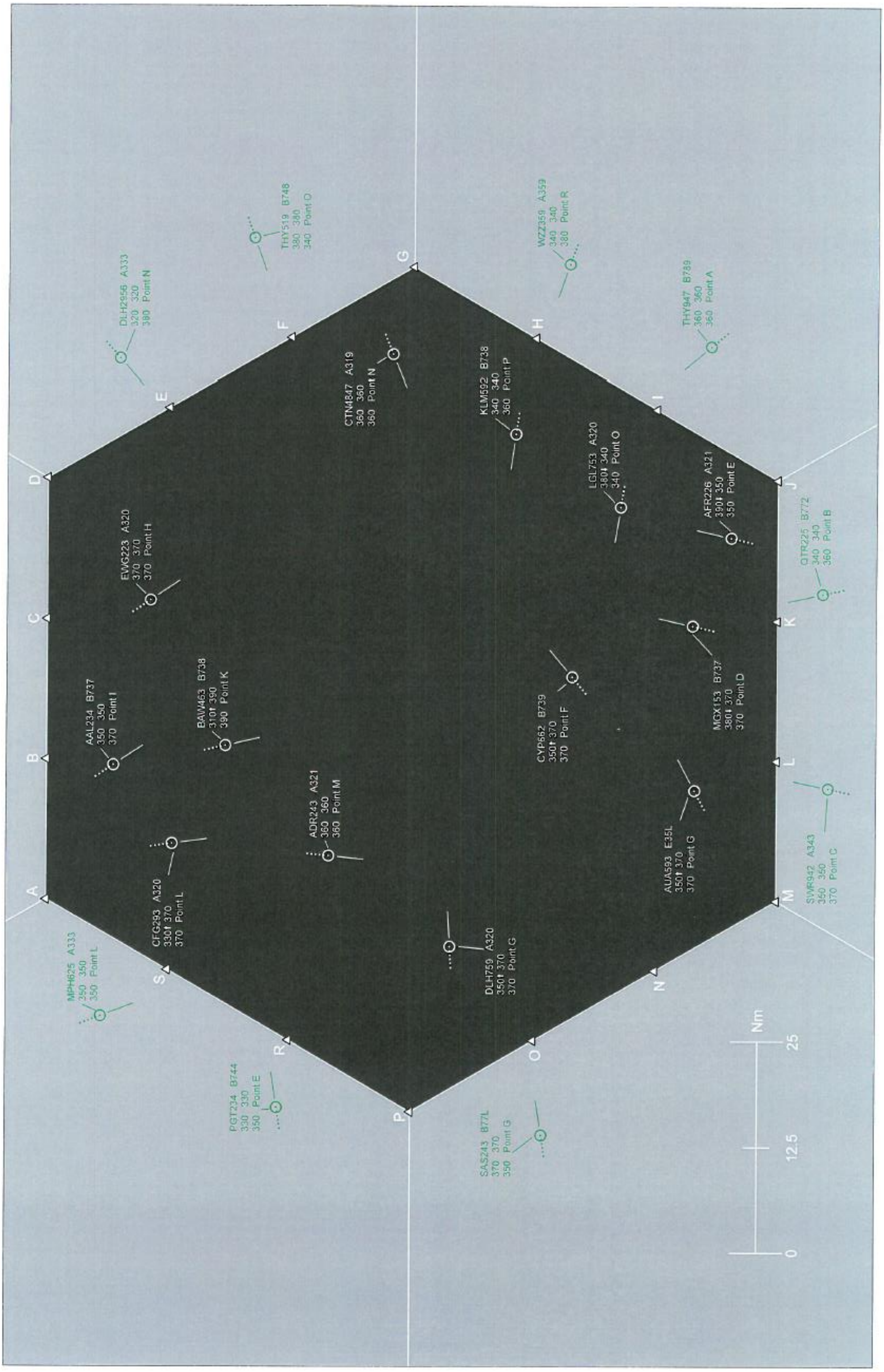


3980 R3 285 C32 049 266 670

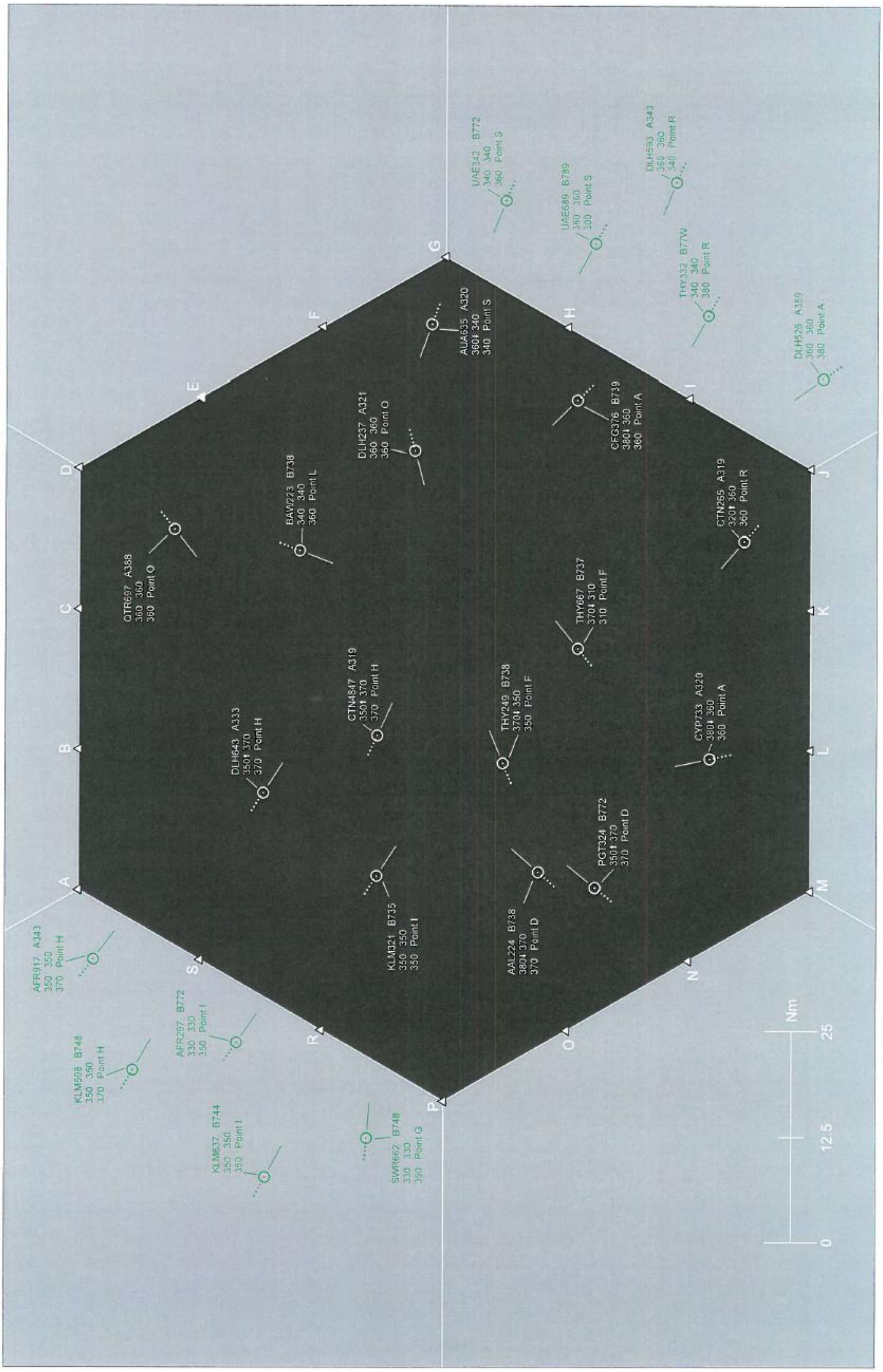
Traffic situation C32



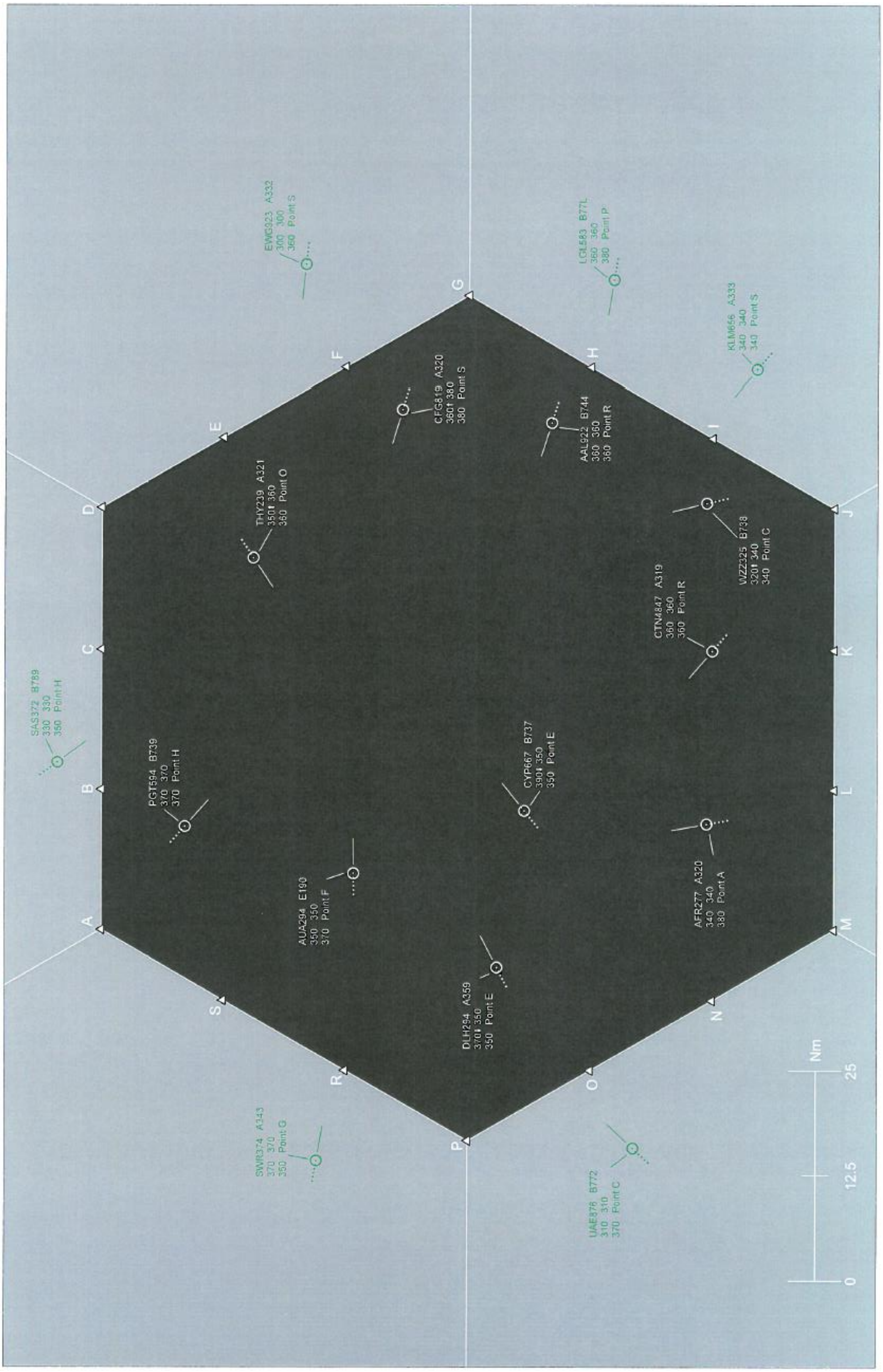
Traffic situation C33



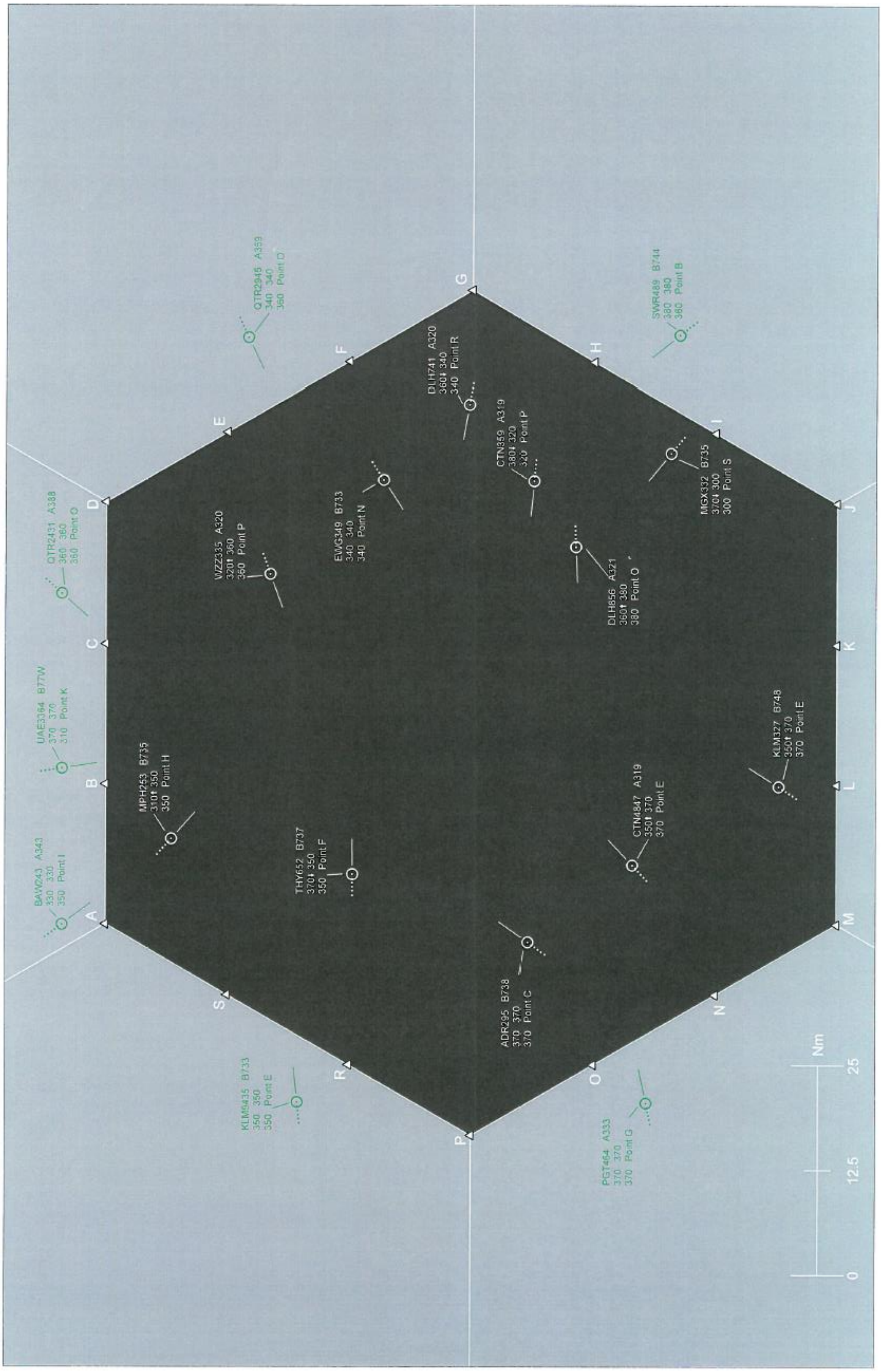
Traffic situation C34



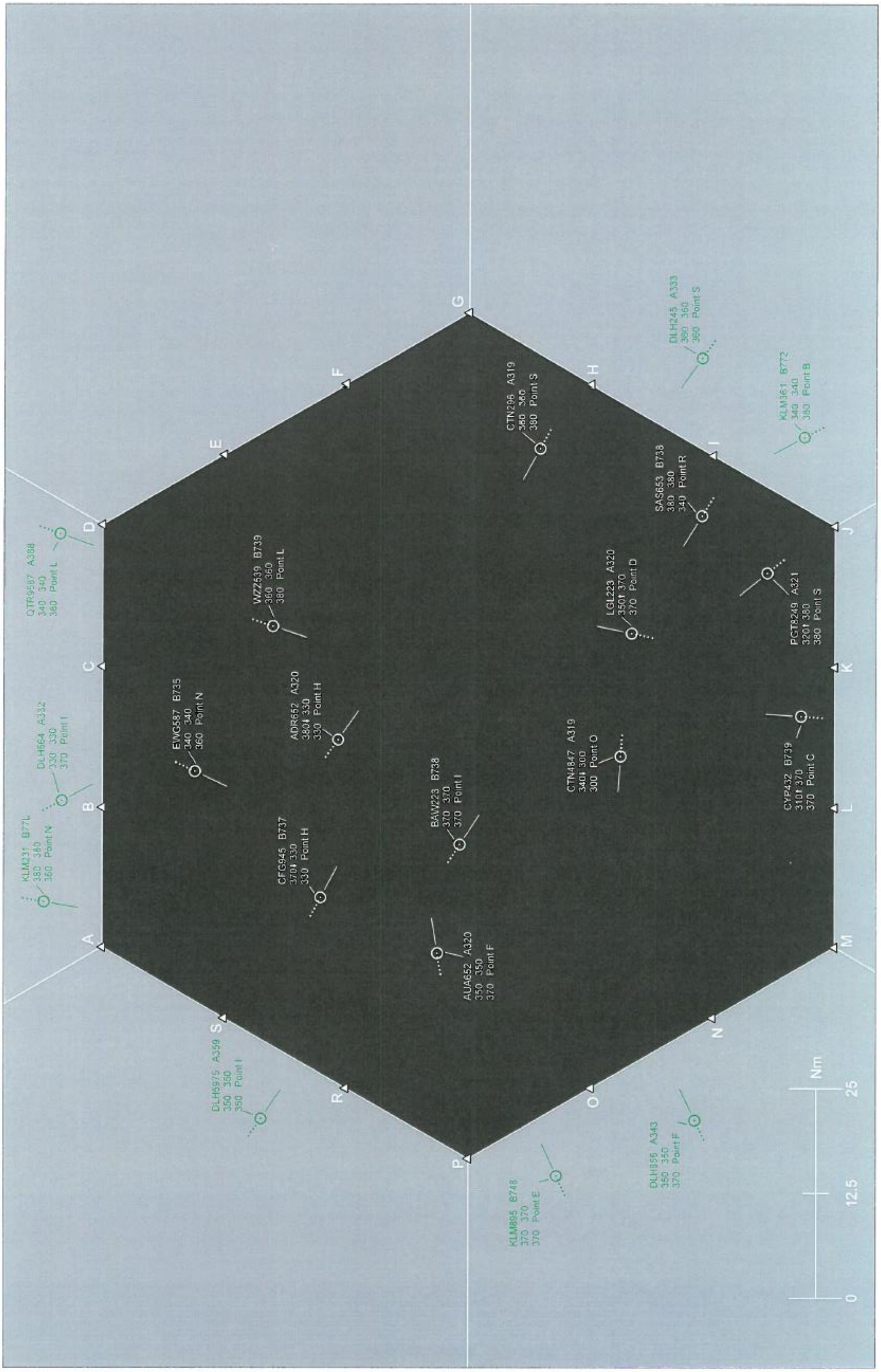
Traffic situation C35



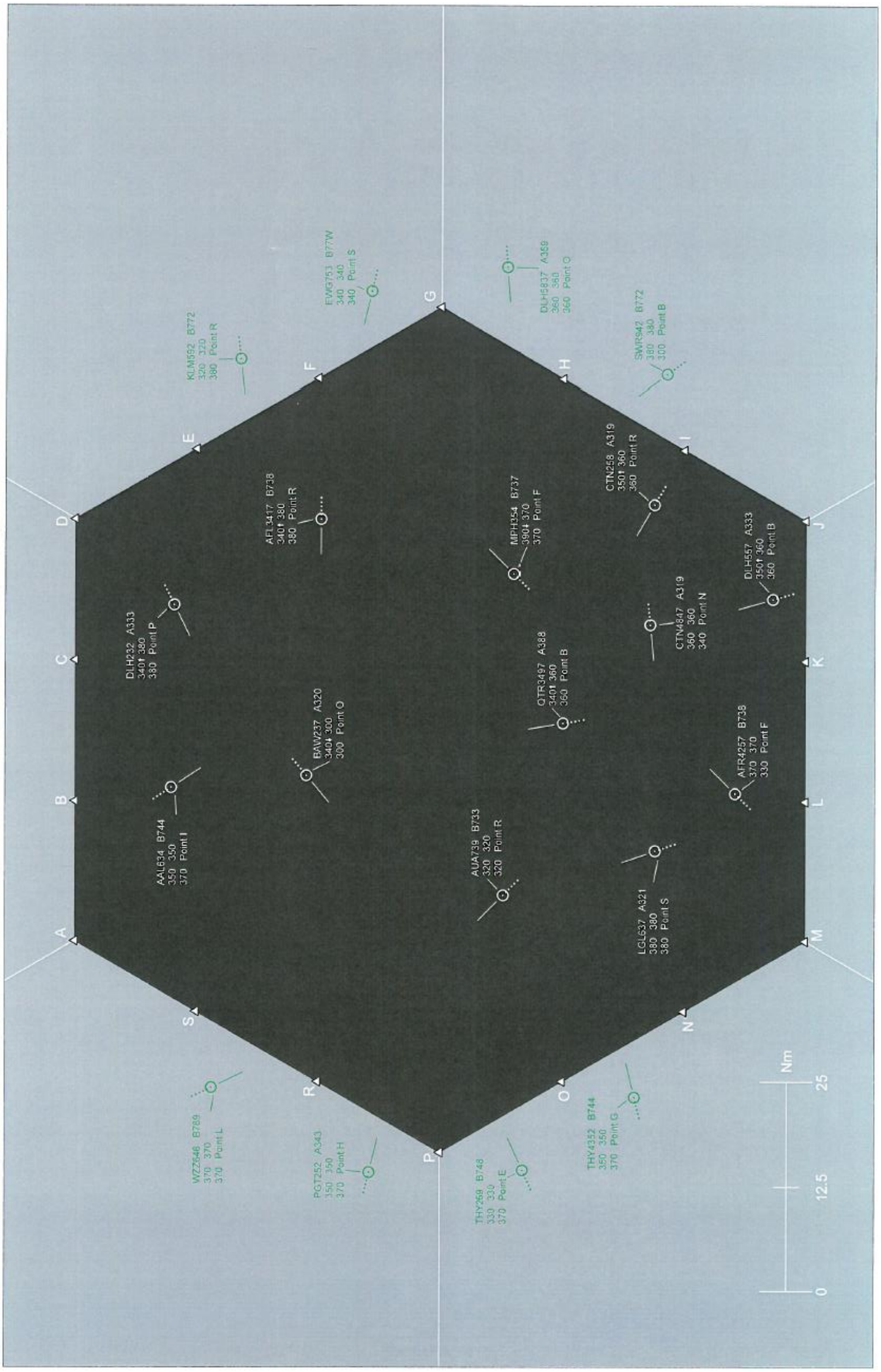
Traffic situation C36



Traffic situation C37

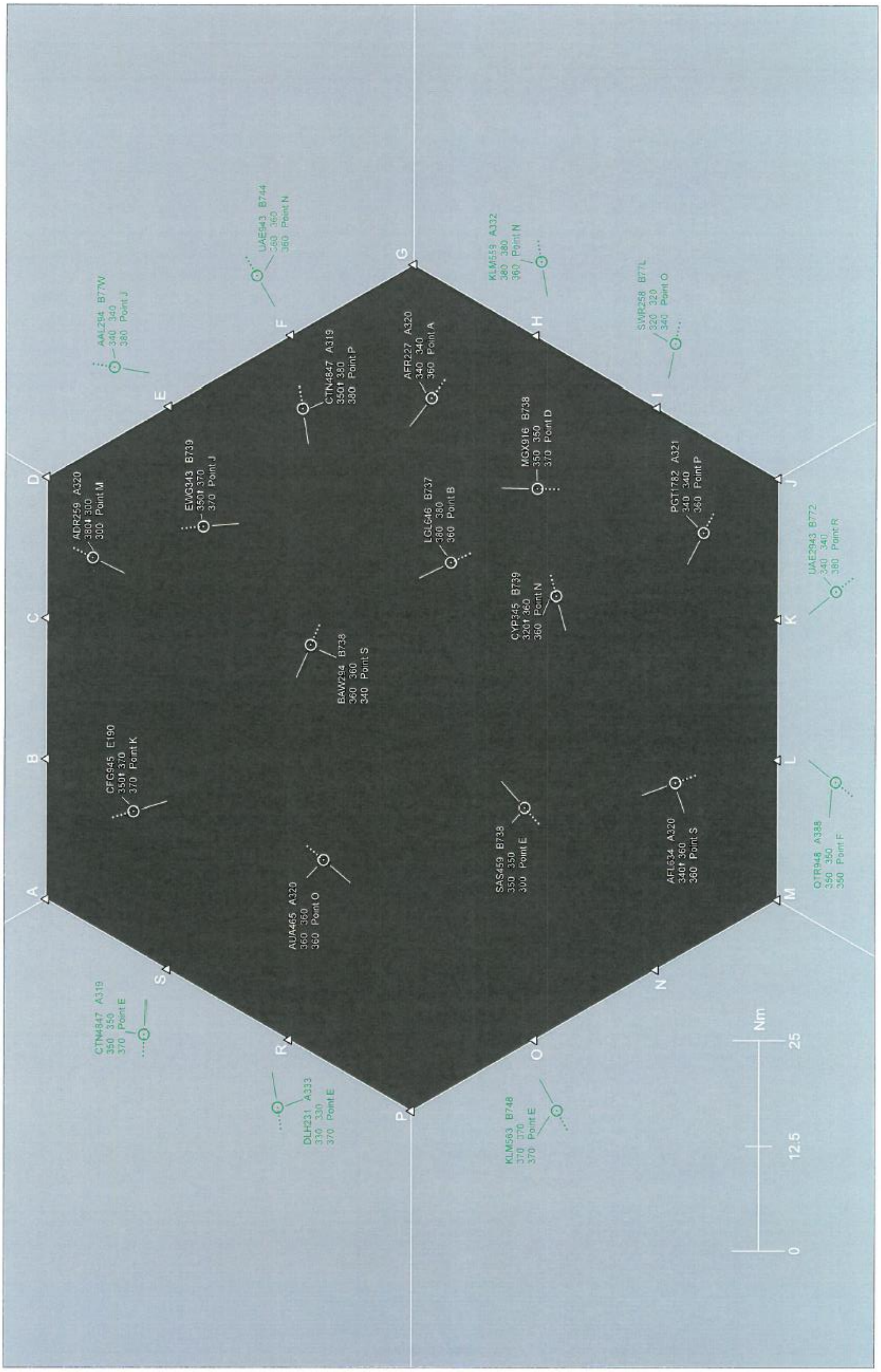


Traffic situation C38



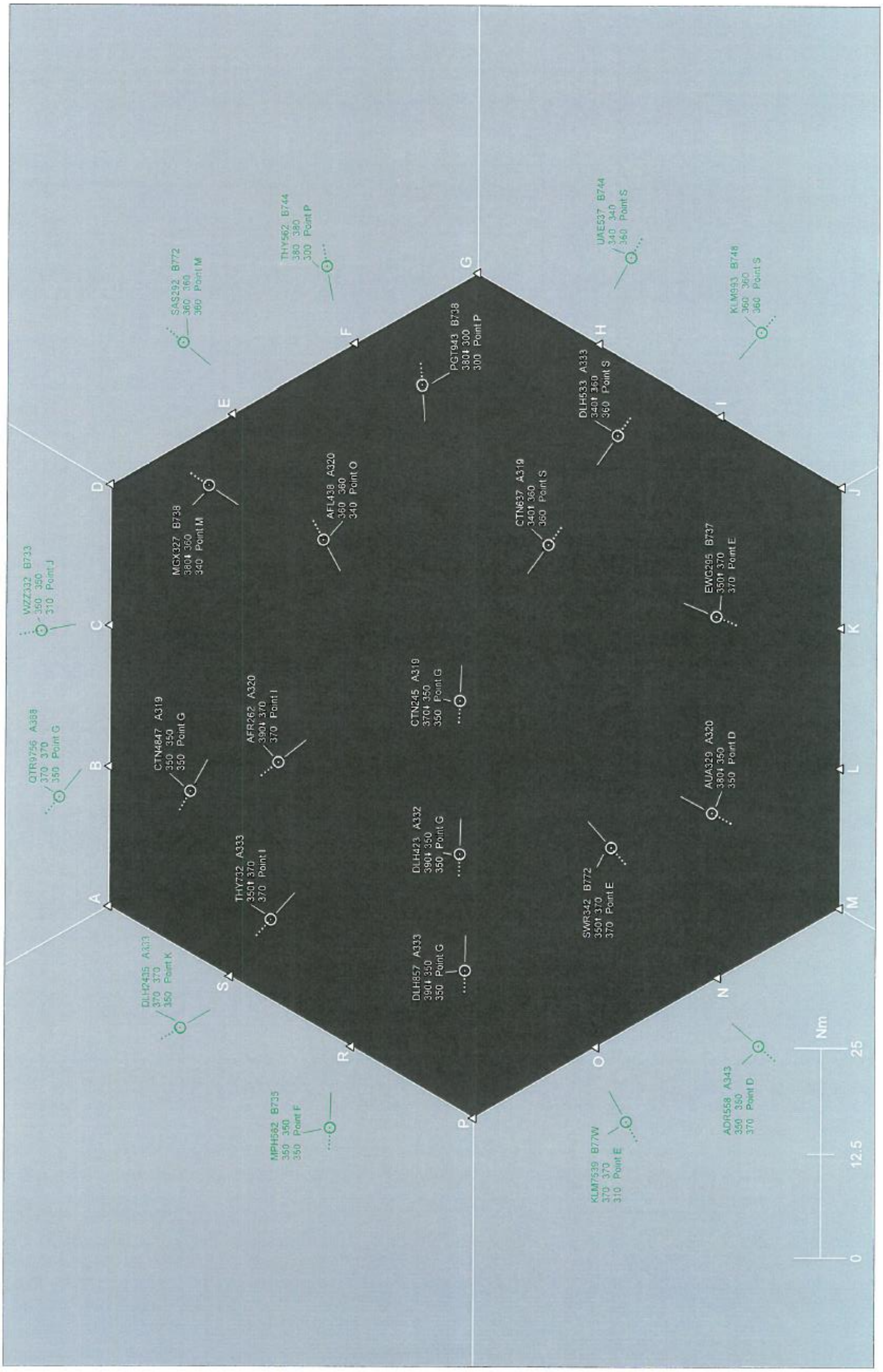
480 Q3 121 C39 141 K69 193

Traffic situation C39



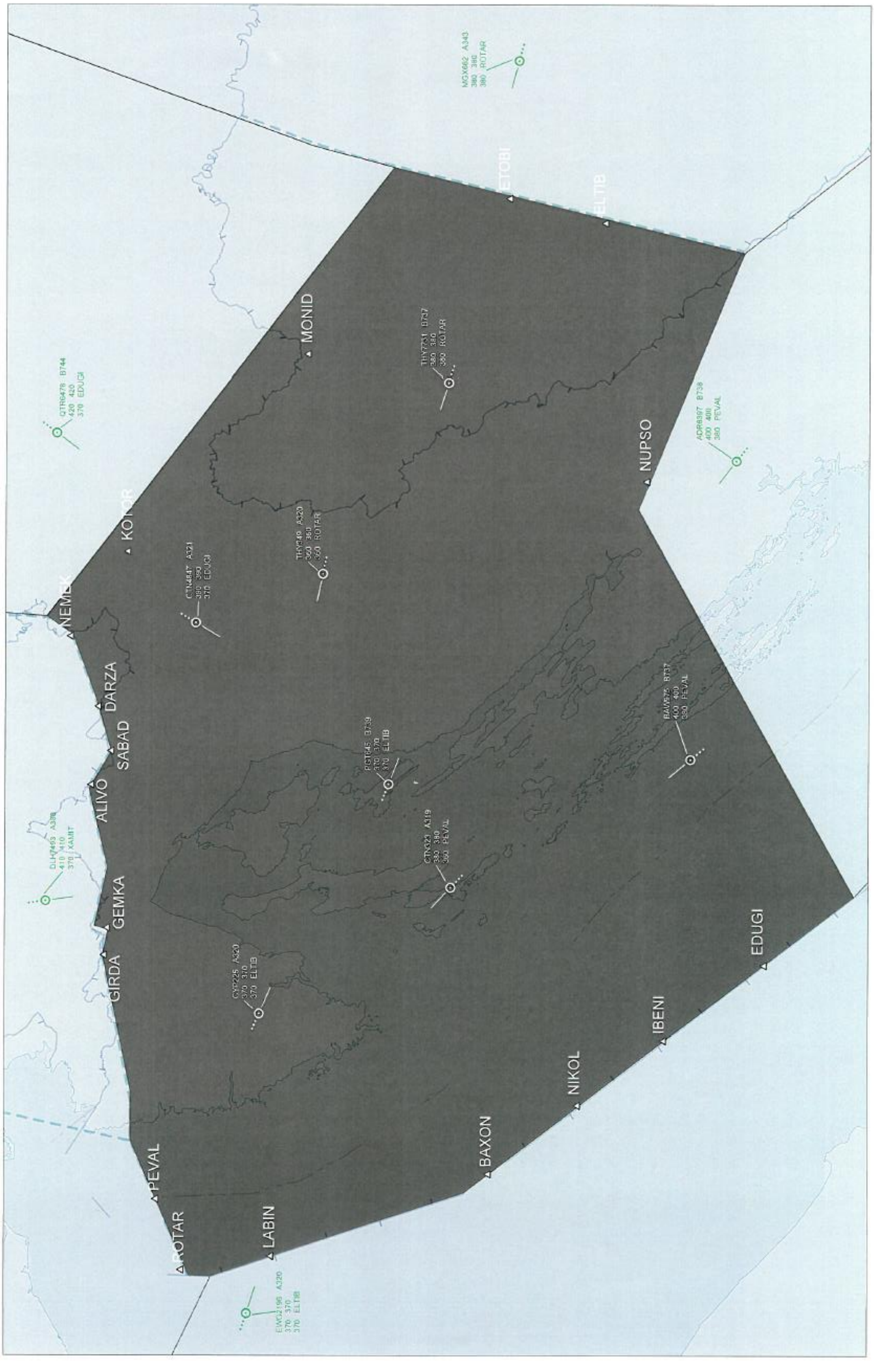
3305 U2 858 C40 340 Z29 333

Traffic situation C40



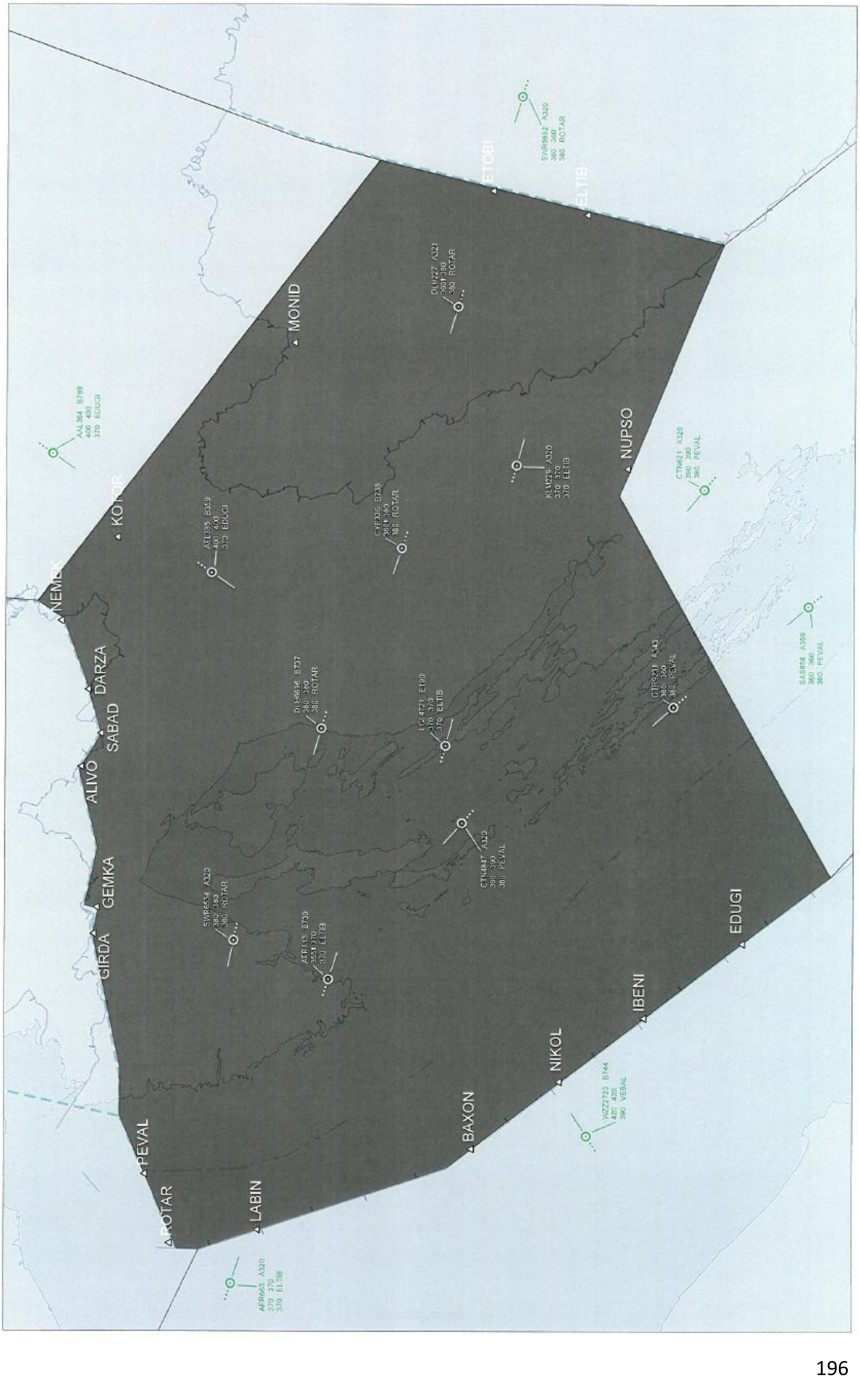
Appendix 2 – Validation traffic situations

Traffic situation V1



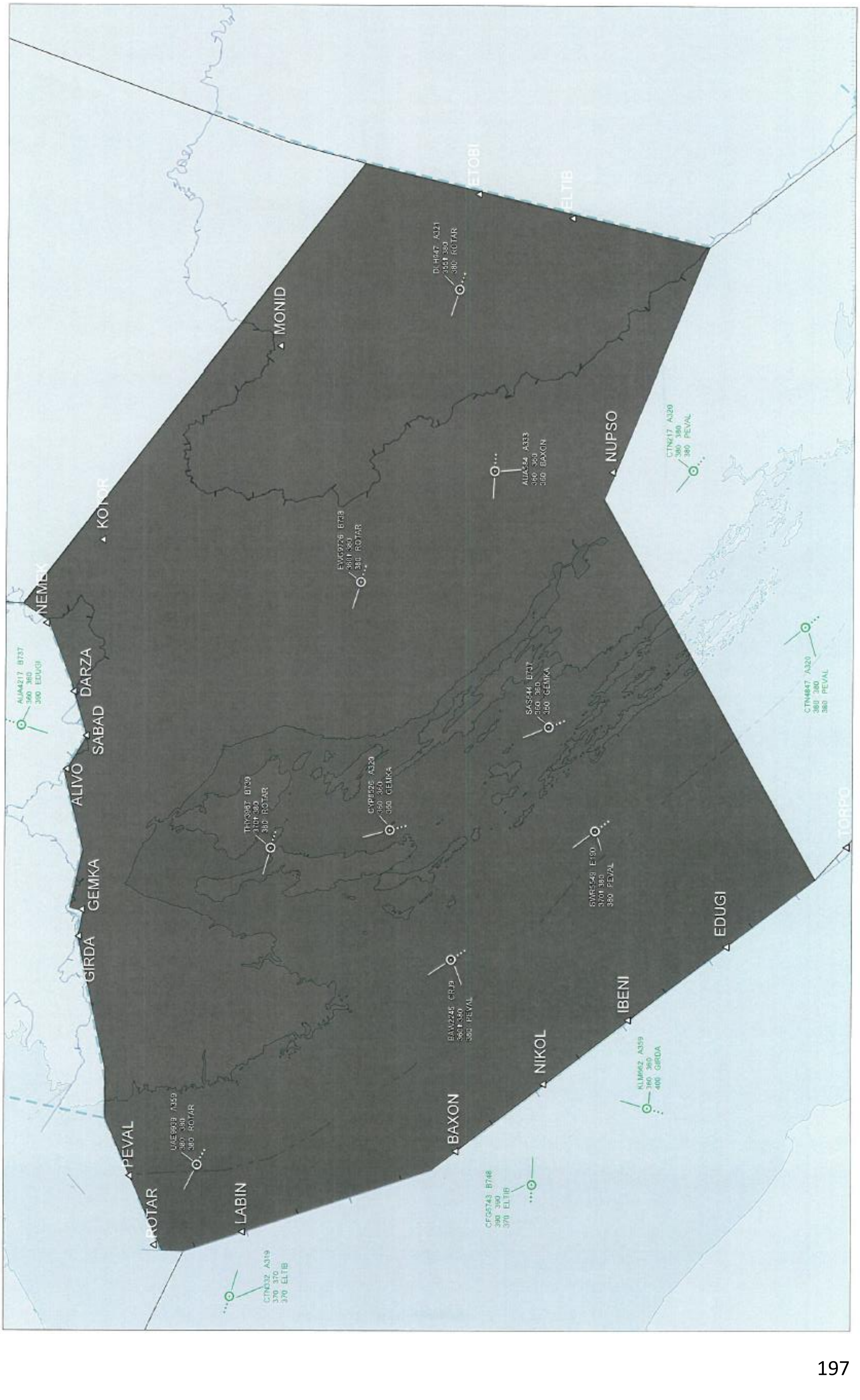
Traffic situation V2

7831 G7 681 V2 318 H97 824



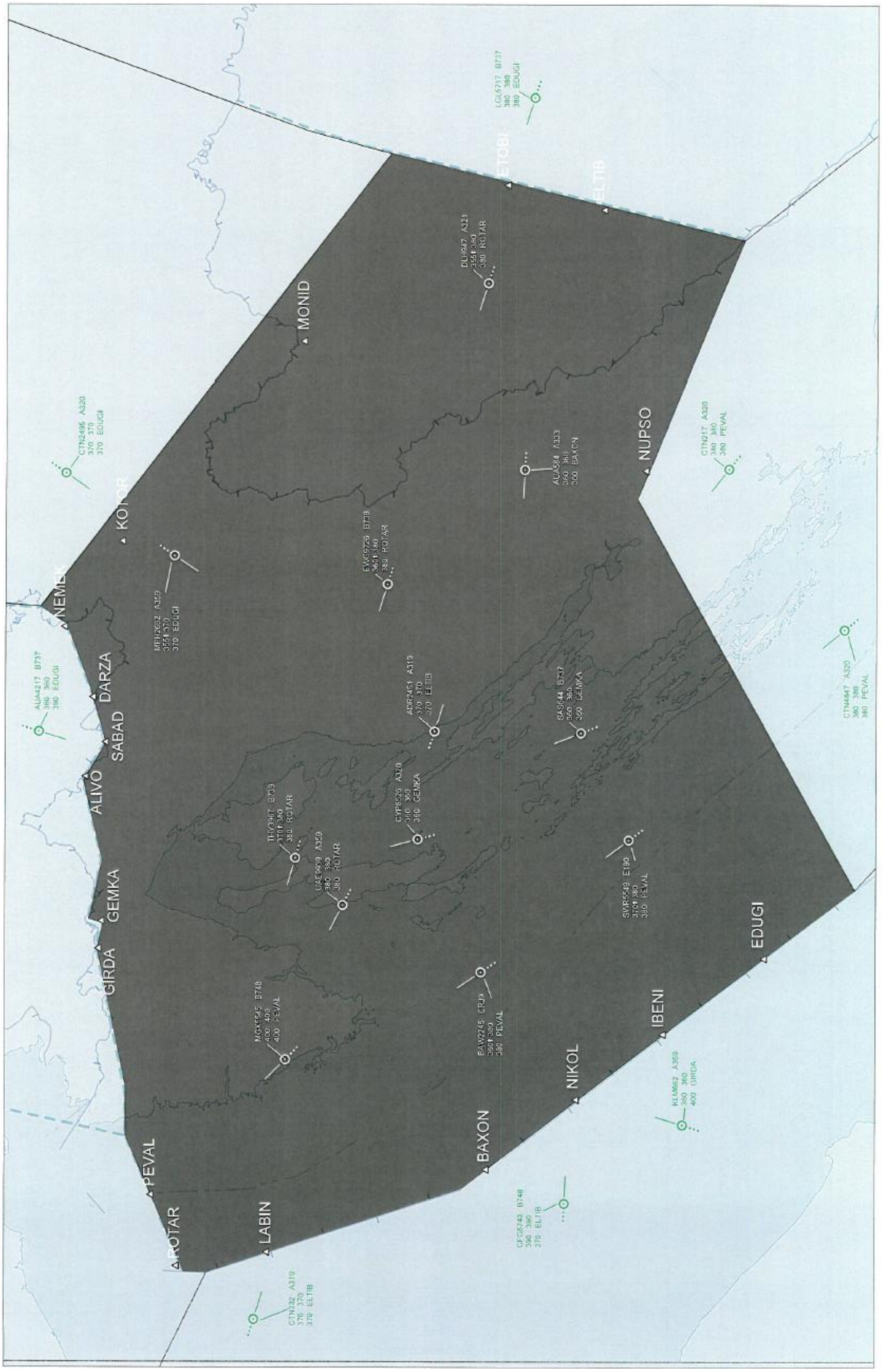
1975 U3 145 V3 974 P14 249

Traffic situation V3

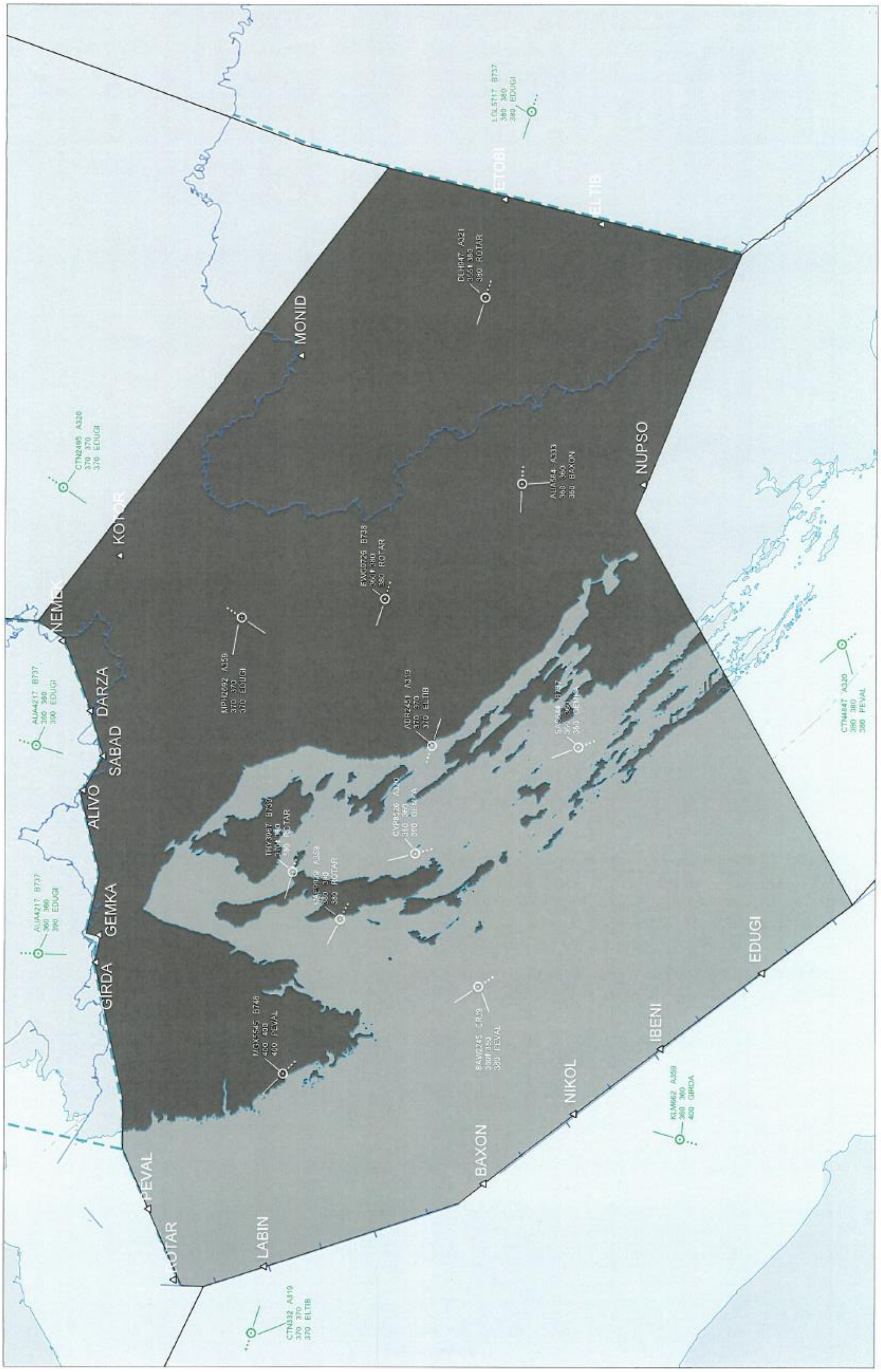


Traffic situation V4

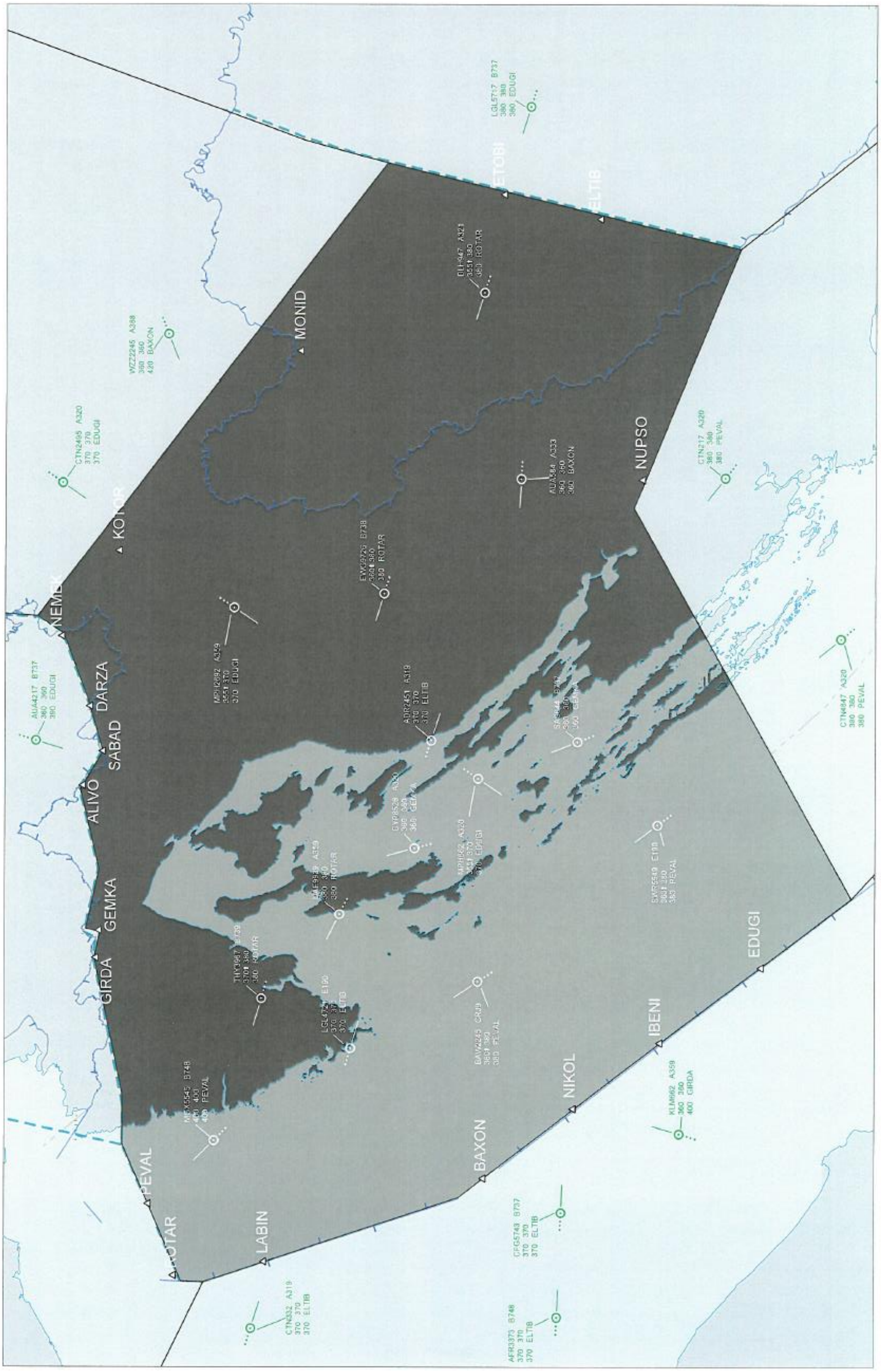
6874 J8 349 V4 971 N74 647



Traffic situation V5

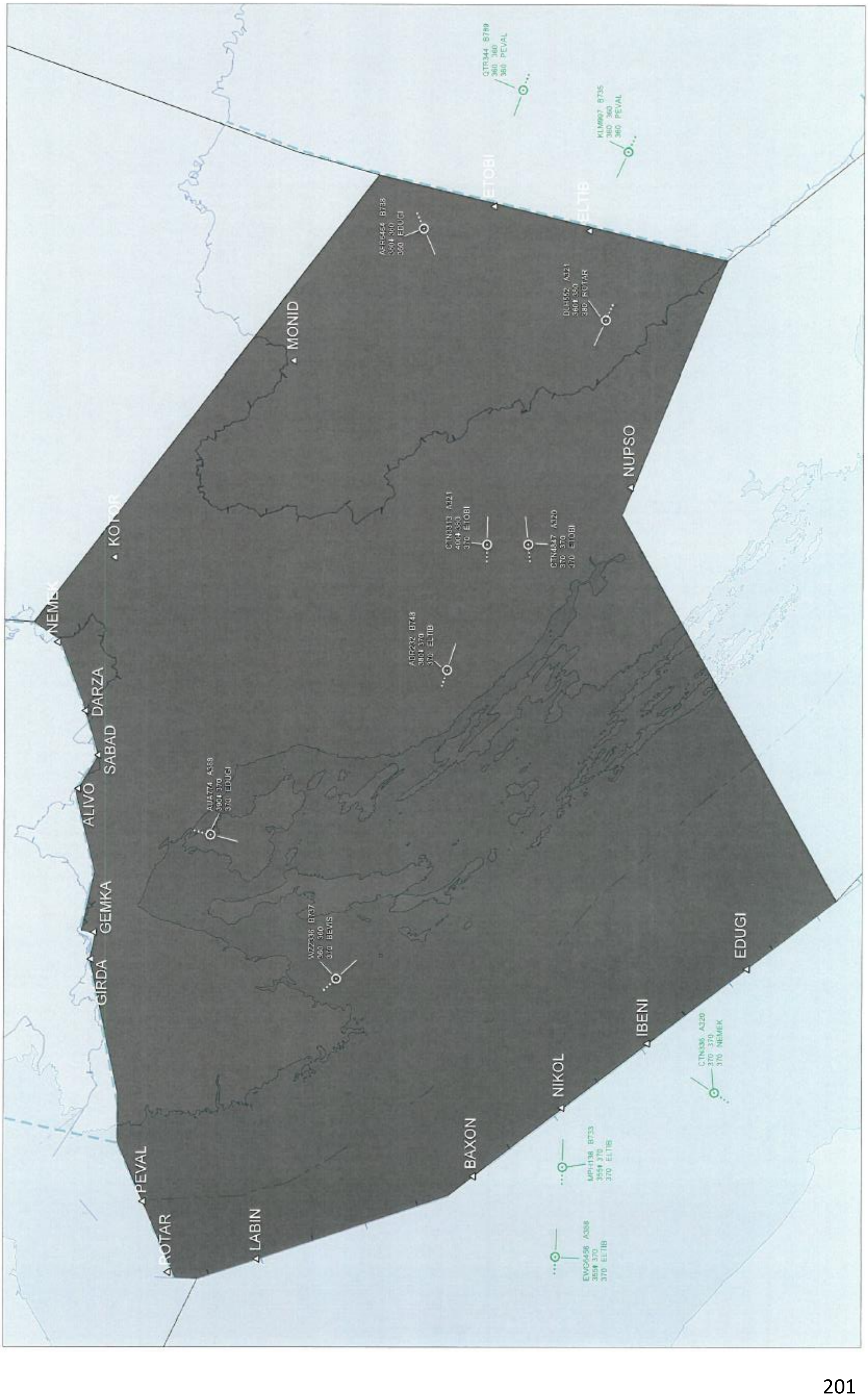


Traffic situation V6



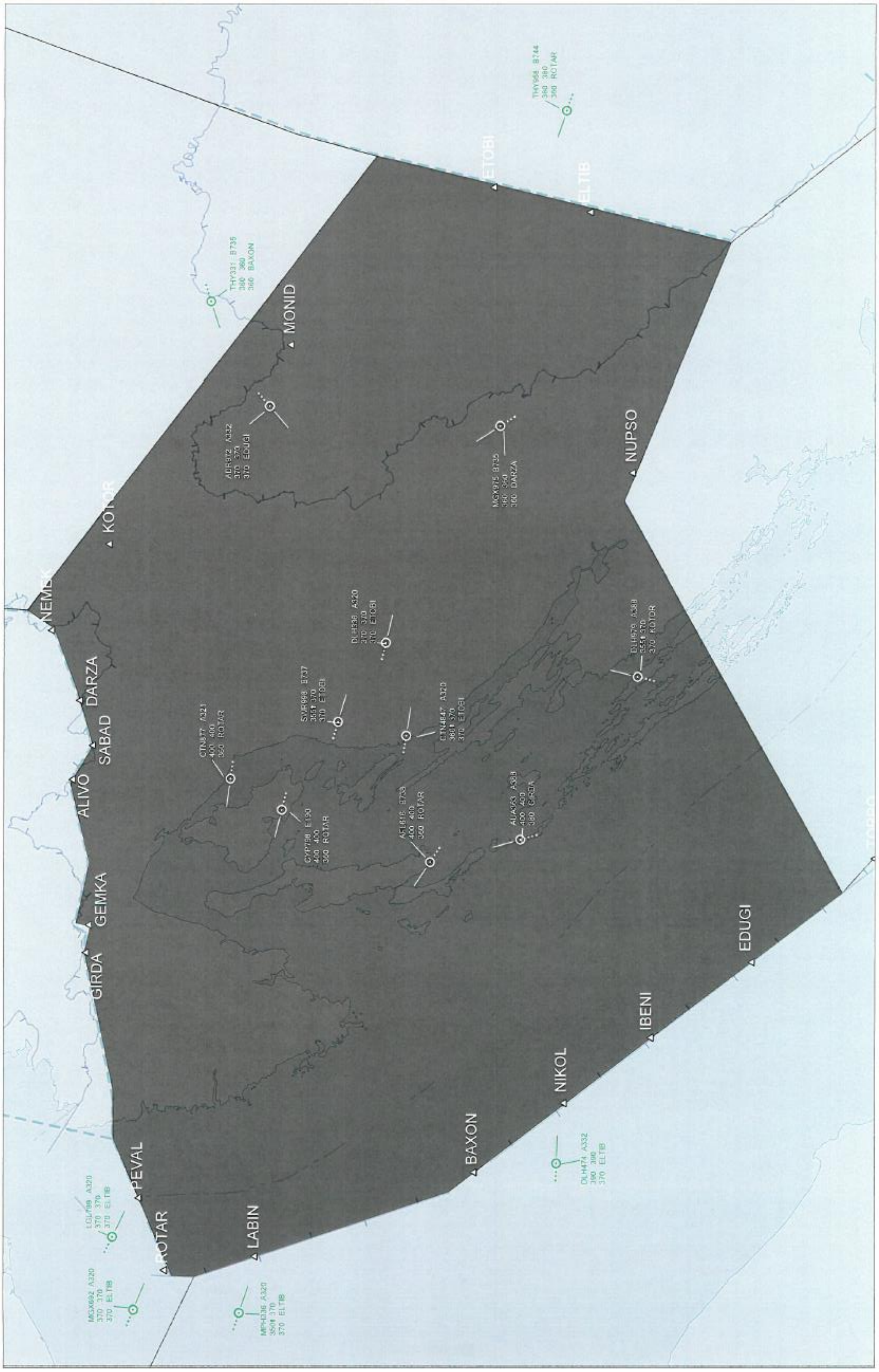
2294 X7 364 V6 294 K42 316

Traffic situation V7

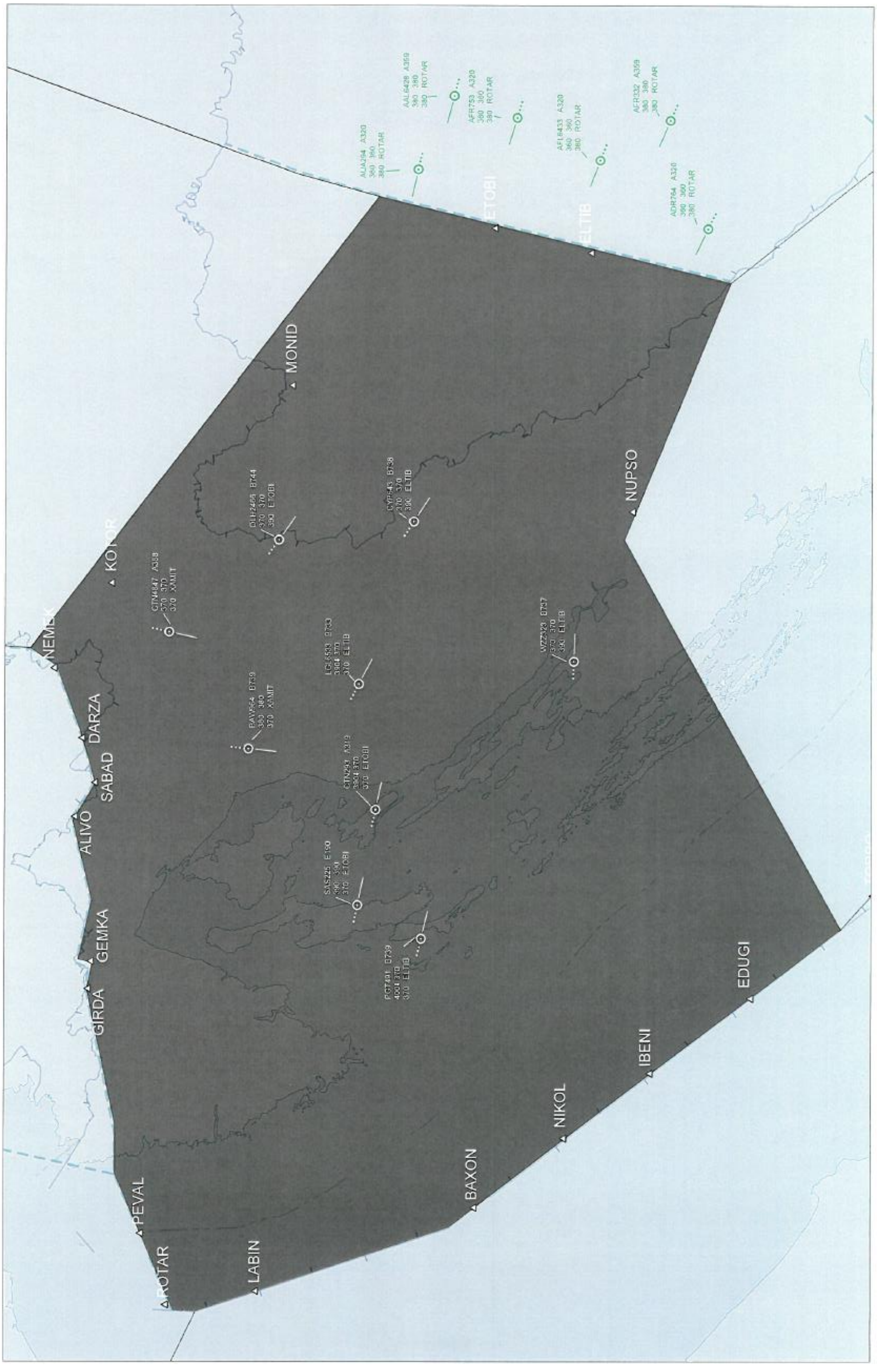


7831 G7 681 V8 318 H97 824

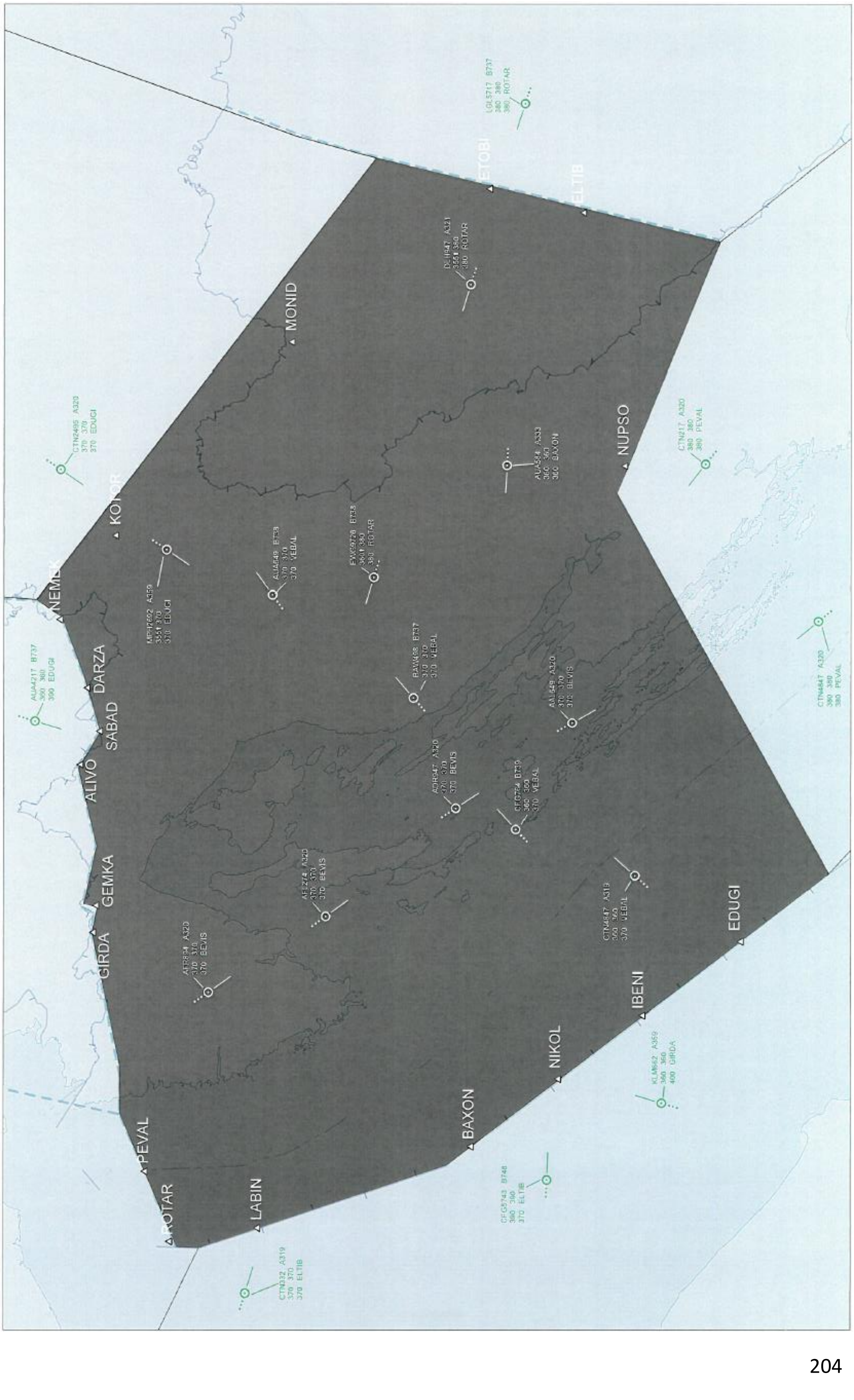
Traffic situation V8



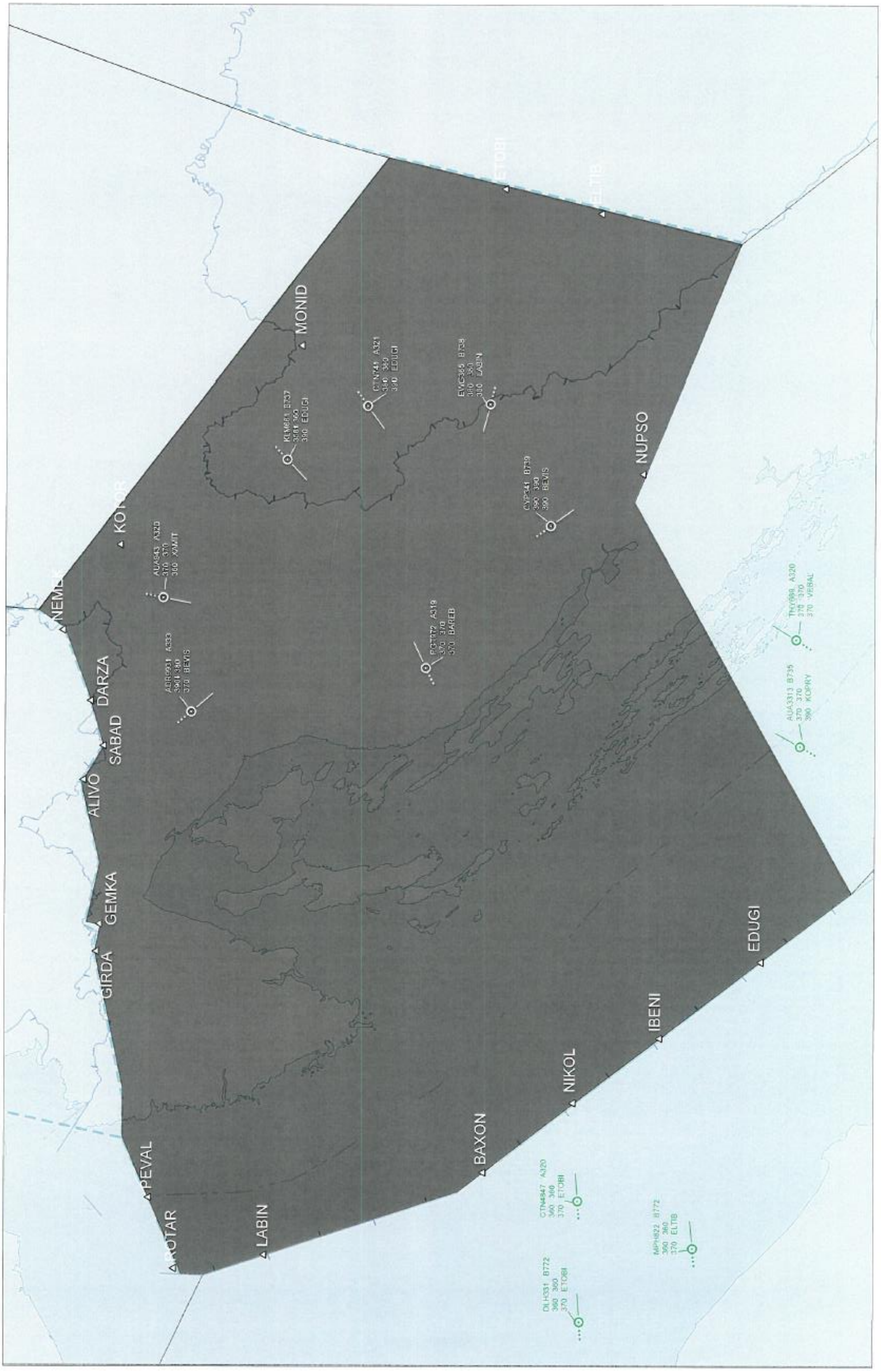
Traffic situation V9



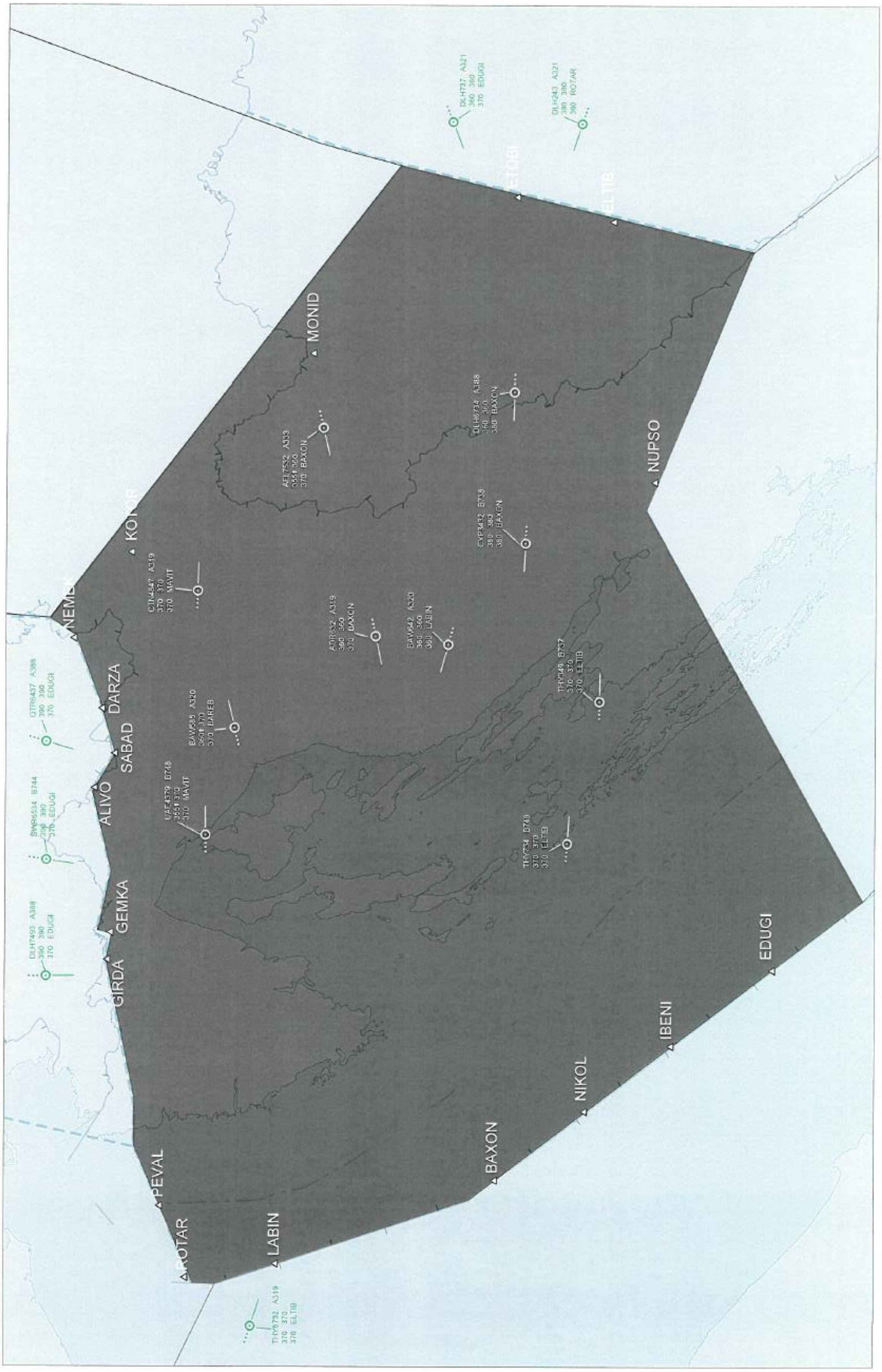
Traffic situation V10



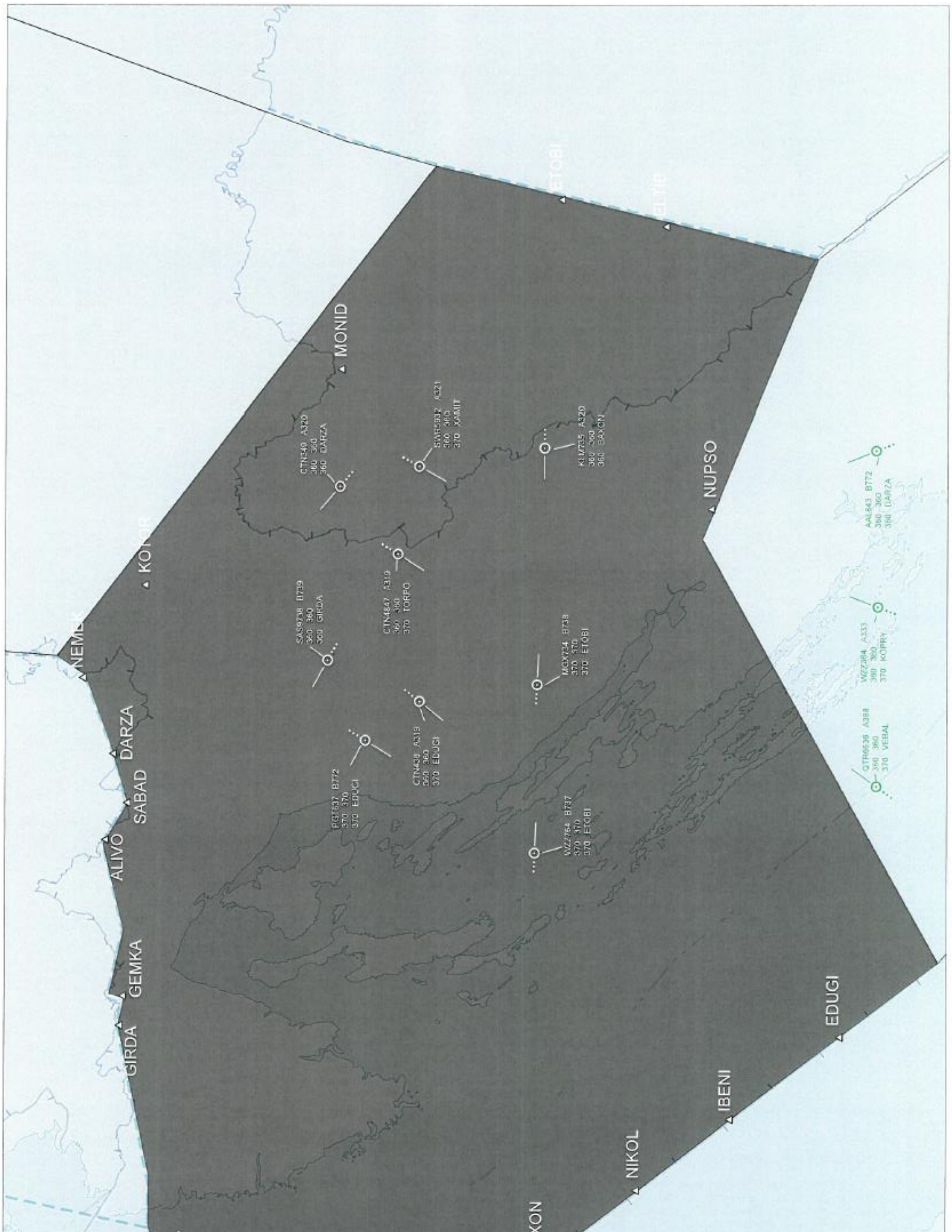
Traffic situation V11



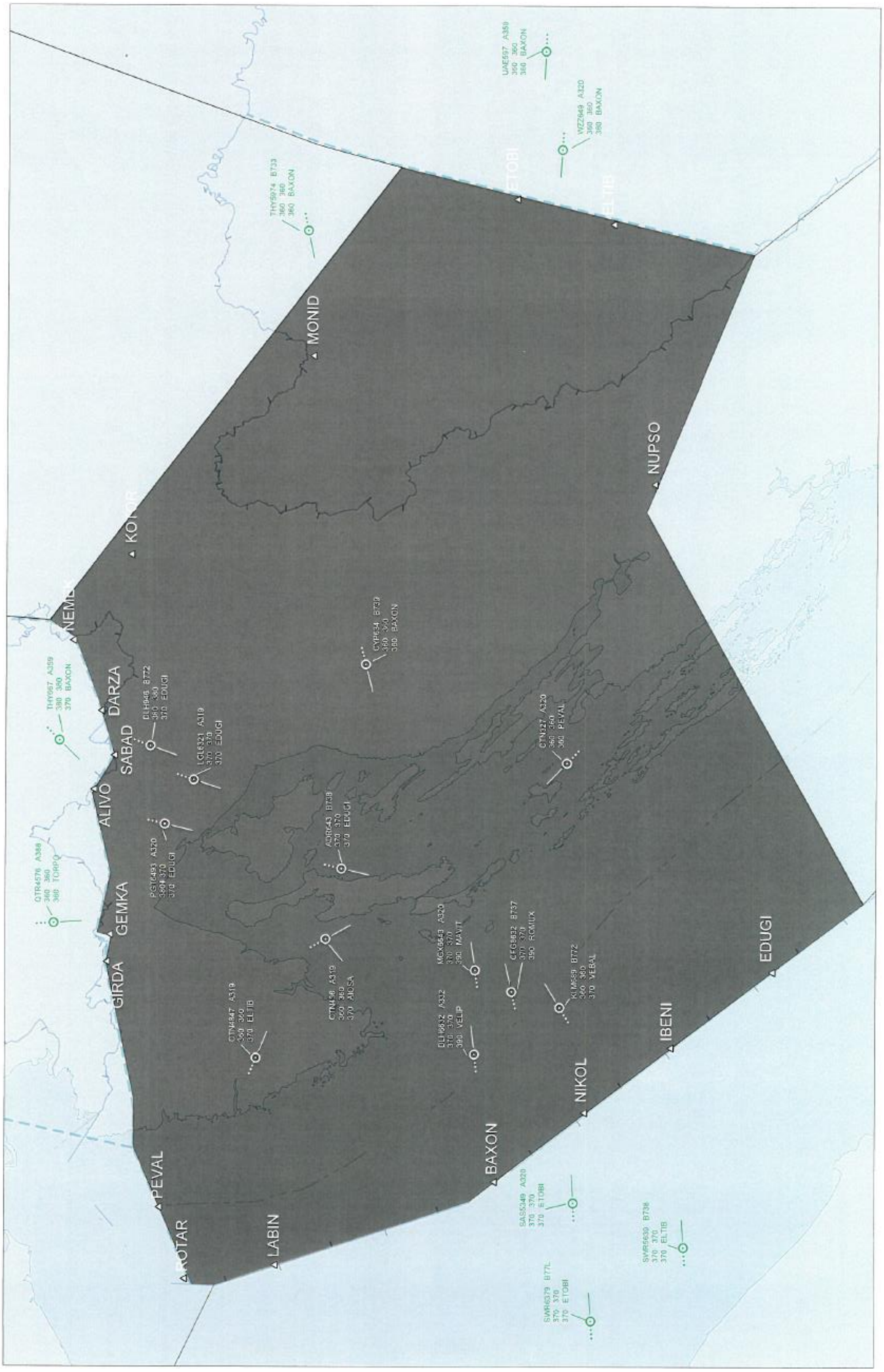
Traffic situation V12



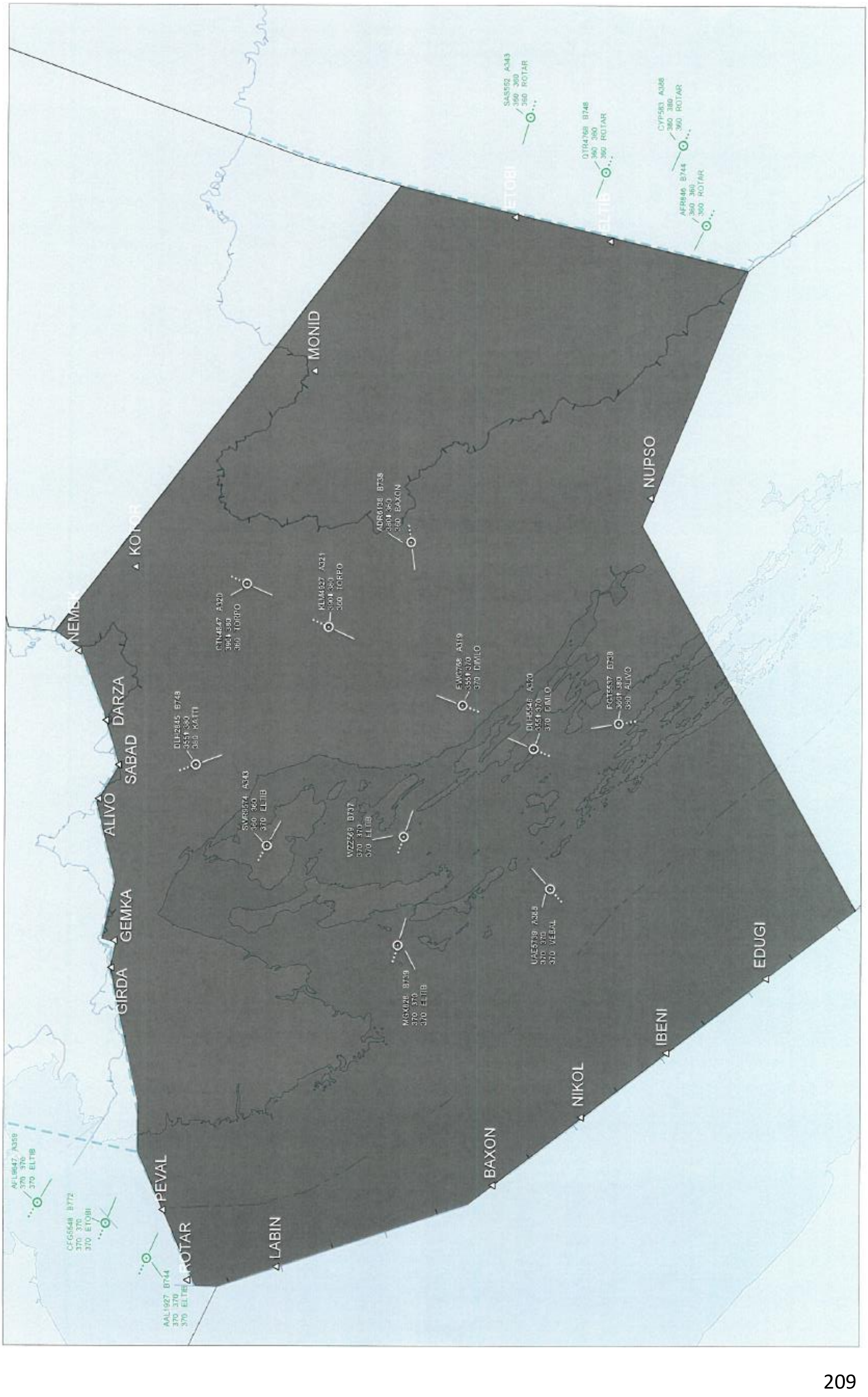
Traffic situation V13



Traffic situation V14

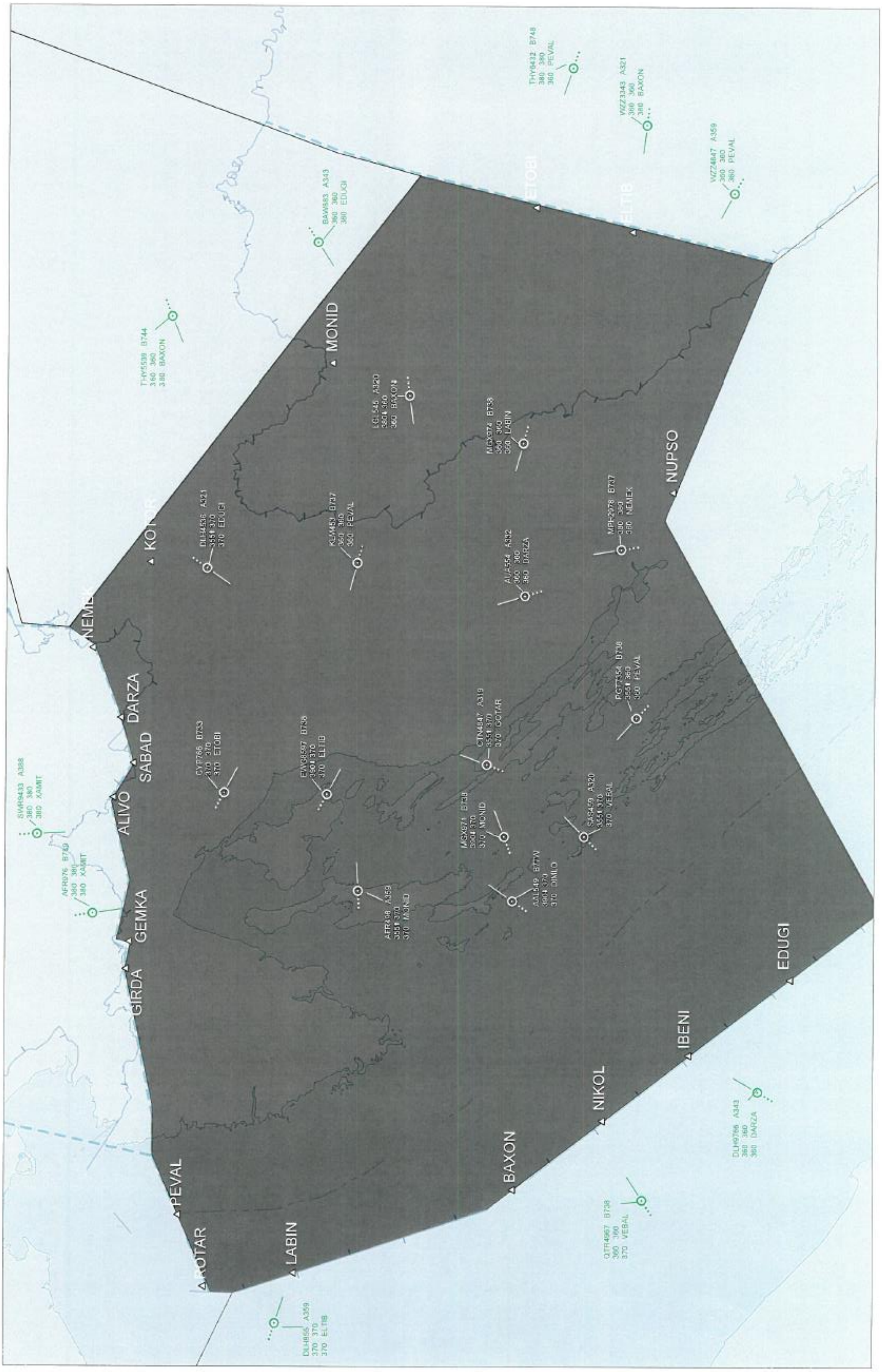


Traffic situation V15

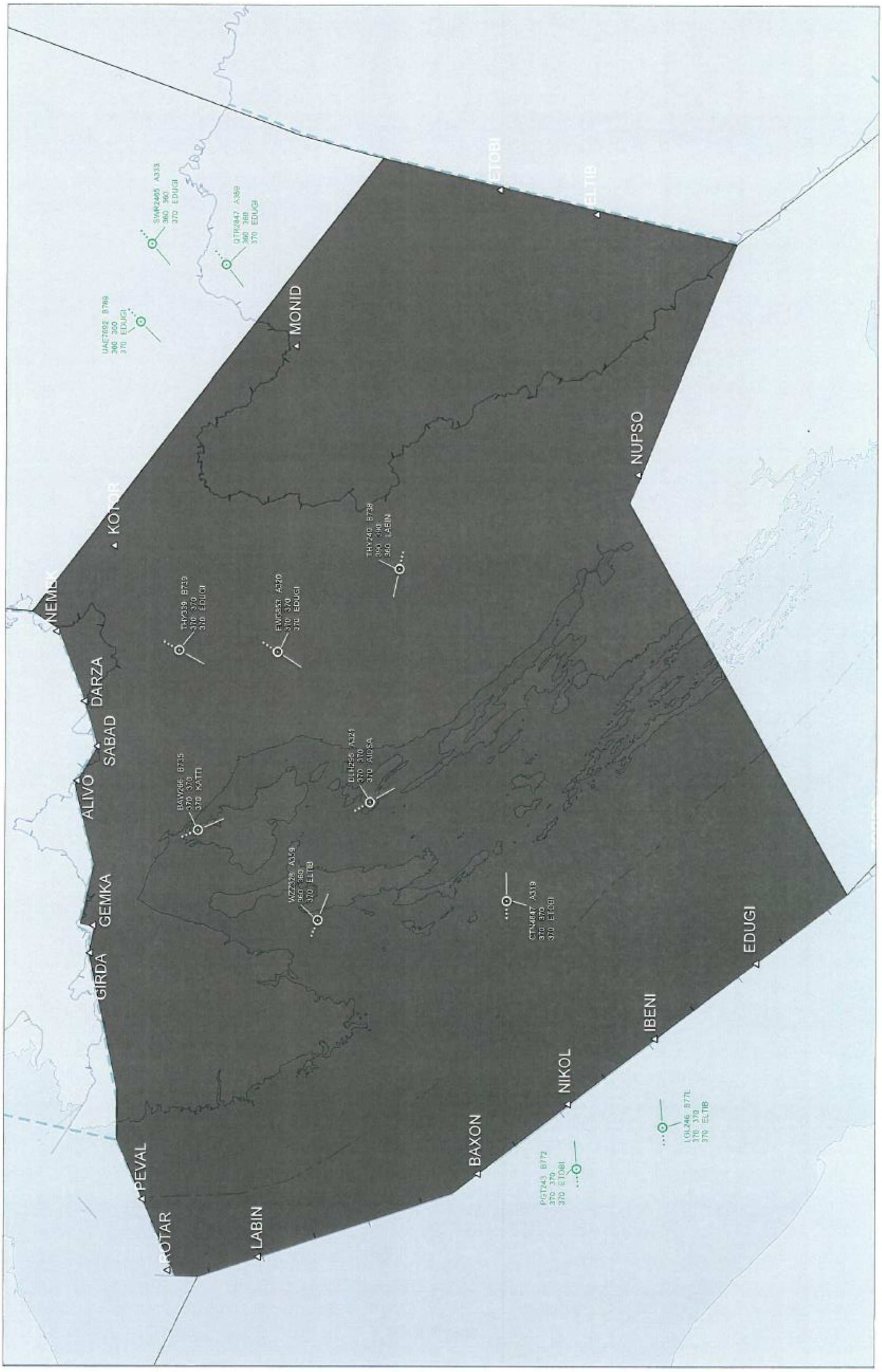


455 F5 464 V18 564 M55 545

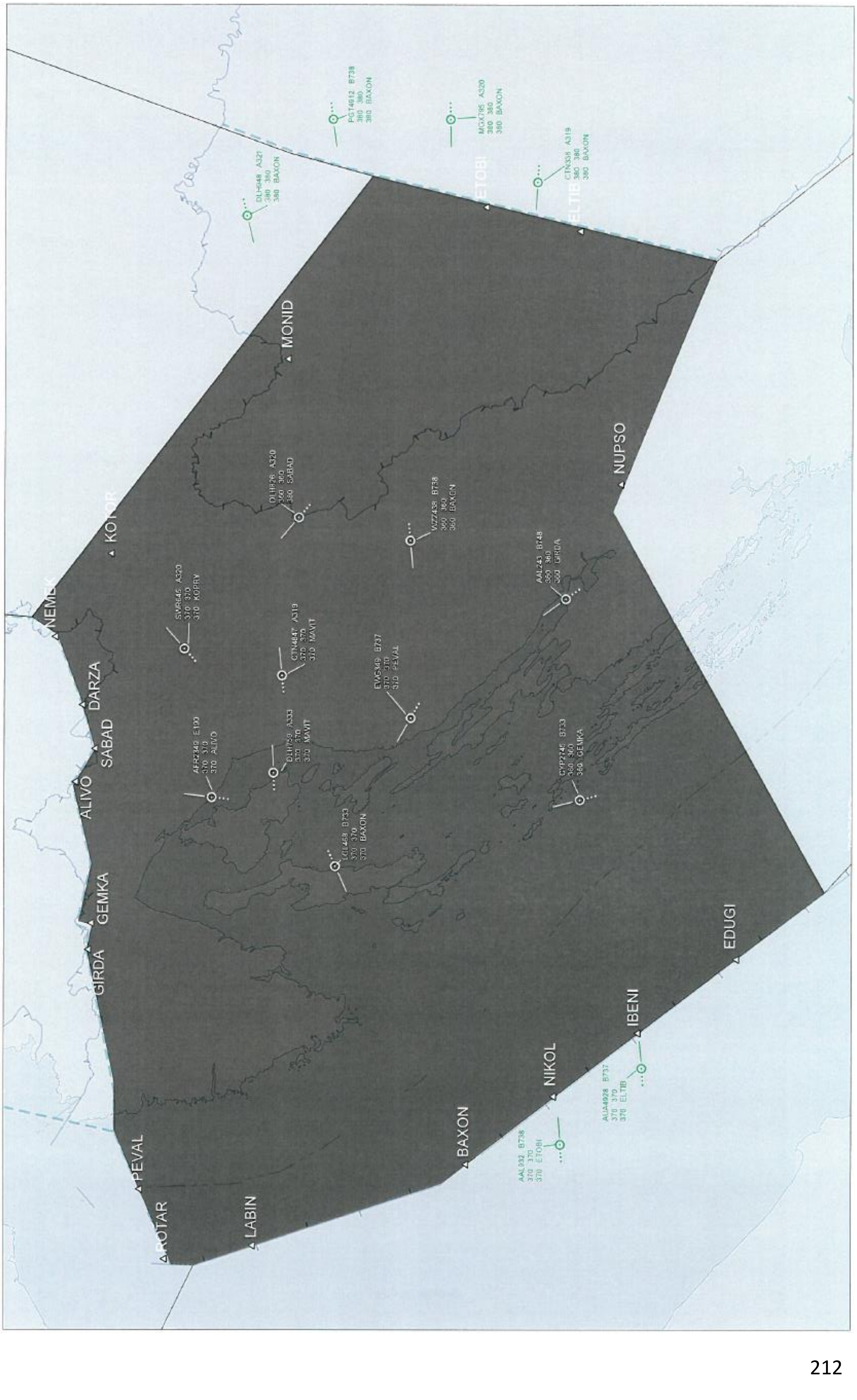
Traffic situation V16



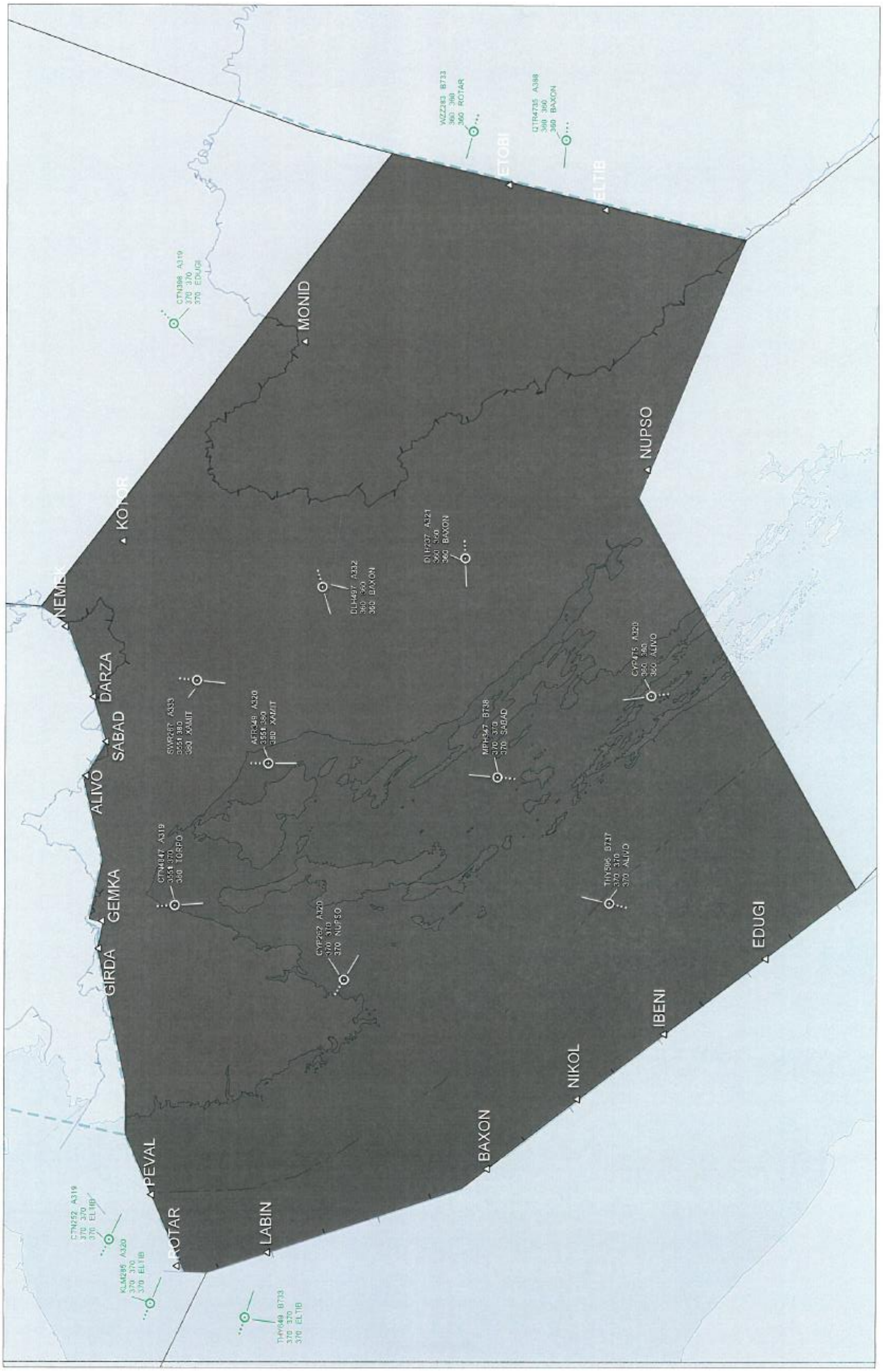
Traffic situation V17



Traffic situation V18

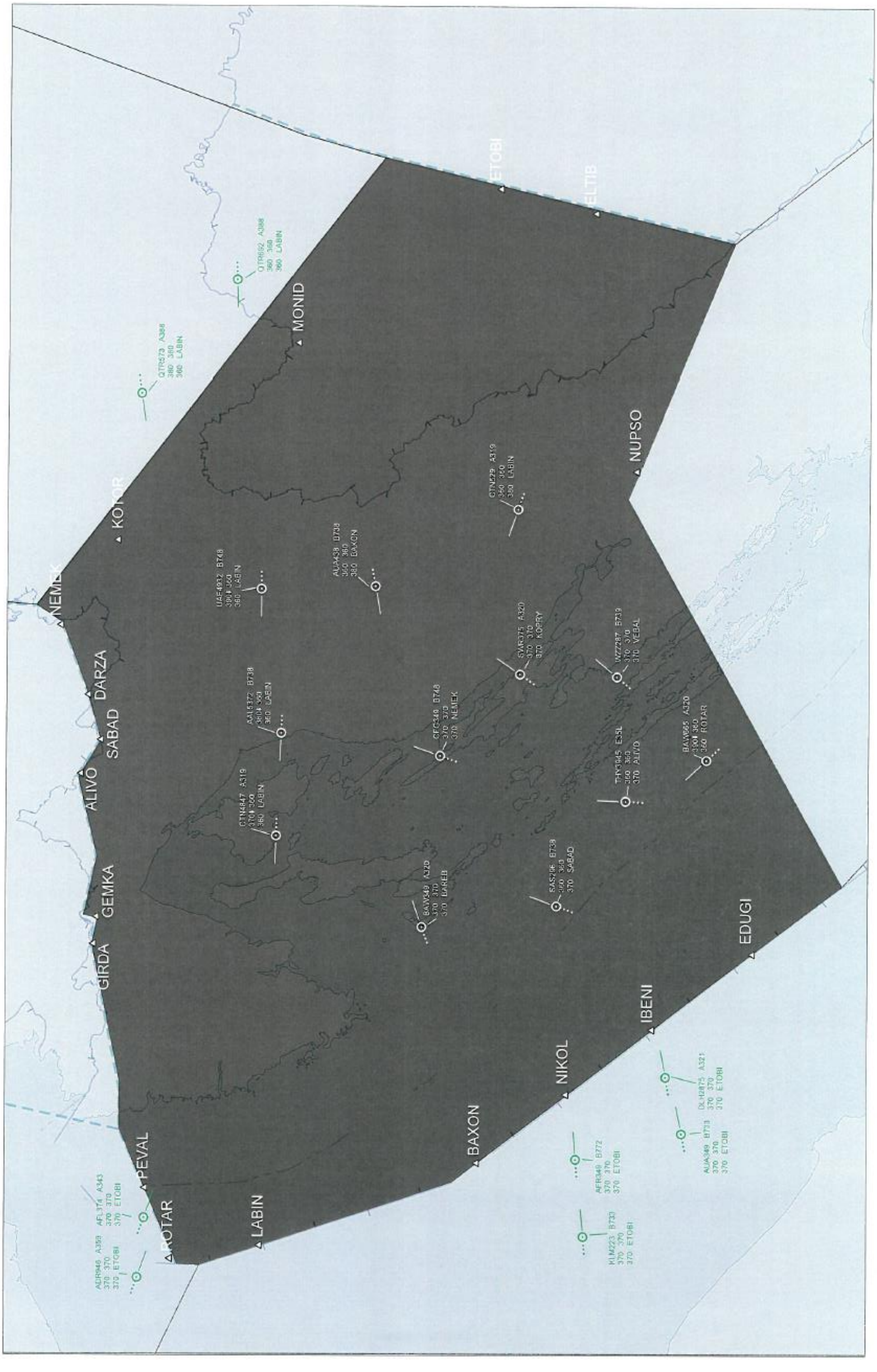


Traffic situation V19

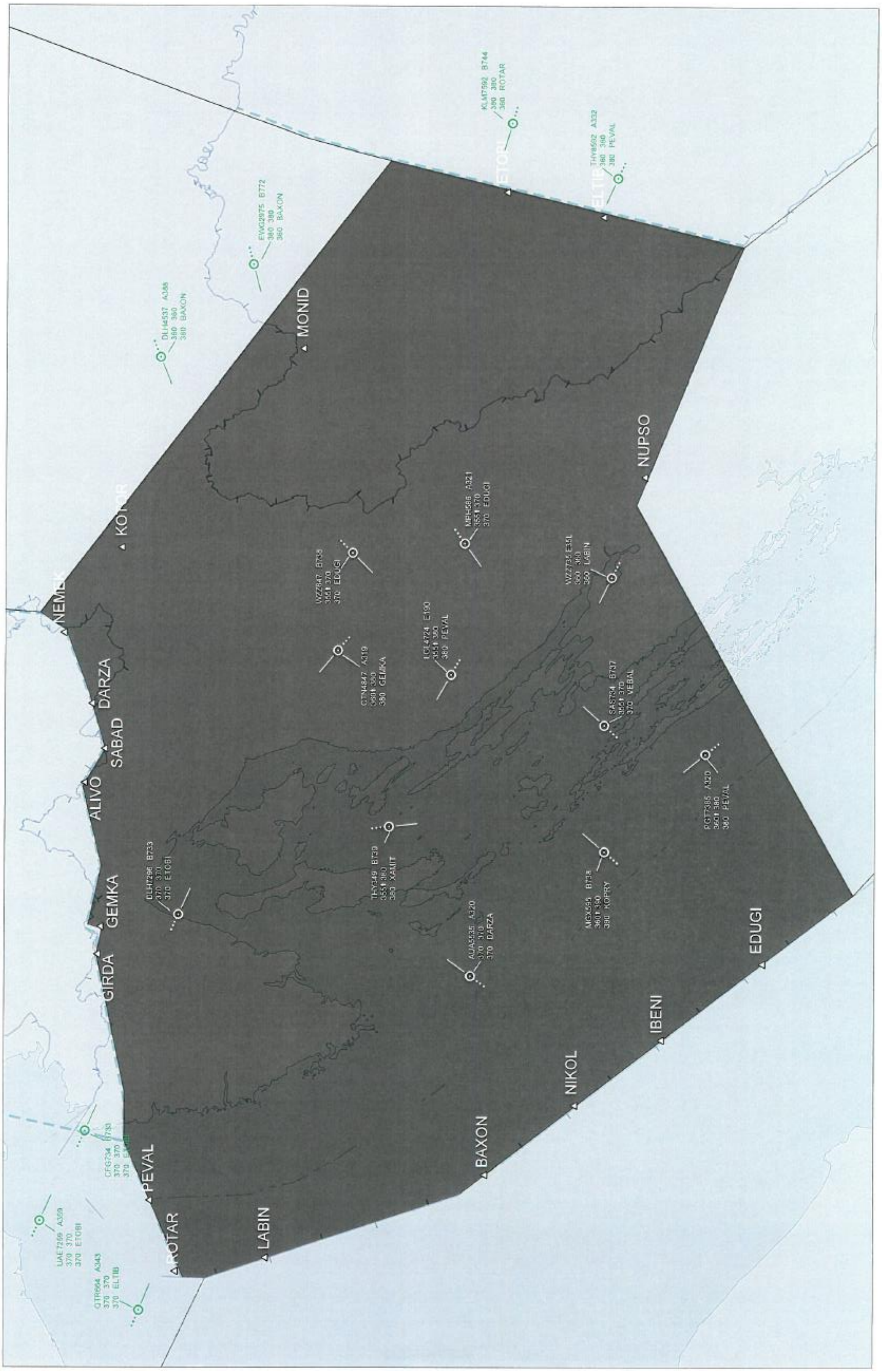


9647 W3 101 V22 311 J6 517

Traffic situation V20

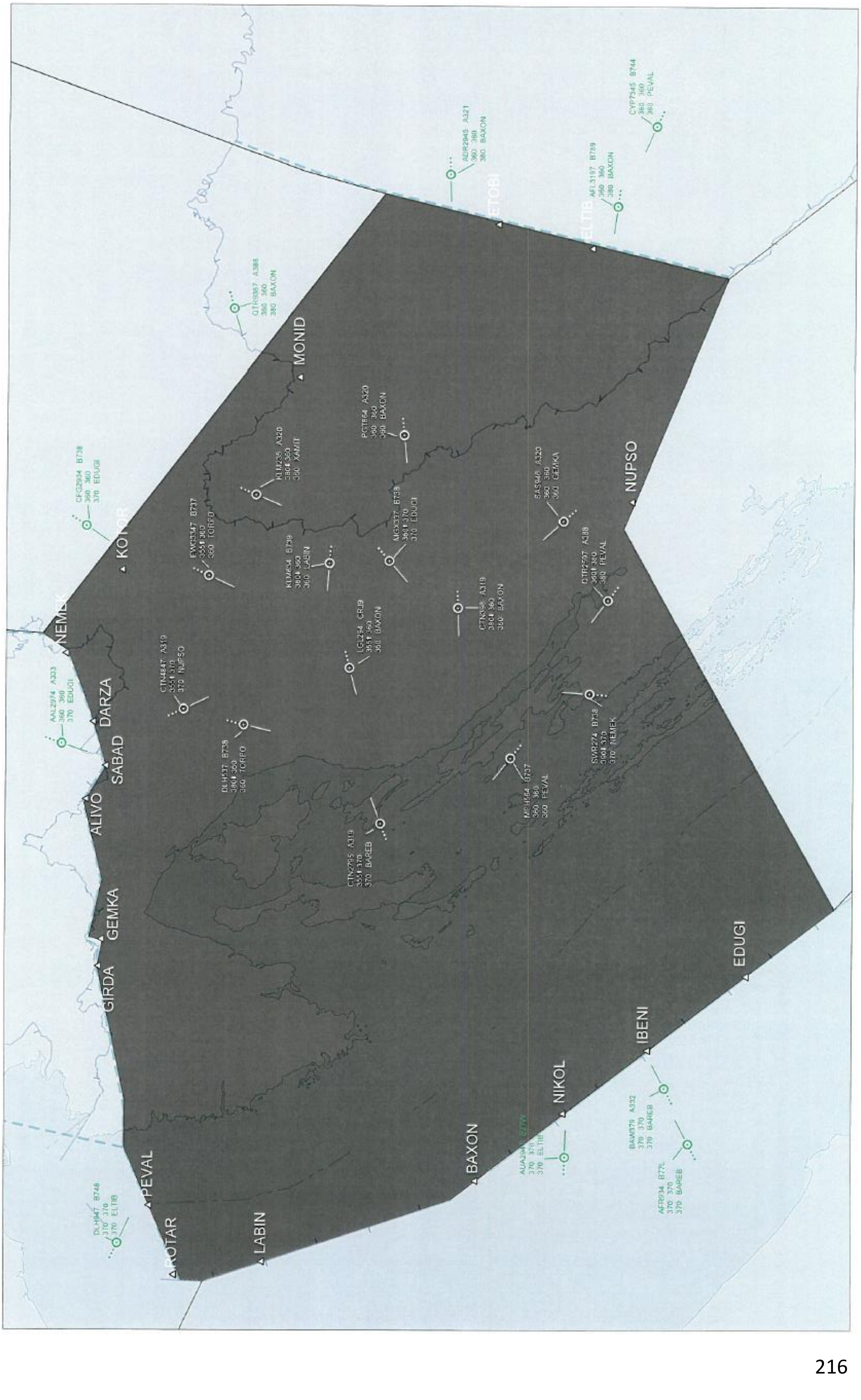


Traffic situation V21



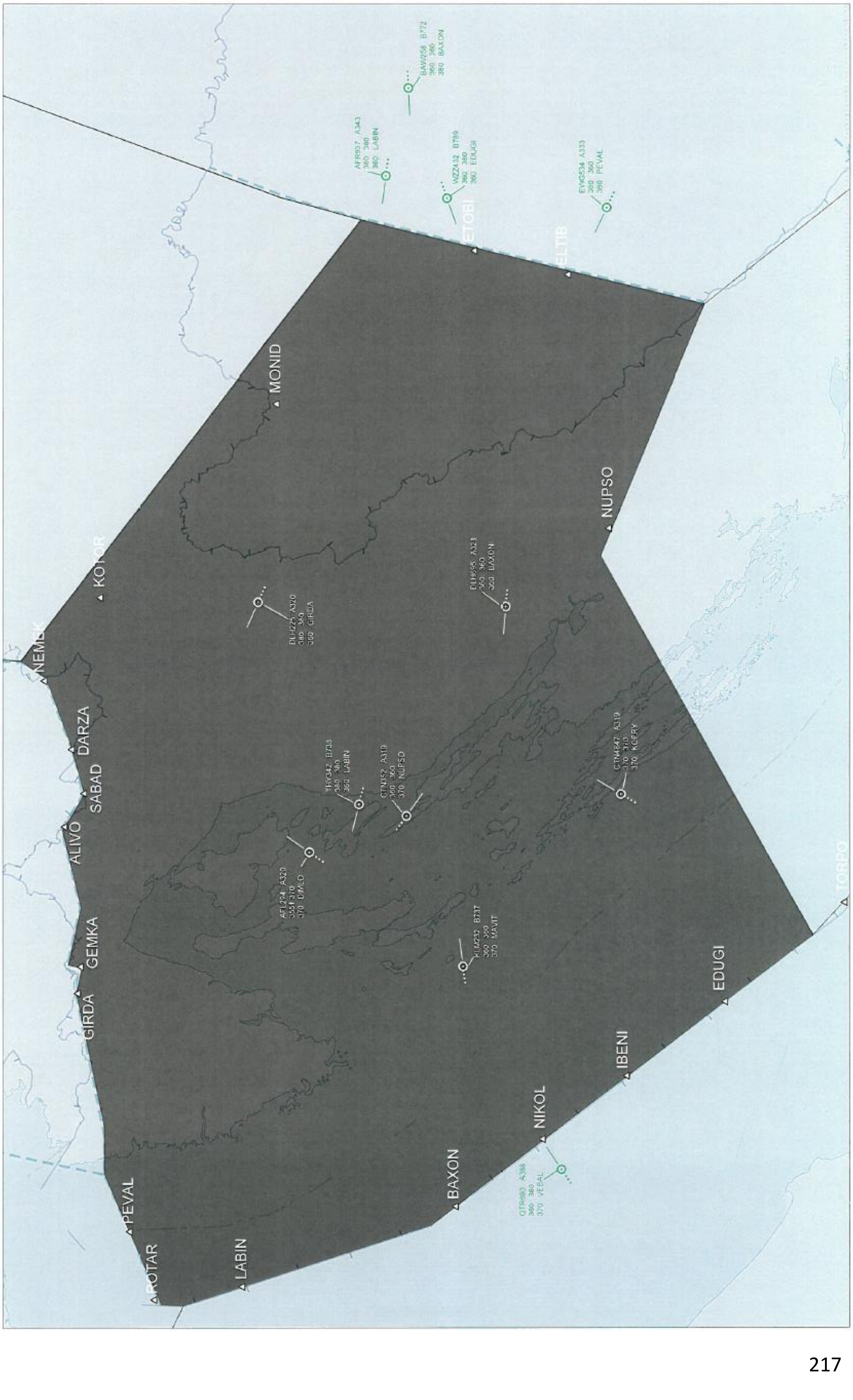
2291 X7 314 V24 294 K42 316

Traffic situation V22

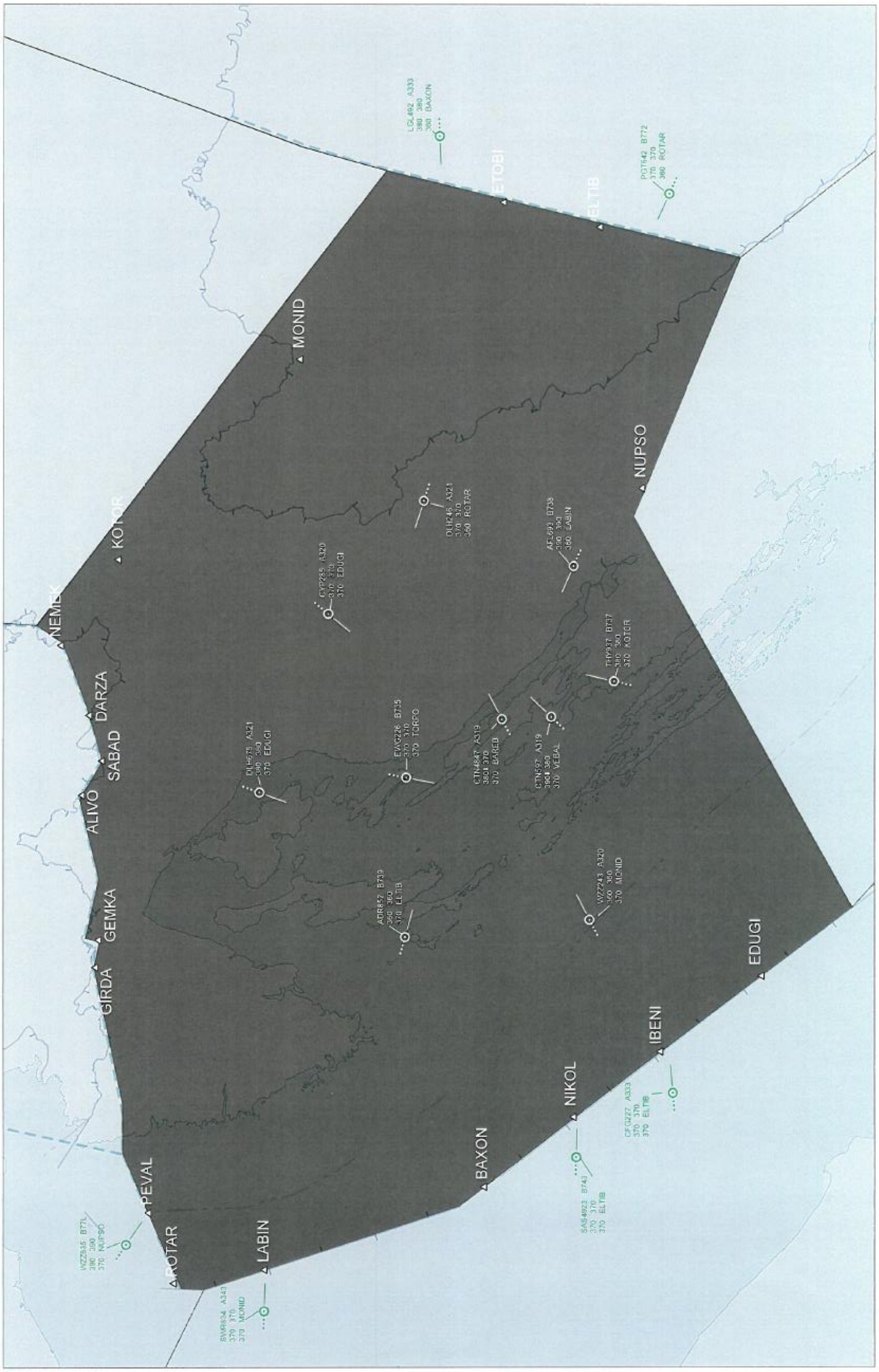


947 M3 111 V25 194 K33 549

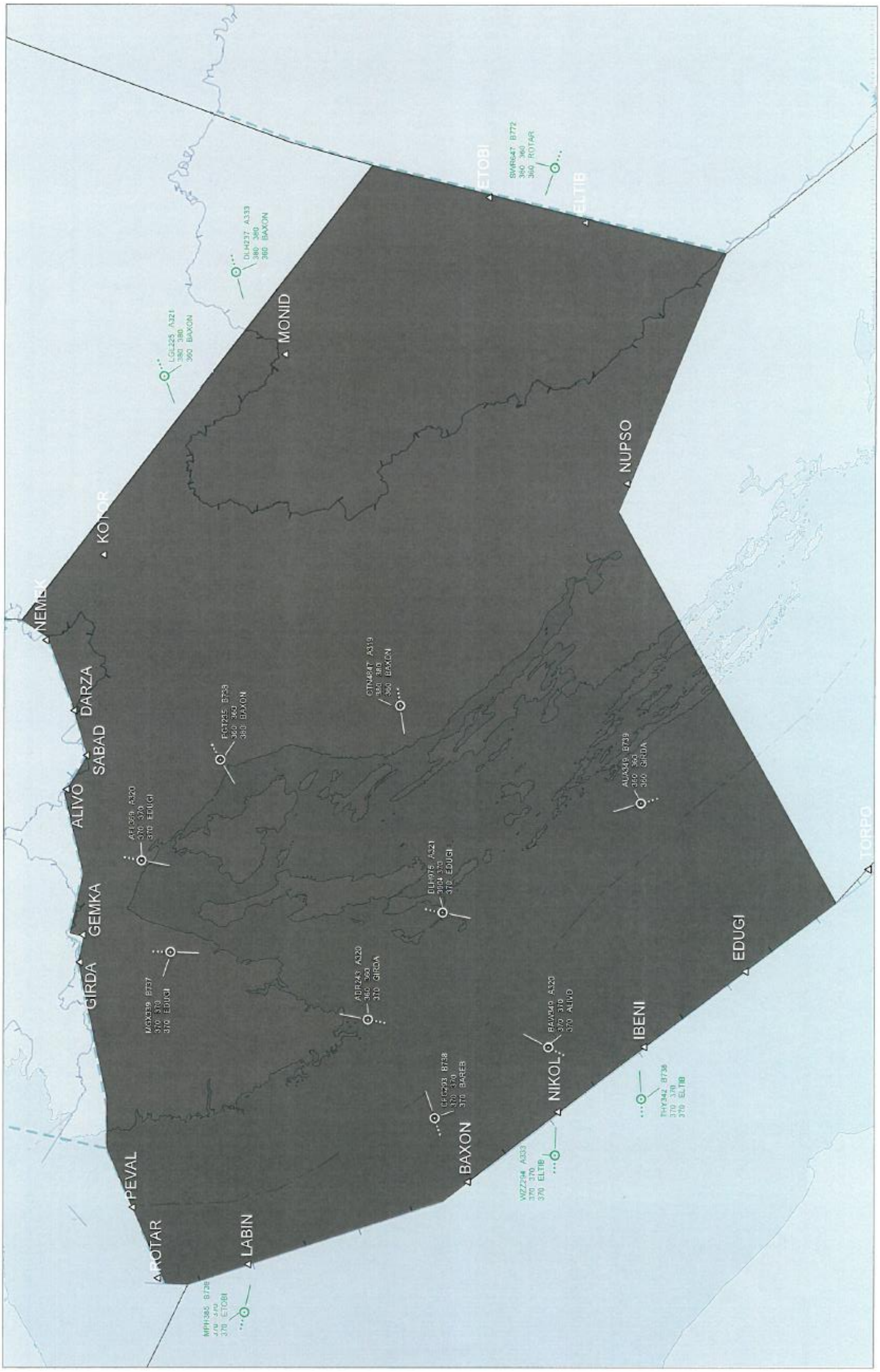
Traffic situation V23



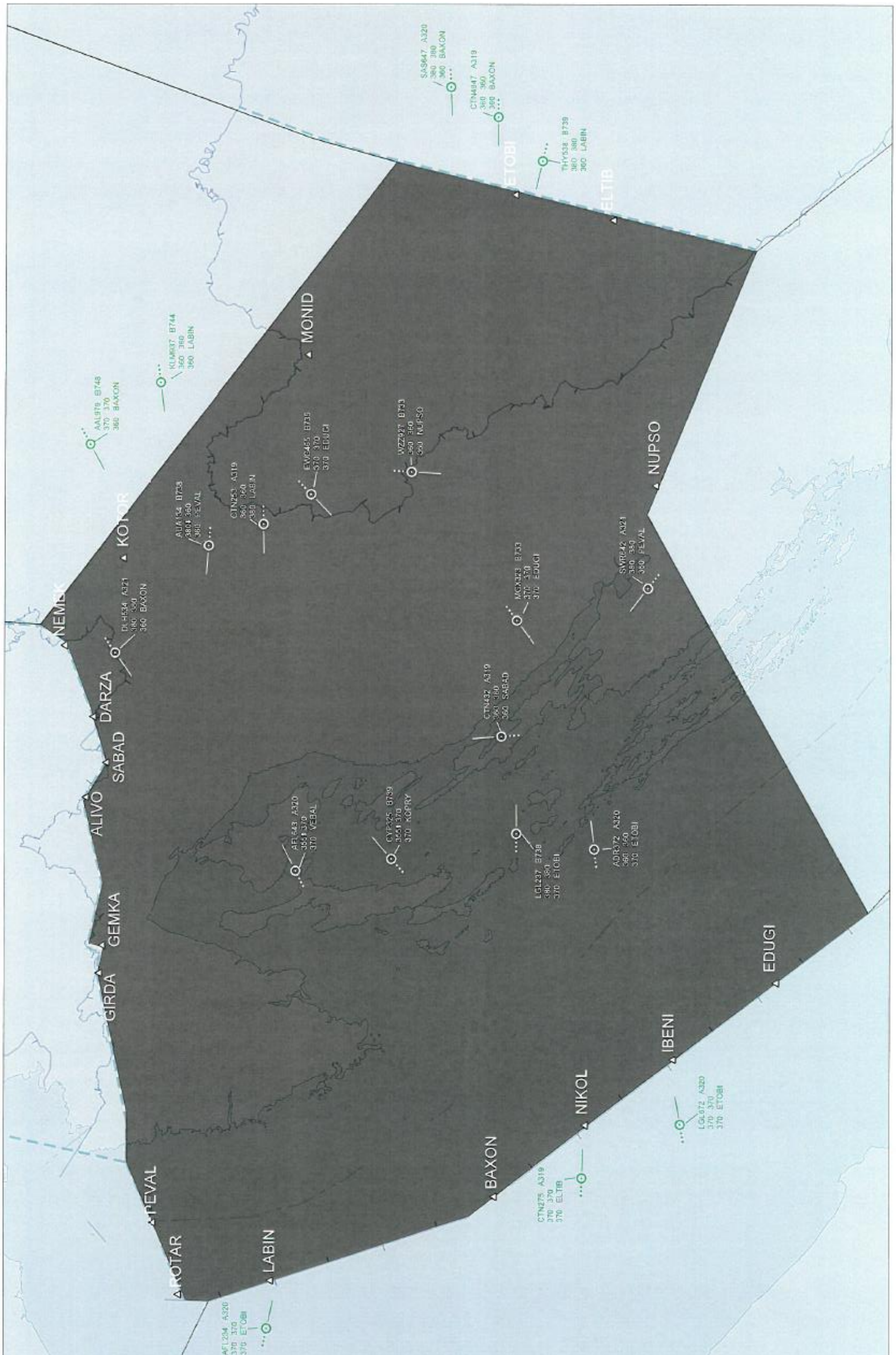
Traffic situation V24



Traffic situation V25

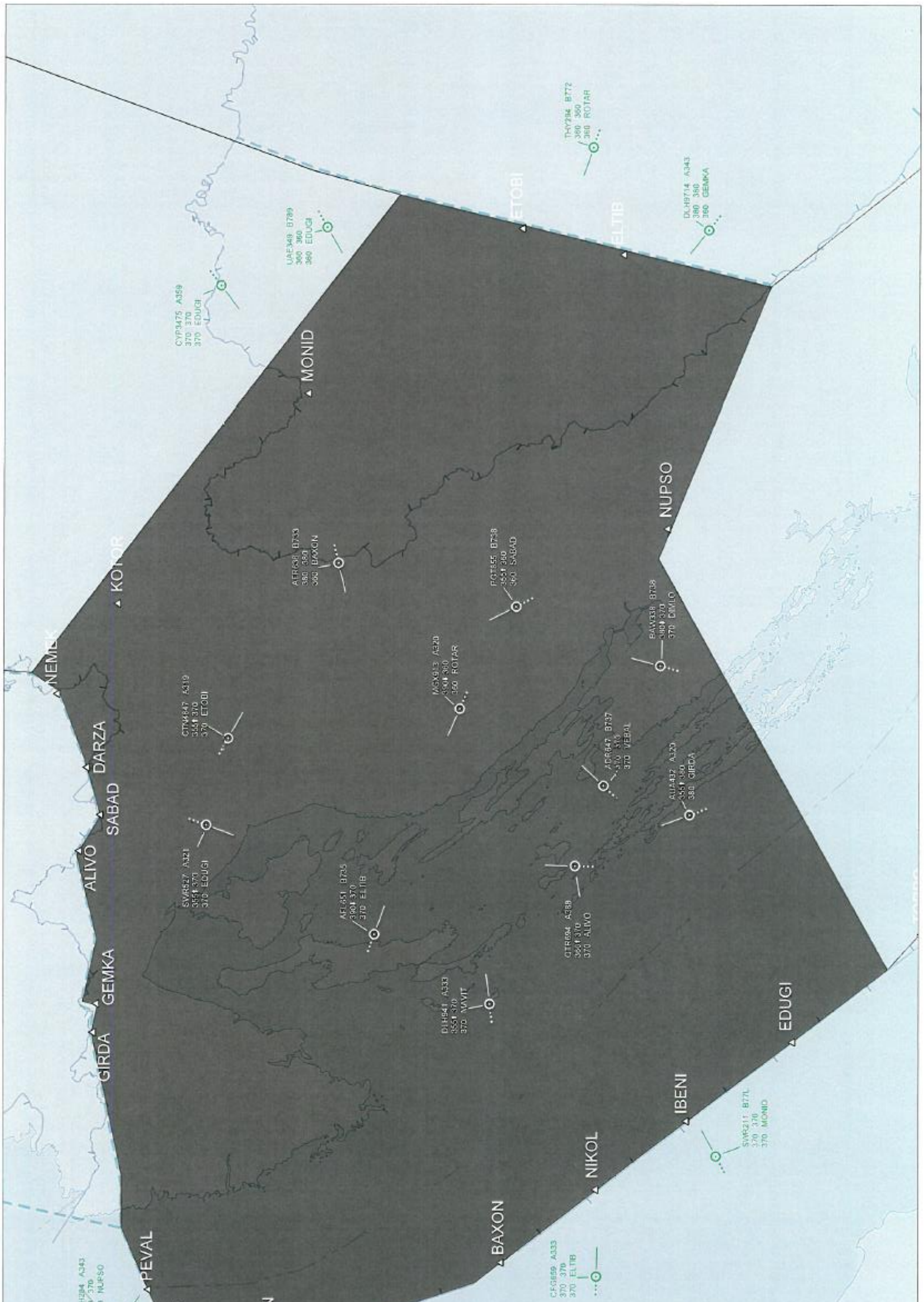


Traffic situation V26

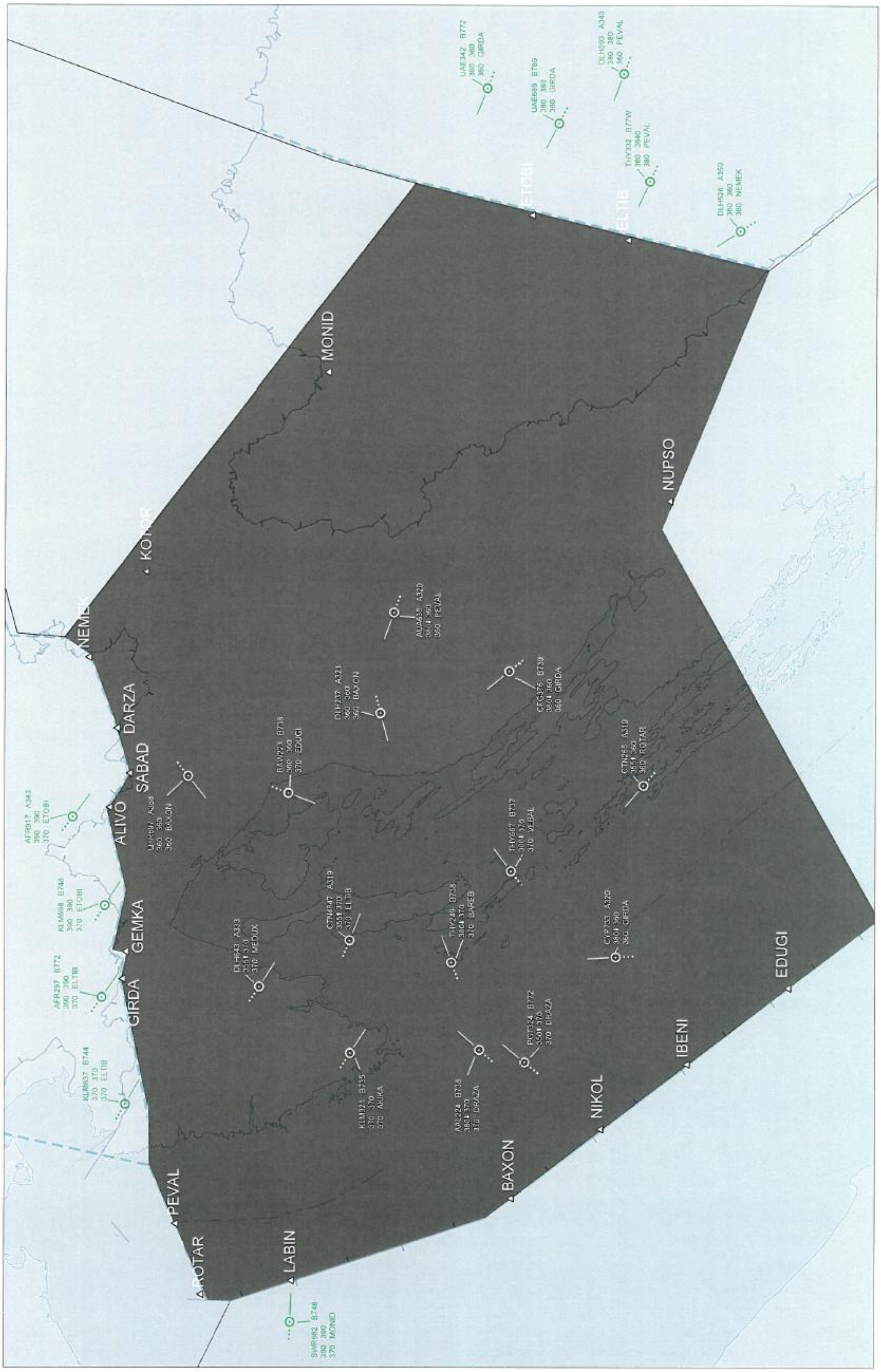


744 S3 373 V29 686 N78 589

Traffic situation V27



Traffic situation V28



Appendix 3 – Wolfram Mathematica code of ATCO tasks automatization

```
(*Import data of traffic situation *)
import =
  Import[
   "D:\\Downloads\\CloudFPZ\\OneDrive - Fakultet prometnih znanosti\\Doktorski studij\\Doktorska disertacija - PhD\\Experiments\\ATC
     Tasks\\Coordinates for traffic situations.xlsx", {"Data", Table[i, {i, 2, 121}]}];
data = {};
boundaries = {};

(*#1 row and #2 column from the loaded table. Boundaries loads from the airspace boundary table and stores them in coordinate pairs*)
Module[{i},
 For[i = 1, i ≤ Length[import], i++,
  AppendTo[data, Array[Which[#2 == 1, import[[i]][[2 #1, 2]],
      #2 == 2, import[[i]][[2 #1, 4]] / 8,
      #2 == 3, import[[i]][[2 #1 + 1, 4]] / 8,
      #2 == 4, import[[i]][[2 #1, 5]] °,
      #2 == 5, import[[i]][[2 #1, 6]],
      #2 == 6, import[[i]][[2 #1, 7]],
      #2 == 7, import[[i]][[2 #1, 8]],
      #2 == 8, import[[i]][[2 #1, 9]]] &, {(Length[import[[i]]] - 1)/2, 8}]
   ]
  ];
 For[i = 1, i ≤ Length[import], i++,
  AppendTo[boundaries, TakeList[import[[i]][[2 ;; 15, 15]], {2, 2, 2, 2, 2, 2}] / 8]
  ];]
r1 = 10;(*radius in NM for conflict within airspace*)
r2 = 15;(*radius in NM for conflict for outer airspace*)
vv = 10; (*ROC/ROD*)
(*ftypeofconflict input variables are the x and y coordinates of the aircraft,the direction of flight,speed,current FL,
cleared FL and exit FL for a pair of aircraft and the radius defined in the f2 function and the airspace coordinates. Function calculates conflicttime-
 conflict time,trueconflictpoint-true conflict coordinates with respect to horizontal and vertical distance and conflicttype *)
ftypeofconflict[{xt_, yt_, θt_, vt_, cuflt_, clflt_, eflt_}, {xt2_, yt2_, θt2_, vt2_, cuflt2_, clflt2_, eflt2_}, r_, b_] :=
 Module[{timetestout, timetestout2, timetestoutb, timetestout2b, heightt1, heightt2, heightt12, heightt22, tt1, tt2, conflictpointt, ttest,
   t, out, y, z, w, reg1, reg2, d, bb = RegionResize[Line[b], Scaled[1.5]][[1]], heighttc1, heighttc2, ctimes, upper, lower, i, differencetimetable,
   breachindex, conflicttime, trueconflictpoint, conflicttype},
  {tt1, tt2, conflictpointt} = fconflictpoint[{xt, yt, θt, vt}, {xt2, yt2, θt2, vt2}, r];
  If[Not[tt1 ∈ Reals], {conflicttime, trueconflictpoint, conflicttype} = {Null, {Null, Null}, Null},
   If[tt2 < 0, {conflicttime, trueconflictpoint, conflicttype} = {Null, {Null, Null}, Null},
    If[tt1 < 0, tt1 = 0];
    timetestout = fdistancetoboundary[{xt, yt, θt}, b] / vt;
    timetestout2 = fdistancetoboundary[{xt2, yt2, θt2}, b] / vt2;
    timetestoutb = fdistancetoboundary[{xt, yt, θt}, bb] / vt;
    timetestout2b = fdistancetoboundary[{xt2, yt2, θt2}, bb] / vt2;
    heighttc1 = { cuflt + Sign[clflt - cuflt] * 60 * t * vv   0 ≤ t <= Abs[clflt - cuflt] / 600
                { clflt                                       Abs[clflt - cuflt] / 600 ≤ t ;
    heighttc2 = { cuflt2 + Sign[clflt2 - cuflt2] * 60 * t * vv   0 ≤ t <= Abs[clflt2 - cuflt2] / 600
                { clflt2                                         Abs[clflt2 - cuflt2] / 600 ≤ t ;
   If[FindInstance[tt1 <= t ≤ Min[tt2, timetestoutb, timetestout2b] && Abs[heighttc1 - heighttc2] < 10, t] == {},
    {heightt1, heightt2} = If[timetestout - Abs[eflt - cuflt] / 600 < 0, {{ cuflt + Sign[eflt - cuflt] * 60 * t * vv   0 ≤ t <= Abs[eflt - cuflt] / 600
                                                                         { eflt                                       Abs[eflt - cuflt] / 600 ≤ t ,
      { cuflt + Sign[eflt - cuflt] * 60 * t * vv   0 ≤ t <= Abs[eflt - cuflt] / 600
      { eflt                                       Abs[eflt - cuflt] / 600 ≤ t        },
      {{ cuflt + Sign[clflt - cuflt] * 60 * t * vv                                   0 ≤ t <= Abs[clflt - cuflt] / 600
       { clflt                                                                       Abs[clflt - cuflt] / 600 ≤ t ≤ timetestout - Abs[eflt - clflt] / 600
       { clflt + Sign[eflt - clflt] * 60 * (t - (timetestout - Abs[eflt - clflt] / 600)) * vv   timetestout - Abs[eflt - clflt] / 600  ≤ t <= timetestout
       { eflt                                                                        timetestout ≤ t ,
      { cuflt + Sign[eflt - cuflt] * 60 * t * vv   0 ≤ t <= Abs[eflt - cuflt] / 600
      { eflt                                       Abs[eflt - cuflt] / 600 ≤ t        }];
    {heightt12, heightt22} = If[timetestout2 - Abs[eflt2 - cuflt2] / 600 < 0, {{ cuflt2 + Sign[eflt2 - cuflt2] * 60 * t * vv   0 ≤ t <= Abs[eflt2 - cuflt2] / 600
                                                                              { eflt2                                         Abs[eflt2 - cuflt2] / 600 ≤ t ,
      { cuflt2 + Sign[eflt2 - cuflt2] * 60 * t * vv   0 ≤ t <= Abs[eflt2 - cuflt2] / 600
      { eflt2                                         Abs[eflt2 - cuflt2] / 600 ≤ t        },
      {{ cuflt2 + Sign[clflt2 - cuflt2] * 60 * t * vv                                    0 ≤ t <= Abs[clflt2 - cuflt2] / 600
       { clflt2                                                                          Abs[clflt2 - cuflt2] / 600 ≤ t ≤ timetestout2 - Abs[eflt2 - clflt2] / 600
       { clflt2 + Sign[eflt2 - clflt2] * 60 * (t - (timetestout2 - Abs[eflt2 - clflt2] / 600)) * vv   timetestout2 - Abs[eflt2 - clflt2] / 600  ≤ t <= timetestout2
       { eflt2                                                                           timetestout2 ≤ t ,
      { cuflt2 + Sign[eflt2 - cuflt2] * 60 * t * vv   0 ≤ t <= Abs[eflt2 - cuflt2] / 600
      { eflt2                                         Abs[eflt2 - cuflt2] / 600 ≤ t        }];
    ctimes = {tt1, tt2, Abs[clflt - cuflt] / 600, timetestout - Abs[eflt - clflt] / 600 , Abs[eflt - cuflt] / 600, timetestout, Abs[eflt2 - cuflt2] / 600,
      timetestout2 - Abs[eflt2 - clflt2] / 600, Abs[eflt2 - cuflt2] / 600, timetestout2, timetestoutb, timetestout2b};
    ctimes = DeleteCases[ctimes, pt1_ /; (pt1 < tt1 || pt1 > Min[tt2, timetestoutb, timetestout2b])];
    ctimes = DeleteDuplicates[Sort[ctimes]];
    If[((Max[heightt1, heightt2] - Min[heightt12, heightt22]) /. t → tt1) ≥ 0,
     {upper = Min[heightt1, heightt2];
      lower = Max[heightt12, heightt22];},
     {upper = Min[heightt12, heightt22];
      lower = Max[heightt1, heightt2];}
    ];
```

```
      differencetimetable = Table[upper /. t → ctimes[[i]], {i, 1, Length[ctimes]}] - Table[lower /. t → ctimes[[i]], {i, 1, Length[ctimes]}];
      breachindex = FirstPosition[differencetimetable, i_ /; i < 10];
      If[breachindex === Missing["NotFound"],
       {conflicttime = Null;
        trueconflictpoint = {Null, Null};
        conflicttype = Null},
       If[breachindex[[1]] == 1,
         {conflicttime = tt1;
          trueconflictpoint = ({xt + conflicttime vt Cos[θt], yt + conflicttime vt Sin[θt]} + {xt2 + conflicttime vt2 Cos[θt2], yt2 + conflicttime vt2 Sin[θt2]}) / 2;
         },
         {conflicttime =
           (t /. Quiet[Solve[{t, i} ∈ InfiniteLine[{{ctimes[[breachindex[[1]] - 1]], (upper - 10) /. t -> ctimes[[breachindex[[1]] - 1]]},
                  {ctimes[[breachindex[[1]]]], (upper - 10) /. t -> ctimes[[breachindex[[1]]]]}}] &&
                {t, i} ∈ InfiniteLine[{{ctimes[[breachindex[[1]] - 1]], lower /. t -> ctimes[[breachindex[[1]] - 1]]},
                  {ctimes[[breachindex[[1]]]], lower /. t -> ctimes[[breachindex[[1]]]]}}], {t, i}]]) [[1]];
          trueconflictpoint = ({xt + conflicttime vt Cos[θt], yt + conflicttime vt Sin[θt]} + {xt2 + conflicttime vt2 Cos[θt2], yt2 + conflicttime vt2 Sin[θt2]}) / 2;
         }
        ];
      ];
      If[conflicttime === Null, conflicttype = Null, Which[r == r1, conflicttype = "potential conflict", r == r2, conflicttype = "potential coordination conflict"]],
      Which[r == r1, conflicttype = "conflict", r == r2, conflicttype = "coordination conflict"];
      conflicttime = Quiet[Minimize[{t, t ≥ tt1 && Reduce[Abs[heighttc1 - heighttc2] < 10, t]}, {t}][[1]]];
      trueconflictpoint = ({xt + conflicttime vt Cos[θt], yt + conflicttime vt Sin[θt]} + {xt2 + conflicttime vt2 Cos[θt2], yt2 + conflicttime vt2 Sin[θt2]}) / 2;
     ];
    ];
   ];
   {conflicttime, trueconflictpoint, conflicttype}
  ];

(*fdistance function calculates the horizontal distance between pairs of aircraft*)
fdistance[{x1_, y1_}, {x2_, y2_}] := EuclideanDistance[{x1, y1}, {x2, y2}]

(*fintesectionangle function calculates the angle of convergence to the point of conflict*)
fintersectionangle[θ1_, θ2_] := If[Abs[θ2 - θ1] > 180 °, 360 ° - Abs[θ2 - θ1], Abs[θ2 - θ1]]

(*calculates the point where the directions of movement of the aircraft intersect but only the horizontal plane*)
fintersectionpoint[{x1_, y1_, θ1_}, {x2_, y2_, θ2_}] := Flatten [{x, y} /. Solve[{(y - y1) Cos[θ1] == (x - x1) Sin[θ1], (y - y2) Cos[θ2] == (x - x2) Sin[θ2]}, {x, y}]]

(*tests whether they are within a smaller airspace*)
finsidetest[{x1_, y1_}, {x2_, y2_}, b_] := Module[{reg = Polygon[b], t1, t2},
  t1 = ({x1, y1} ∈ reg);
  t2 = ({x2, y2} ∈ reg);
  t1 && t2
 ]

(*tests whether they are within a larger airspace*)
finsidetest2[{x1_, y1_}, b_] := Module[{reg = RegionResize[Polygon[b], Scaled[1.5]]},
  {x1, y1} ∈ reg
 ]
(*calculates the conflict point where the separation of the aircraft is violated but only on horizontal plane*)
fconflictpoint[{x1_, y1_, θ1_, v1_}, {x2_, y2_, θ2_, v2_}, r_] := Module[{t1, t2, t3, sol, out, t},
  sol = Solve[(x1 - x2 + t (v1 Cos[θ1] - v2 Cos[θ2]))^2 + (y1 - y2 + t (v1 Sin[θ1] - v2 Sin[θ2]))^2 == r^2, t];
  If[sol ≠ {},
   t1 = t /. sol[[1]];
   t2 = t /. sol[[2]];
   t3 = Which[Not[t1 ∈ Reals] || (t1 < 0 && t2 < 0), ∞, 0 ≤ t1 ≤ t2, t1, t1 < 0 ≤ t2, 0];
   out = {t1, t2, If[t3 ≠ ∞, ({x1 + t3 v1 Cos[θ1], y1 + t3 v1 Sin[θ1]} + {x2 + t3 v2 Cos[θ2], y2 + t3 v2 Sin[θ2]}) / 2, {Null, Null}]};,
   If[fdistance[{x1, y1}, {x2, y2}] >= r, out = {i, i, {Null, Null}}, out = {0, 2 / 3, ({x1, y1} + {x2, y2}) / 2}]];

  out
 ]

(*calculates the distance of the aircraft from the further border*)
fdistancetoboundary[{x_, y_, θ_}, b_] := Module[{b2 = Append[b, b[[1]]], reg1, reg2, exitpoint},
  reg1 = Line[b2];
  reg2 = HalfLine[{x, y}, {Cos[θ], Sin[θ]}];
  exitpoint = RegionIntersection[reg1, reg2];
  Max[EuclideanDistance[{x, y}, #] & /@ exitpoint[[1]]]
 ]
(*calculates the distance of the aircraft to the nearest border*)
fdistanceclosertoboundary[{x_, y_, θ_}, b_] := Module[{b2 = Append[b, b[[1]]], reg1, reg2, exitpoint},
  reg1 = Line[b2];
  reg2 = HalfLine[{x, y}, {Cos[θ], Sin[θ]}];
  exitpoint = RegionIntersection[reg1, reg2];
  Min[EuclideanDistance[{x, y}, #] & /@ exitpoint[[1]]]
 ]

(*calculates the speed of the wake vortex turbulence, who is faster and who is slower in relation to the distance from the conflict point*)
fwtc[v1_, v2_, dcp1_, dcp2_] := Module[{},
  If[dcp1 < dcp2, Which[(v1 == 520 && v2 < 520) || (v1 ≥ 470 && v2 ≤ 460), "faster",
    (429 ≤ v1 ≤ 460 && 429 ≤ v2 ≤ 460) || (470 ≤ v1 ≤ 510 && 470 ≤ v2 ≤ 510) || (520 == v1 && v2 == 520), "same", (429 ≤ v1 ≤ 460 && 470 ≤ v2) || (470 ≤ v1 ≤ 510 && v2 == 520),
    "slower"], Which[(429 ≤ v1 ≤ 460 && 470 ≤ v2) || (470 ≤ v1 ≤ 510 && v2 == 520), "faster",
    (429 ≤ v1 ≤ 460 && 429 ≤ v2 ≤ 460) || (470 ≤ v1 ≤ 510 && 470 ≤ v2 ≤ 510) || (520 == v1 && v2 == 520), "same", (v1 == 520 && v2 < 520) || (v1 ≥ 470 && v2 ≤ 460), "slower"]]
 ]
```

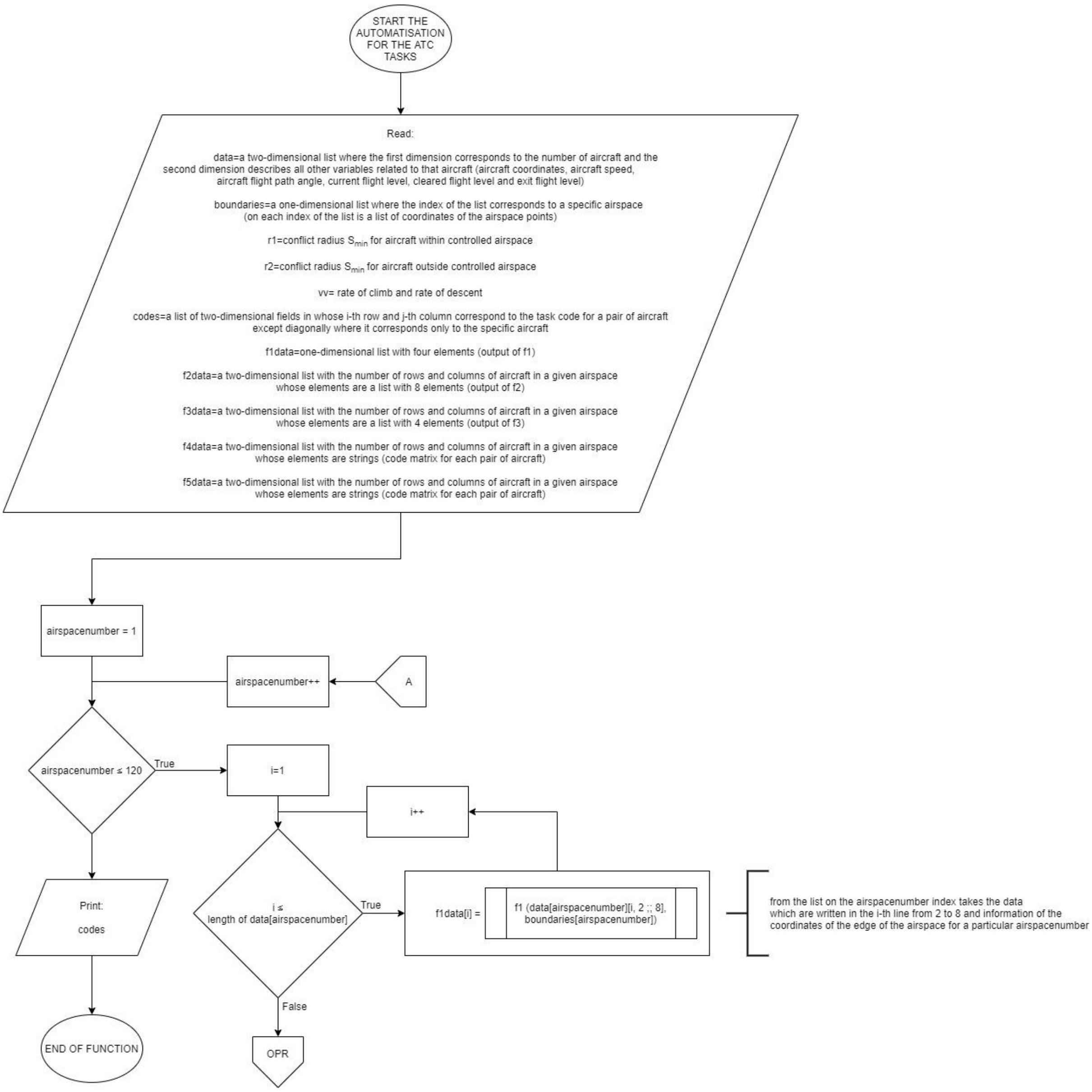
```
(*calculates thorizontalfreecw-whether the aircraft is free clockwise in degrees 5-45 in steps of 5 degrees and thorizontalfreeccw-
 counterclockwise with the same degrees*)
fthorizontalfree[{x1_, y1_, θ1_, v1_, cuflt_, clflt_}, {x2_, y2_, θ2_, v2_, cuflt2_, clflt2_}, b_] :=
 Module[{t, heighttc1, heighttc2, differencetimetablecw, differencetimetableccw, dθ, tvc1, tvc2, thorizontalfreecw, thorizontalfreeccw,
   r = If[finsidetest[{x1, y1}, {x2, y2}, b], r1, r2]},
  heighttc1 = { cuflt + Sign[clflt - cuflt] * 60 * t * vv   0 ≤ t <= Abs[clflt - cuflt] / 600
              { clflt                                      Abs[clflt - cuflt] / 600 ≤ t          ;
  heighttc2 = { cuflt2 + Sign[clflt2 - cuflt2] * 60 * t * vv   0 ≤ t <= Abs[clflt2 - cuflt2] / 600
              { clflt2                                        Abs[clflt2 - cuflt2] / 600 ≤ t     ;
  tvc1 = Max[Quiet[Minimize[{t, t ≥ 0 && Reduce[Abs[heighttc1 - heighttc2] < 10, t]}, {t}][[1]]]];
  tvc2 = Min[Quiet[Maximize[{t, t ≥ 0 && Reduce[Abs[heighttc1 - heighttc2] < 10, t]}, {t}][[1]]], 1/3];
  differencetimetablecw = Table[If[fconflictpoint[{x1, y1, dθ + θ1, v1}, {x2, y2, θ2, v2}, r][[3]] === {Null, Null}, -1,
     (Min[1/(v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) (-v1 x1 Cos[dθ + θ1] + v1 x2 Cos[dθ + θ1] + v2 x1 Cos[θ2] - v2 x2 Cos[θ2] - v1 y1 Sin[dθ + θ1] + v1 y2 Sin[dθ + θ1] +
          v2 y1 Sin[θ2] - v2 y2 Sin[θ2] + √((r^2 - (x1 - x2)^2 - (y1 - y2)^2) (v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) +
             (v1 (x1 - x2) Cos[dθ + θ1] + v2 (-x1 + x2) Cos[θ2] + (y1 - y2) (v1 Sin[dθ + θ1] - v2 Sin[θ2]))^2)), tvc2] -
       Max[-1/(v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) (v1 x1 Cos[dθ + θ1] - v1 x2 Cos[dθ + θ1] - v2 x1 Cos[θ2] + v2 x2 Cos[θ2] + v1 y1 Sin[dθ + θ1] -
          v1 y2 Sin[dθ + θ1] - v2 y1 Sin[θ2] + v2 y2 Sin[θ2] +
          √((r^2 - (x1 - x2)^2 - (y1 - y2)^2) (v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) + (v1 (x1 - x2) Cos[dθ + θ1] + v2 (-x1 + x2) Cos[θ2] +
               (y1 - y2) (v1 Sin[dθ + θ1] - v2 Sin[θ2]))^2)), tvc1])], {dθ, Table[i, {i, -45 °, -5 °, 5 °}]}];
  differencetimetableccw = Table[If[fconflictpoint[{x1, y1, dθ + θ1, v1}, {x2, y2, θ2, v2}, r][[3]] === {Null, Null}, -1,
     (Min[1/(v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) (-v1 x1 Cos[dθ + θ1] + v1 x2 Cos[dθ + θ1] + v2 x1 Cos[θ2] - v2 x2 Cos[θ2] - v1 y1 Sin[dθ + θ1] + v1 y2 Sin[dθ + θ1] +
          v2 y1 Sin[θ2] - v2 y2 Sin[θ2] + √((r^2 - (x1 - x2)^2 - (y1 - y2)^2) (v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) +
             (v1 (x1 - x2) Cos[dθ + θ1] + v2 (-x1 + x2) Cos[θ2] + (y1 - y2) (v1 Sin[dθ + θ1] - v2 Sin[θ2]))^2)), tvc2] -
       Max[-1/(v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) (v1 x1 Cos[dθ + θ1] - v1 x2 Cos[dθ + θ1] - v2 x1 Cos[θ2] + v2 x2 Cos[θ2] + v1 y1 Sin[dθ + θ1] -
          v1 y2 Sin[dθ + θ1] - v2 y1 Sin[θ2] + v2 y2 Sin[θ2] +
          √((r^2 - (x1 - x2)^2 - (y1 - y2)^2) (v1^2 + v2^2 - 2 v1 v2 Cos[dθ + θ1 - θ2]) + (v1 (x1 - x2) Cos[dθ + θ1] + v2 (-x1 + x2) Cos[θ2] +
               (y1 - y2) (v1 Sin[dθ + θ1] - v2 Sin[θ2]))^2)), tvc1])], {dθ, Table[i, {i, 5 °, 45 °, 5 °}]}];
  thorizontalfreecw = If[FirstCase[differencetimetablecw, x_ /; x > 0] === Missing["NotFound"], True, False];
  thorizontalfreeccw = If[FirstCase[differencetimetableccw, x_ /; x > 0] === Missing["NotFound"], True, False];
  {thorizontalfreecw, thorizontalfreeccw}
 ];
(*calculates whether there is vertical and horizontal separation violation for a pair of aircraft*)
ftverticalfree[{xt_, yt_, θt_, vt_, cuflt_, clflt_}, {xt2_, yt2_, θt2_, vt2_, cuflt2_, clflt2_}, b_] :=
 Module[{timetestoutb, timetestout2b, tt1, tt2, conflictpointt, ttest, t, bb = RegionResize[Line[b], Scaled[1.5]][[1]], heighttc1, heighttc2,
   r = If[finsidetest[{xt, yt}, {xt2, yt2}, b], r1, r2]},
  {tt1, tt2, conflictpointt} = fconflictpoint[{xt, yt, θt, vt}, {xt2, yt2, θt2, vt2}, r];
  If[Not[tt1 ∈ Reals], ttest = True,
   If[tt2 < 0, ttest = True,
     If[tt1 < 0, tt1 = 0];
     timetestoutb = fdistancetoboundary[{xt, yt, θt}, bb] / vt;
     timetestout2b = fdistancetoboundary[{xt2, yt2, θt2}, bb] / vt2;
     heighttc1 = { cuflt + Sign[clflt - cuflt] * 60 * t * vv   0 ≤ t <= Abs[clflt - cuflt] / 600
                 { clflt                                      Abs[clflt - cuflt] / 600 ≤ t          ;
     heighttc2 = { cuflt2 + Sign[clflt2 - cuflt2] * 60 * t * vv   0 ≤ t <= Abs[clflt2 - cuflt2] / 600
                 { clflt2                                        Abs[clflt2 - cuflt2] / 600 ≤ t     ;
     If[FindInstance[tt1 <= t ≤ Min[tt2, timetestoutb, timetestout2b] && Abs[heighttc1 - heighttc2] < 10, t] == {}, ttest = True, ttest = False];
    ];
  ];
  ttest
 ]
(*returns boundarydistance,er,ft,ic*)
f1[info1_, boundary_] :=
  Module[{x01 = info1[[1]], y01 = info1[[2]], θ1 = info1[[3]], v1 = info1[[4]], cufl1 = info1[[5]], clfl1 = info1[[6]], efl1 = info1[[7]],
    b = boundary, boundarydistance, er, ft, ic},
   boundarydistance = fdistancetoboundary[{x01, y01, θ1}, b];
   er = If[clfl1 == efl1, Null, "ER"];
   ft = If[boundarydistance <= 15, "FT", Null];
   ic = If[{x01, y01} ∈ Polygon[b], Null,
     If[fdistanceclosertoboundary[{x01, y01, θ1}, b] ≤ 20, "IC"]];
   {boundarydistance, er, ft, ic}
  ];
```

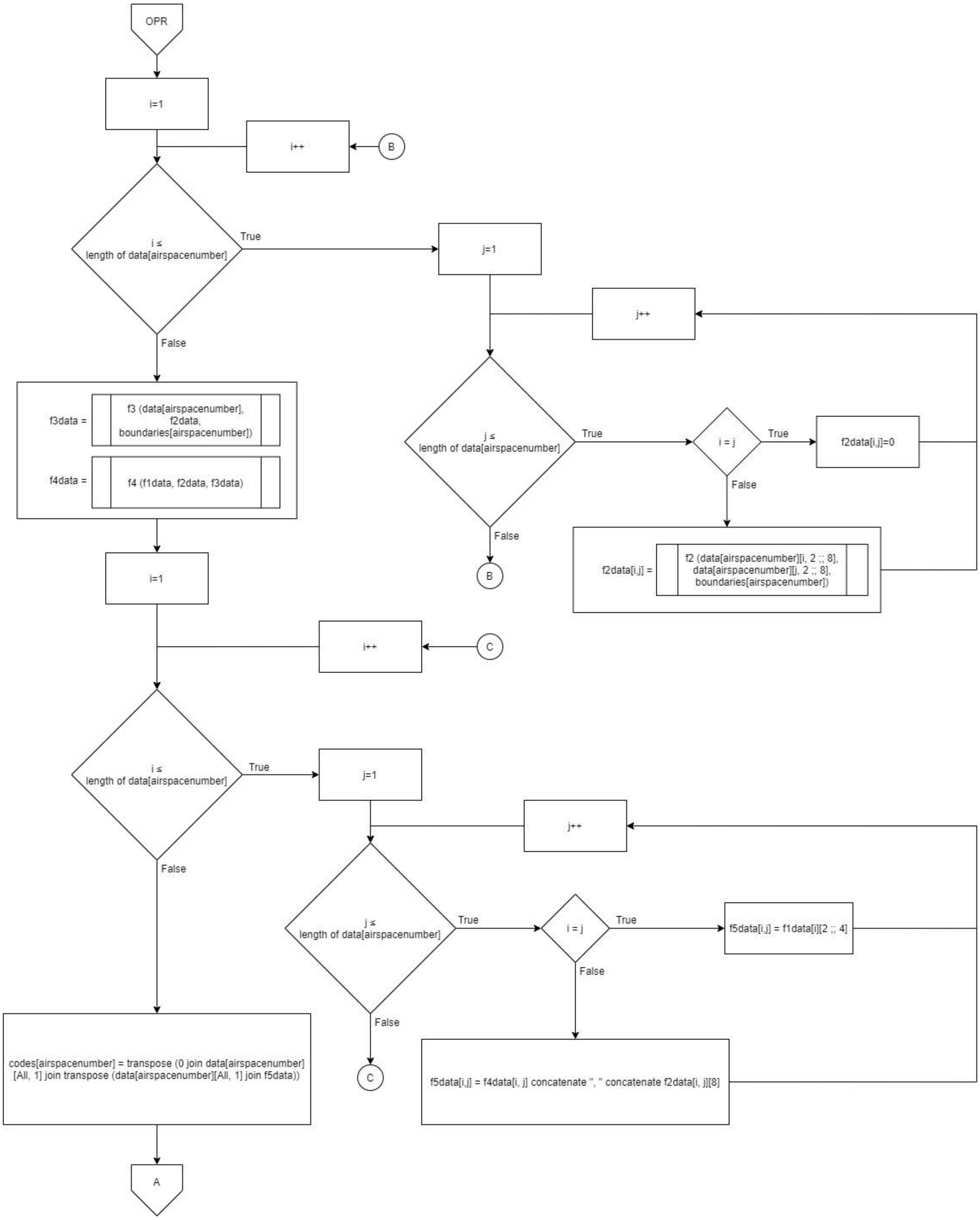
```
(*calculates only if there is a conflict within the outer airspace the following parameters d0,convergingangle,dcp1,dcp2,wtc,conflicttrack,typeofconflict,cs*)
f2[info1_, info2_, boundary_] :=
 Module[{x01 = info1[[1]], y01 = info1[[2]], θ1 = info1[[3]], v1 = info1[[4]], cufl1 = info1[[5]], clfl1 = info1[[6]], efl1 = info1[[7]],
   x02 = info2[[1]], y02 = info2[[2]], θ2 = info2[[3]], v2 = info2[[4]], cufl2 = info2[[5]], clfl2 = info2[[6]], efl2 = info2[[7]],
   b = boundary, d0, convergingangle, conflicttrack, intersectionangle, intersectionpoint, r, conflictpoint, conflictpointtest, dcp1, dcp2,
   wtc, t1, t2, typeofconflict, conflicttime, cs},
  r = If[finsidetest[{x01, y01}, {x02, y02}, b], r1, r2];
  {conflicttime, conflictpoint, typeofconflict} = ftypeofconflict[{x01, y01, θ1, v1, cufl1, clfl1, efl1}, {x02, y02, θ2, v2, cufl2, clfl2, efl2}, r, b];
  conflictpointtest = finsidetest2[conflictpoint, b];
  If[conflictpointtest == False, typeofconflict = Null];
  d0 = If[conflictpoint =!= {Null, Null} && conflictpointtest, fdistance[{x01, y01}, {x02, y02}]];
  dcp1 = If[conflictpoint =!= {Null, Null} && conflictpointtest, EuclideanDistance[conflictpoint, {x01, y01}], Null];
  dcp2 = If[conflictpoint =!= {Null, Null} && conflictpointtest, EuclideanDistance[conflictpoint, {x02, y02}], Null];
  convergingangle = If[intersectionpoint =!= {x, y} && conflictpoint =!= {Null, Null} && conflictpointtest == True,
    intersectionpoint = Flatten[{x, y} /. Quiet[Solve[{x, y} ∈ HalfLine[{x01, y01}, {Cos[θ1], Sin[θ1]}] && {x, y} ∈ HalfLine[{x02, y02}, {Cos[θ2], Sin[θ2]}]]]];
    intersectionangle = fintersectionangle[θ1, θ2];
    intersectionangle, Null];
  conflicttrack = If[conflictpoint =!= {Null, Null} && conflictpointtest,
    Which[0 <= intersectionangle < 45 °, "same track", 45 ° <= intersectionangle ≤ 135 °, "crossing track", 135 ° < intersectionangle ≤ 180 °, "opposite track"], Null];
  wtc = If[conflictpoint =!= {Null, Null} && conflictpointtest, fwtc[v1, v2, dcp1, dcp2]];
  cs = Which[typeofconflict === Null, "", typeofconflict == "coordination conflict" || typeofconflict == "conflict", "SI",
    typeofconflict == "potential coordination conflict" || typeofconflict == "potential conflict", "SP"];

  {d0, convergingangle, dcp1, dcp2, wtc, conflicttrack, typeofconflict, cs}
 ]
(*calculate whether the aircraft is free horizontally and vertically; tcwfree,tccwfree,tupfree,tdownfree*)
f3[dataall_, infoall_, boundary_] :=
 Module[{noofdatapts = Length[dataall], thorizontalfree, tverticalfree, i, j, k, l, tcwfree, tccwfree, tempout, hfreematrix, tupfree, tdownfree,
   tempupclfl, tempdownclfl},
  hfreematrix = Table[Null, {l, 1, noofdatapts}, {m, 1, noofdatapts}];
  For[i = 1, i <= noofdatapts, i++; l = True,
   For[If[i == 1, j = 2, j = 1], j <= noofdatapts, Which[j + 1 ≠ i, j++, j + 1 == i && j + 2 <= noofdatapts, j = j + 2, True, Break[]],
    If[infoall[[i, j]] =!= {Null, Null, Null, Null, Null, Null, Null},
     {tcwfree, tccwfree} = {True, True};
     thorizontalfree = True;
     For[If[i == 1, k = 2, k = 1], k ≤ noofdatapts && thorizontalfree == True, Which[k + 1 ≠ i, k++, k + 1 == i && k + 2 <= noofdatapts, k = k + 2, True, Break[]],
      tempout = fthorizontalfree[dataall[[i, 2 ;; 7]], dataall[[k, 2 ;; 7]], boundary];
      If[tempout[[1]] == False, tcwfree = False];
      If[tempout[[2]] == False, tccwfree = False];
      If[{tcwfree, tccwfree} == {False, False}, thorizontalfree = False];
     ];
     If[l,
      {tupfree, tdownfree} = {False, False};
      For[tempupclfl = Max[dataall[[i, 6]], dataall[[i, 7]]], tempupclfl ≤ 400 && tupfree == False, tempupclfl = tempupclfl + 10,
       tverticalfree = True;
       For[If[i == 1, k = 2, k = 1], k ≤ noofdatapts && tverticalfree == True, Which[k + 1 ≠ i, k++, k + 1 == i && k + 2 <= noofdatapts, k = k + 2, True, Break[]],
        tempout = ftverticalfree[ReplacePart[dataall[[i, 2 ;; 7]], 6 -> tempupclfl], dataall[[k, 2 ;; 7]], boundary];
        If[tempout == False, tverticalfree = False];
       ];
       If[tverticalfree == True, tupfree = True]
      ];
      For[tempupclfl = Min[dataall[[i, 6]], dataall[[i, 7]]], tempupclfl ≥ 300 && tdownfree == False, tempupclfl = tempupclfl - 10,
       tverticalfree = True;
       For[If[i == 1, k = 2, k = 1], k ≤ noofdatapts && tverticalfree == True, Which[k + 1 ≠ i, k++, k + 1 == i && k + 2 <= noofdatapts, k = k + 2, True, Break[]],
        tempout = ftverticalfree[ReplacePart[dataall[[i, 2 ;; 7]], 6 -> tempupclfl], dataall[[k, 2 ;; 7]], boundary];
        If[tempout == False, tverticalfree = False];
       ];
       If[tverticalfree == True, tdownfree = True]
      ];
      l = False;
     ];
     hfreematrix[[i, j]] = {tcwfree, tccwfree, tupfree, tdownfree};
    ];
   ]
  ];
  hfreematrix
 ]
(*calculates a matrix printout for each pair of aircraft*)
f4[f1data_, f2data_, f3data_] := Module[{noofdatapts = Length[f1data], rawdata, data},
   rawdata = Table[If[i ≠ j, Join[{f1data[[i, 1]]}, {f1data[[j, 1]]}, f2data[[i, j]][[1 ;; 7]], f3data[[i, j]], f3data[[j, i]]], 0],
     {i, 1, noofdatapts}, {j, 1, noofdatapts}];

   data = Table[
     If[i == j, 0,
      If[rawdata[[i, j]][[9]] === Null, "SN", Which[rawdata[[i, j]][[9]] == "conflict", "C", rawdata[[i, j]][[9]] == "potential conflict", "P",
         rawdata[[i, j]][[9]] == "coordination conflict", "CC", rawdata[[i, j]][[9]] == "potential coordination conflict", "CP", True, ""] <>
       Which[rawdata[[i, j]][[8]] == "same track", "S", rawdata[[i, j]][[8]] == "crossing track", "C", rawdata[[i, j]][[8]] == "opposite track", "O", True, ""] <>
       Which[0 <= rawdata[[i, j]][[3]] < 10.5, "1", 10.5 <= rawdata[[i, j]][[3]] < 20.5, "2", 20.5 <= rawdata[[i, j]][[3]] < 30.5, "3",
        30.5 <= rawdata[[i, j]][[3]] < 50.5, "4", 50.5 <= rawdata[[i, j]][[3]] < 80.5, "5", 80.5 <= rawdata[[i, j]][[3]], "6", True, ""] <>
       Which[rawdata[[i, j]][[7]] == "faster", "1", rawdata[[i, j]][[7]] == "same", "2", rawdata[[i, j]][[7]] == "slower", "3", True, ""] <>
       Which[0 <= rawdata[[i, j]][[4]] < 20.5 °, "1", 20.5 ° <= rawdata[[i, j]][[4]] < 45 °, "2", 45 ° <= rawdata[[i, j]][[4]] < 90.5 °, "3",
        90.5 ° <= rawdata[[i, j]][[4]] ≤ 135 °, "4", 135 ° < rawdata[[i, j]][[4]] < 159.5 °, "5", 159.5 ° <= rawdata[[i, j]][[4]] ≤ 180 °, "6"] <>
       Which[rawdata[[i, j]][[5]] == rawdata[[i, j]][[6]], "0", 0 <= rawdata[[i, j]][[5]] < 10.5, "1", 10.5 <= rawdata[[i, j]][[5]] < 20.5,
        "2", 20.5 <= rawdata[[i, j]][[5]] < 30.5, "3", 30.5 <= rawdata[[i, j]][[5]] < 50.5, "4", 50.5 <= rawdata[[i, j]][[5]] < 80.5, "5",
        80.5 <= rawdata[[i, j]][[5]], "6"] <>
       Which[rawdata[[i, j]][[5]] == rawdata[[i, j]][[6]], "0", 0 <= rawdata[[i, j]][[6]] < 10.5, "1", 10.5 <= rawdata[[i, j]][[6]] < 20.5,
        "2", 20.5 <= rawdata[[i, j]][[6]] < 30.5, "3", 30.5 <= rawdata[[i, j]][[6]] < 50.5, "4", 50.5 <= rawdata[[i, j]][[6]] < 80.5, "5",
        80.5 <= rawdata[[i, j]][[6]], "6"] <>
```

```
          If[rawdata[[i, j]][[11]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[10]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[15]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[14]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[12]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[13]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[16]] == True, "1", "2"] <>
          If[rawdata[[i, j]][[17]] == True, "1", "2"] <>
          Which[0 <= rawdata[[i, j]][[1]] < 15.5, "1", 15.5 <= rawdata[[i, j]][[1]] < 30.5, "2", 30.5 <= rawdata[[i, j]][[1]] < 45.5, "3",
           45.5 <= rawdata[[i, j]][[1]], "4"] <>
          Which[0 <= rawdata[[i, j]][[2]] < 15.5, "1", 15.5 <= rawdata[[i, j]][[2]] < 30.5, "2", 30.5 <= rawdata[[i, j]][[2]] < 45.5, "3",
           45.5 <= rawdata[[i, j]][[2]], "4"]
        ]
       ]
      , {i, 1, noofdatapts}, {j, 1, noofdatapts}];
    data
   ];
(*Initialization 120 vacancies to fill matrices of f1,f2,f3 and f4 and checks and saves the codes*)
codes = Table[Null, {i, 1, 120}];

Monitor[
 For[airspacenumber = 1, airspacenumber ≤ 120, airspacenumber++,
  f1data = Table[f1[data[[airspacenumber]][[i, 2 ;; 8]], boundaries[[airspacenumber]]], {i, 1, Length[data[[airspacenumber]]]}];
  f2data = Table[If[i == j, 0, f2[data[[airspacenumber]][[i, 2 ;; 8]], data[[airspacenumber]][[j, 2 ;; 8]], boundaries[[airspacenumber]]]],
    {i, 1, Length[data[[airspacenumber]]]}, {j, 1, Length[data[[airspacenumber]]]}];
  f3data = f3[data[[airspacenumber]], f2data, boundaries[[airspacenumber]]];
  f4data = f4[f1data, f2data, f3data];
  f5data = Table[If[i == j, f1data[[i]][[2 ;; 4]], f4data[[i, j]] <> ", " <> f2data[[i, j]][[8]]], {i, 1, Length[data[[airspacenumber]]]},
    {j, 1, Length[data[[airspacenumber]]]}];
  codes[[airspacenumber]] =
   f6data = Transpose[Prepend[Transpose[Prepend[f5data, data[[airspacenumber]][[All, 1]]]], Join[{0}, data[[airspacenumber]][[All, 1]]]]];
  Print[airspacenumber];]
 , airspacenumber]
```

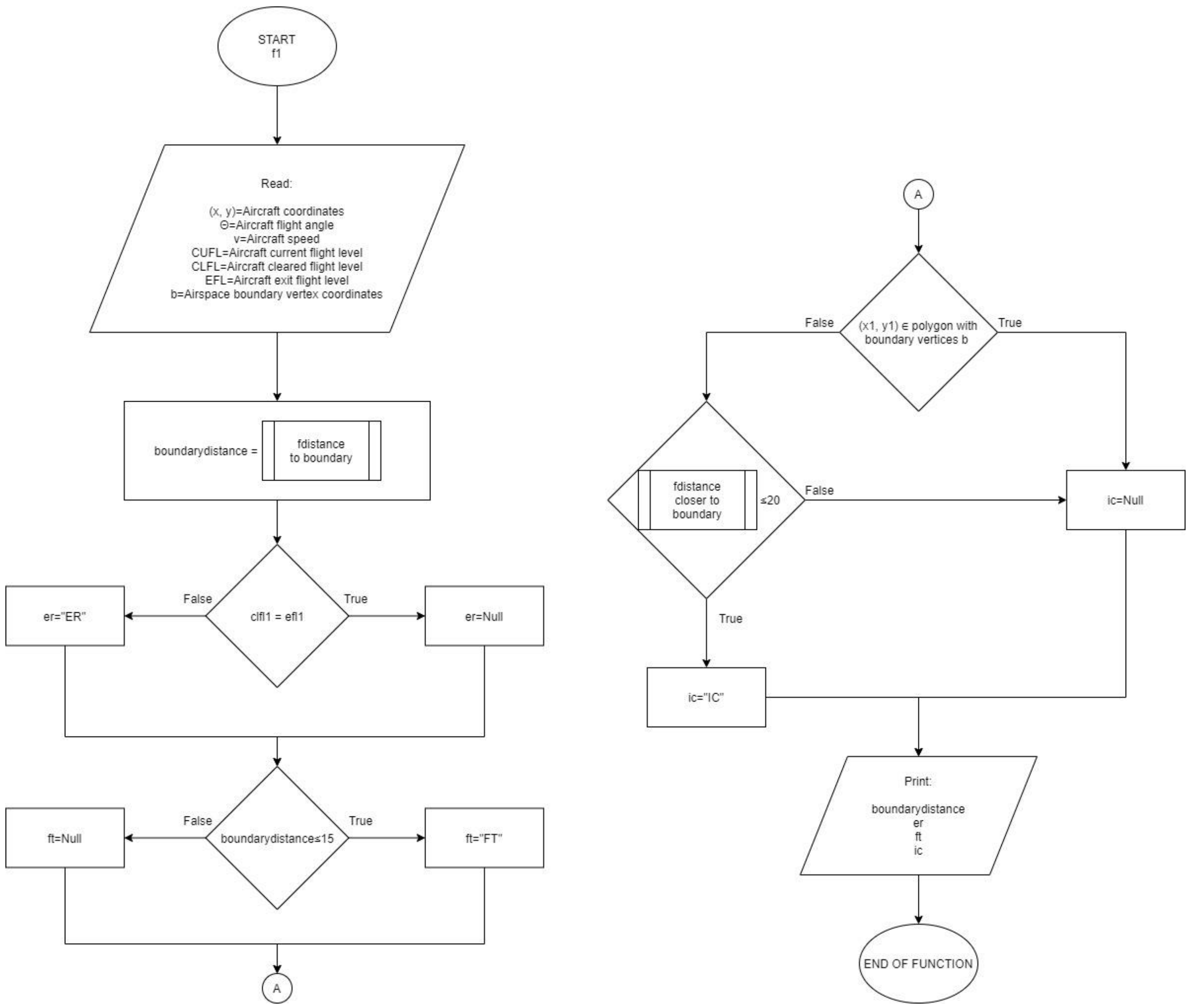
Appendix 4 – Flowchart of ATCO tasks automatization

The main automation program

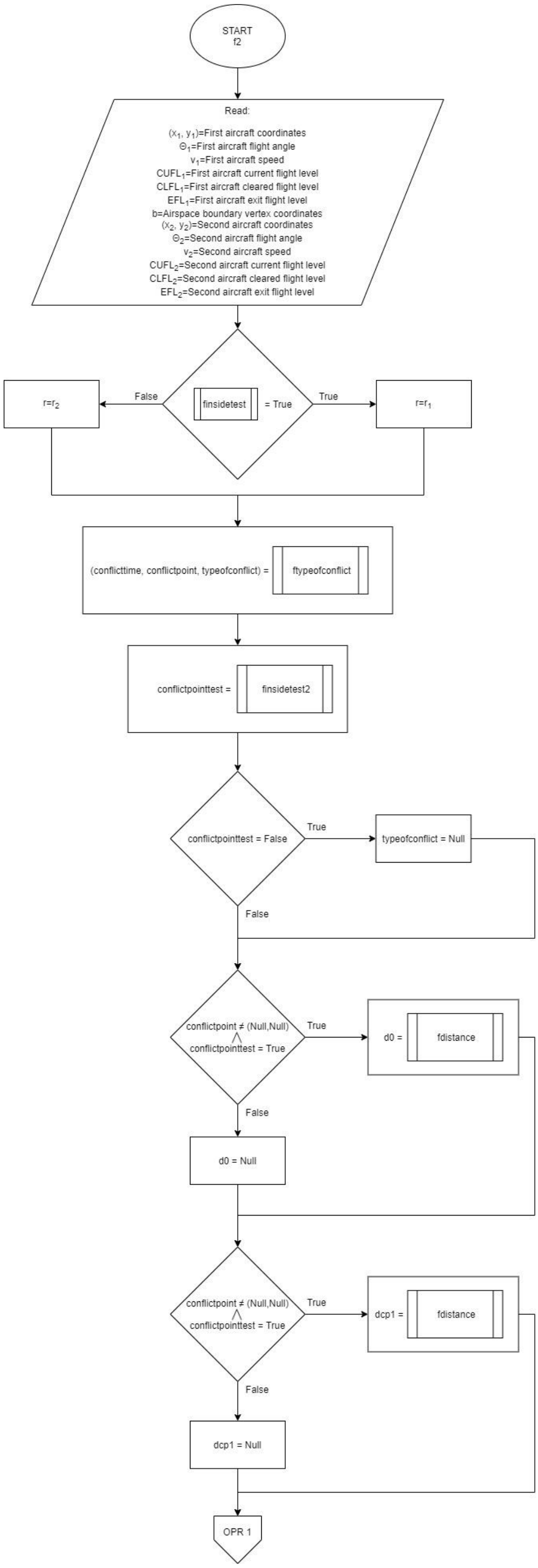


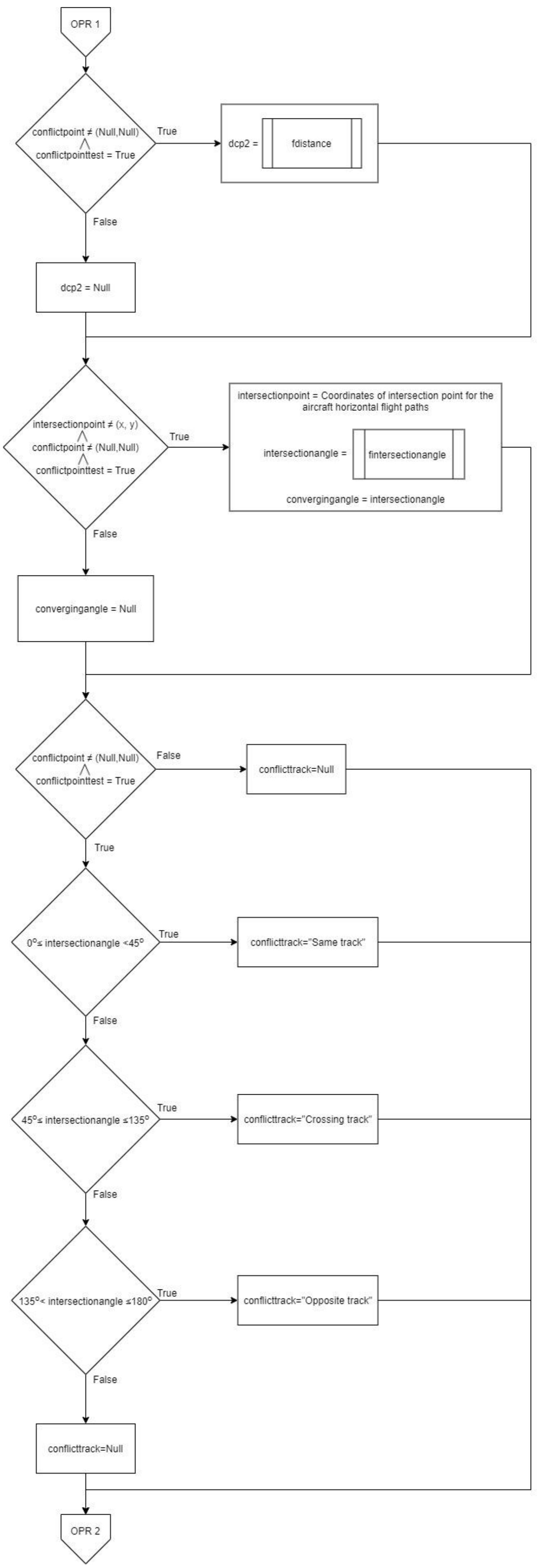

OPR
i=1
i++
B
i ≤
length of data[airspacenumber]
True
False
j=1
j++
j ≤
length of data[airspacenumber]
True
False
B
i = j
True
False
f2data[i,j]=0
f2data[i,j] =
f2 (data[airspacenumber][i, 2 ;; 8],
data[airspacenumber][j, 2 ;; 8],
boundaries[airspacenumber])
f3data =
f3 (data[airspacenumber],
f2data,
boundaries[airspacenumber])
f4data =
f4 (f1data, f2data, f3data)
i=1
i++
C
i ≤
length of data[airspacenumber]
True
False
j=1
j++
j ≤
length of data[airspacenumber]
True
False
C
i = j
True
False
f5data[i,j] = f1data[i][2 ;; 4]
f5data[i,j] = f4data[i, j] concatenate ", " concatenate f2data[i, j][8]
codes[airspacenumber] = transpose (0 join data[airspacenumber]
[All, 1] join transpose (data[airspacenumber][All, 1] join f5data))
A

The f1 function

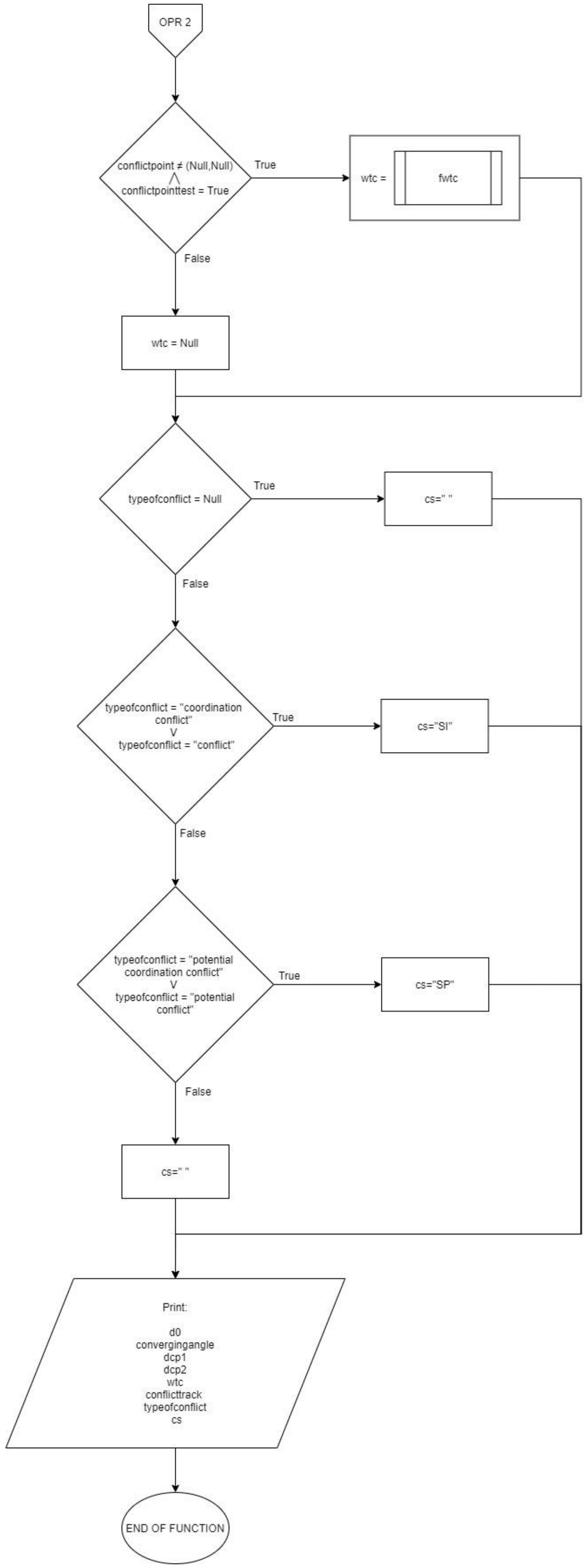


The f2 function

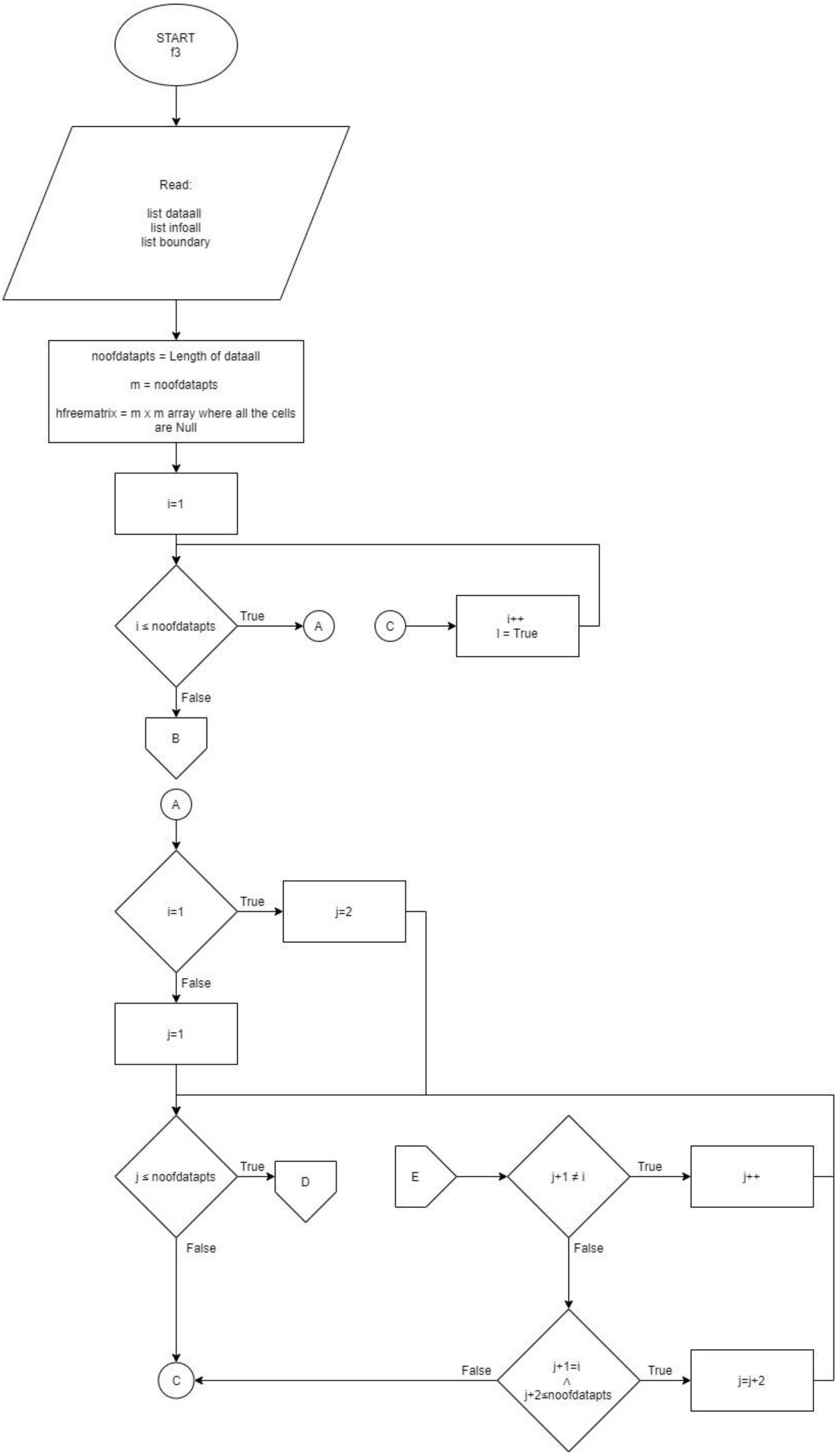


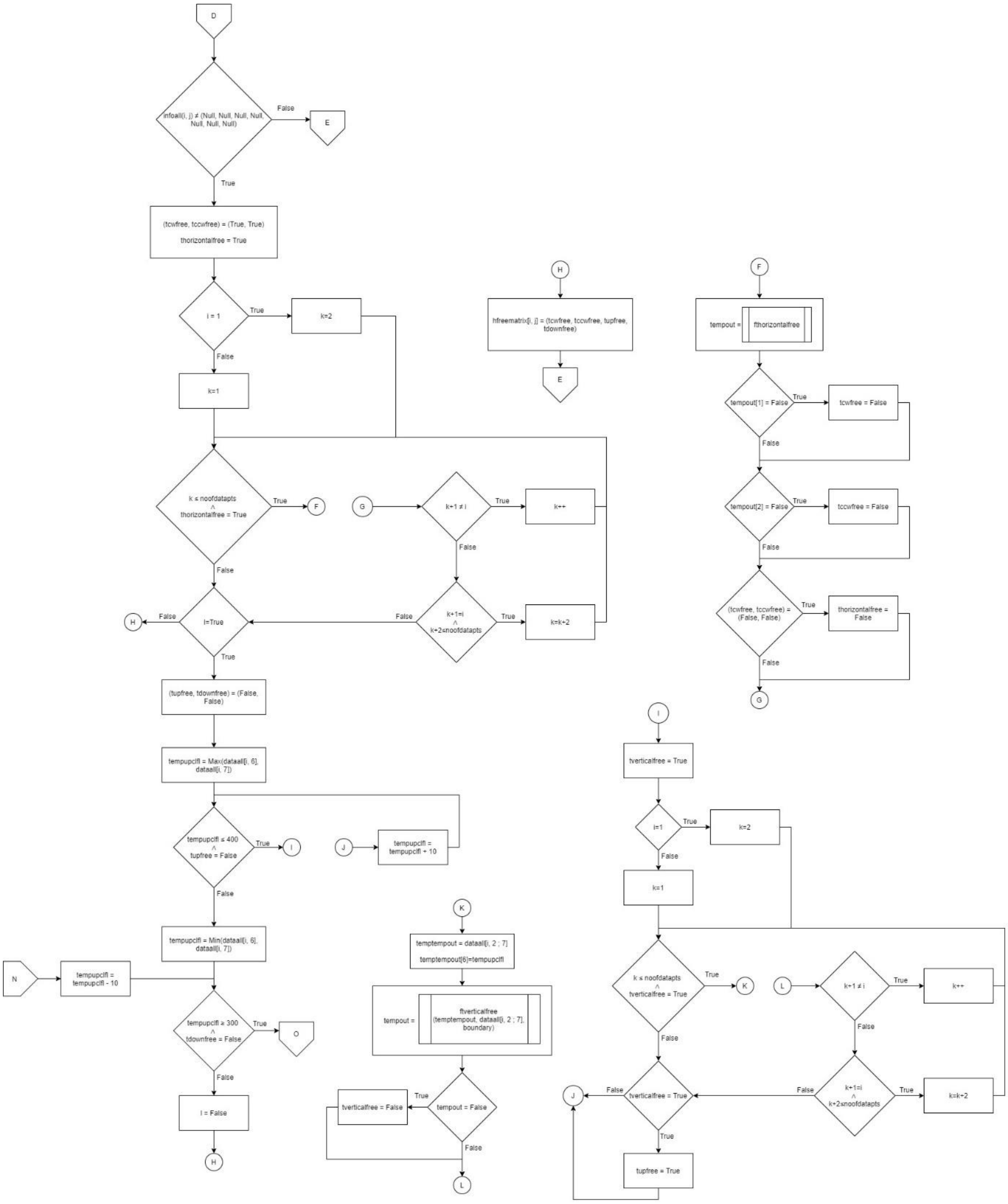


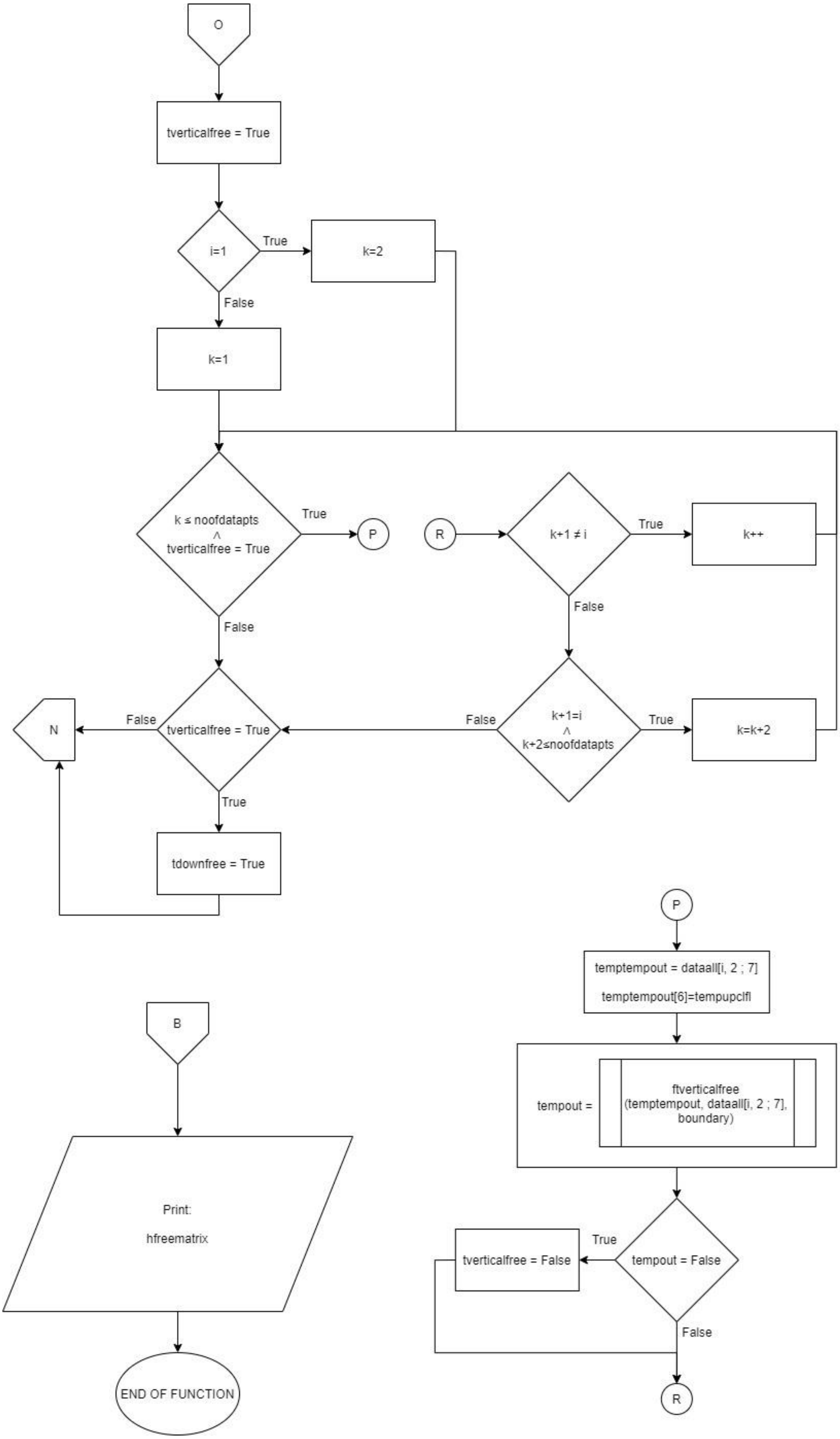
OPR 1
conflictpoint ≠ (Null,Null) ⋀ conflictpointtest = True
True
dcp2 =
fdistance
False
dcp2 = Null
intersectionpoint ≠ (x, y) ⋀ conflictpoint ≠ (Null,Null) ⋀ conflictpointtest = True
True
intersectionpoint = Coordinates of intersection point for the aircraft horizontal flight paths
intersectionangle =
fintersectionangle
convergingangle = intersectionangle
False
convergingangle = Null
conflictpoint ≠ (Null,Null) ⋀ conflictpointtest = True
False
conflicttrack=Null
True
0°≤ intersectionangle <45°
True
conflicttrack="Same track"
False
45°≤ intersectionangle ≤135°
True
conflicttrack="Crossing track"
False
135°< intersectionangle ≤180°
True
conflicttrack="Opposite track"
False
conflicttrack=Null
OPR 2


OPR 2
conflictpoint ≠ (Null,Null) ⋀ conflictpointtest = True
True
wtc =
fwtc
False
wtc = Null
typeofconflict = Null
True
cs=" "
False
typeofconflict = "coordination conflict" V typeofconflict = "conflict"
True
cs="SI"
False
typeofconflict = "potential coordination conflict" V typeofconflict = "potential conflict"
True
cs="SP"
False
cs=" "
Print:
d0
convergingangle
dcp1
dcp2
wtc
conflicttrack
typeofconflict
cs
END OF FUNCTION

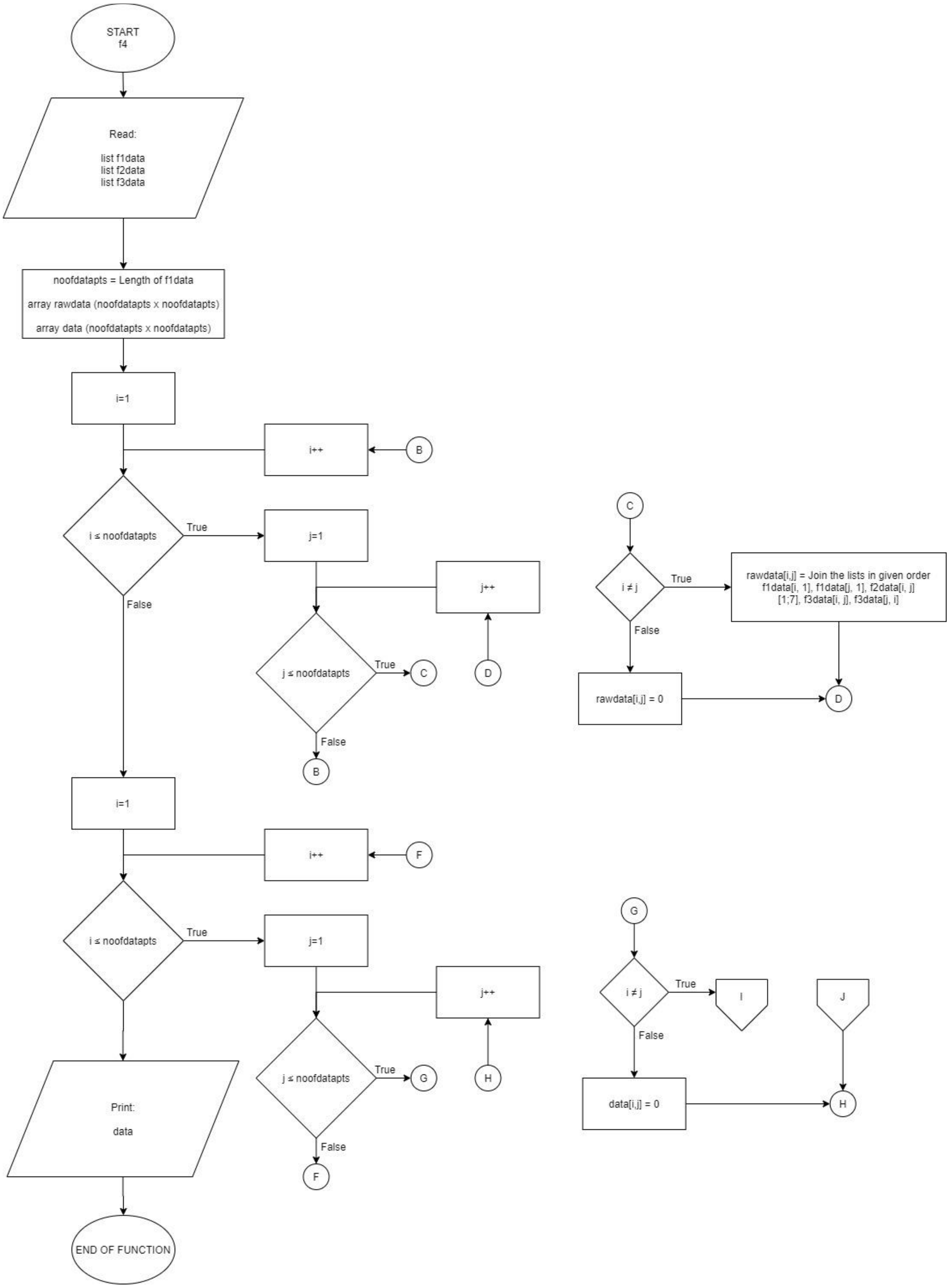
The f3 function

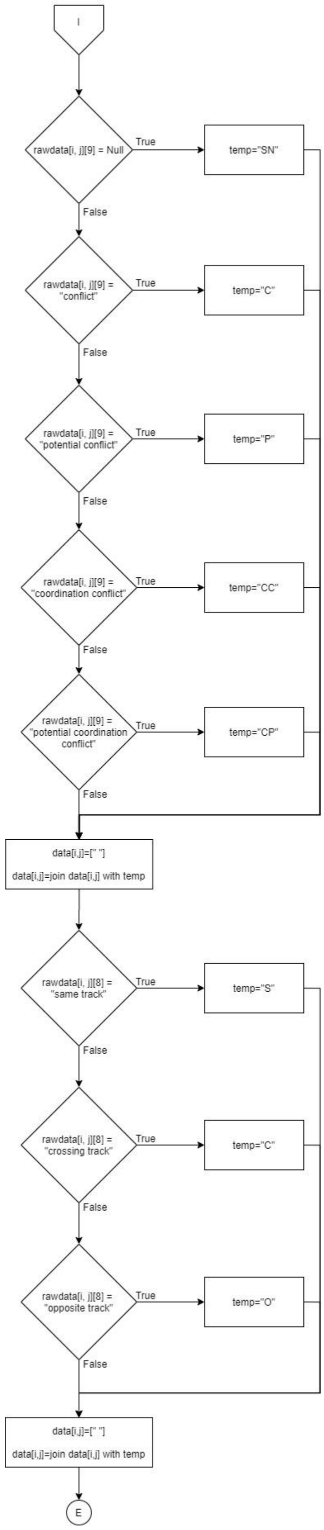


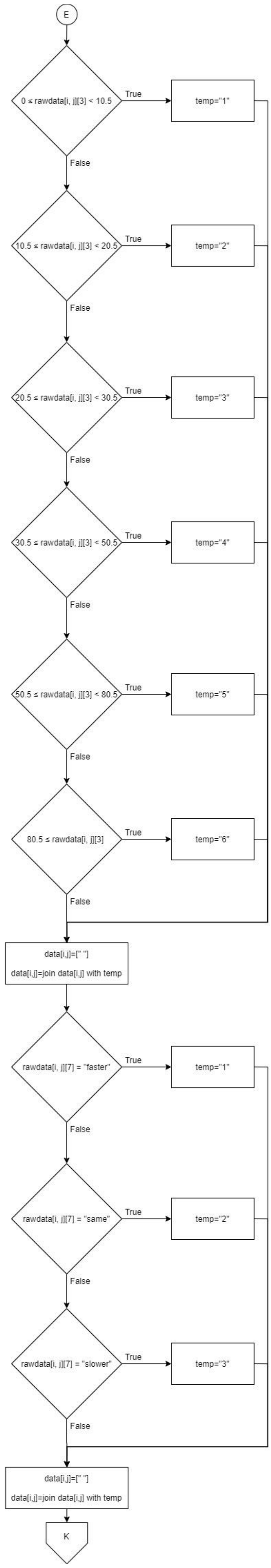

D
infoall(i, j) ≠ (Null, Null, Null, Null, Null, Null, Null)
False
E
True
(tcwfree, tccwfree) = (True, True)
thorizontalfree = True
i = 1
True
k=2
False
k=1
k ≤ noofdatapts ∧ thorizontalfree = True
True
F
False
G
k+1 ≠ i
True
k++
False
k+1=i ∧ k+2≤noofdatapts
True
k=k+2
False
I=True
False
H
True
(tupfree, tdownfree) = (False, False)
tempupclfl = Max(dataall[i, 6], dataall[i, 7])
tempupclfl ≤ 400 ∧ tupfree = False
True
I
J
tempupclfl = tempupclfl + 10
False
tempupclfl = Min(dataall[i, 6], dataall[i, 7])
N
tempupclfl = tempupclfl - 10
tempupclfl ≥ 300 ∧ tdownfree = False
True
O
False
I = False
H
H
hfreematrix[i, j] = (tcwfree, tccwfree, tupfree, tdownfree)
E
F
tempout = fthorizontalfree
tempout[1] = False
True
tcwfree = False
False
tempout[2] = False
True
tccwfree = False
False
(tcwfree, tccwfree) = (False, False)
True
thorizontalfree = False
False
G
K
temptempout = dataall[i, 2 ; 7]
temptempout[6]=tempupclfl
tempout = ftverticalfree (temptempout, dataall[i, 2 ; 7], boundary)
tempout = False
True
tverticalfree = False
False
L
I
tverticalfree = True
i=1
True
k=2
False
k=1
k ≤ noofdatapts ∧ tverticalfree = True
True
K
L
k+1 ≠ i
True
k++
False
k+1=i ∧ k+2≤noofdatapts
True
k=k+2
False
tverticalfree = True
False
J
True
tupfree = True

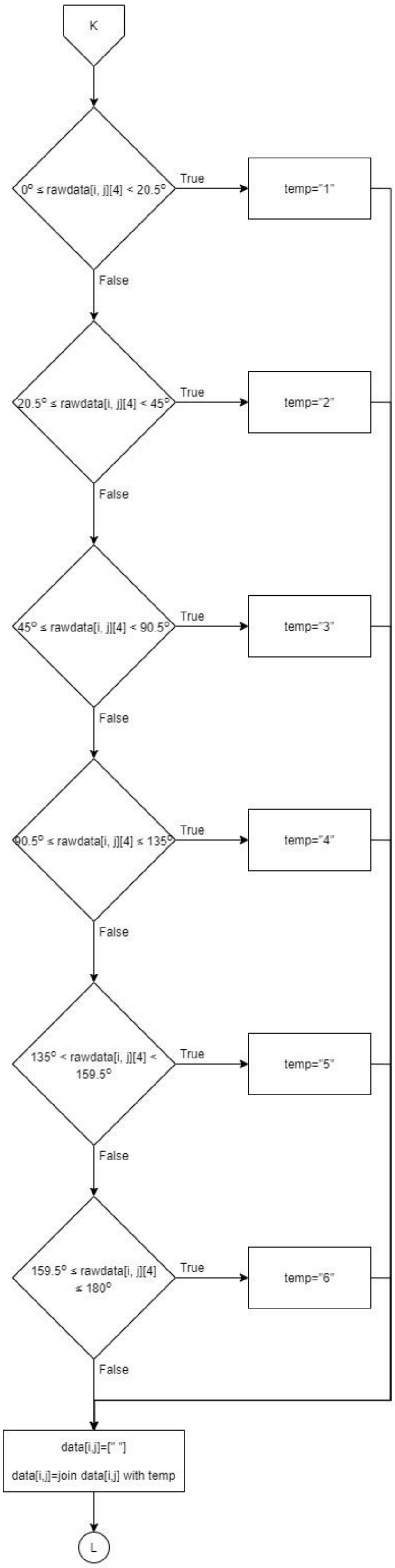

O
tverticalfree = True
i=1
True
k=2
False
k=1
k ≤ noofdatapts
∧
tverticalfree = True
True
P
R
k+1 ≠ i
True
k++
False
False
N
False
tverticalfree = True
False
k+1=i
∧
k+2≤noofdatapts
True
k=k+2
True
tdownfree = True
P
temptempout = dataall[i, 2 ; 7]
temptempout[6]=tempupclfl
tempout =
ftverticalfree
(temptempout, dataall[i, 2 ; 7],
boundary)
True
tverticalfree = False
tempout = False
False
R
B
Print:
hfreematrix
END OF FUNCTION

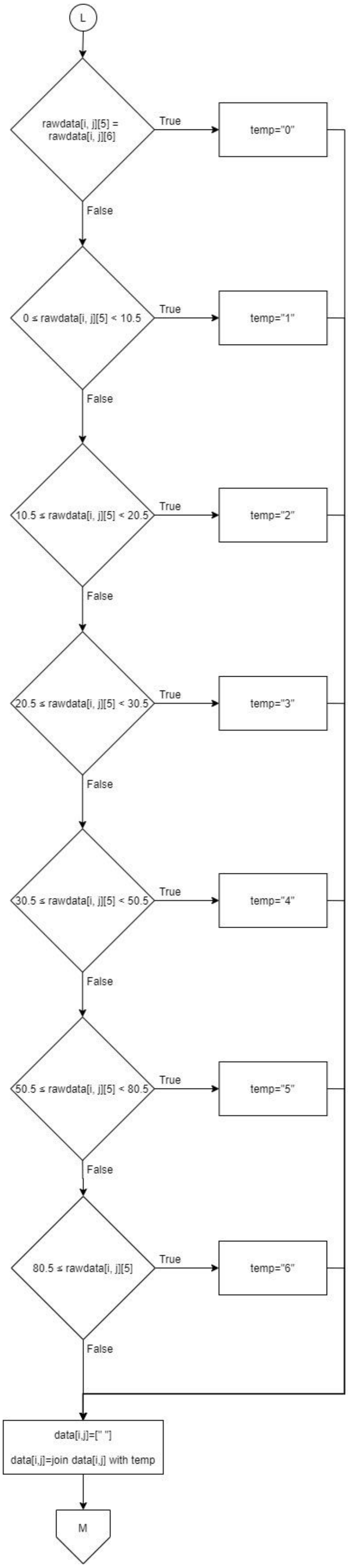
The f4 function

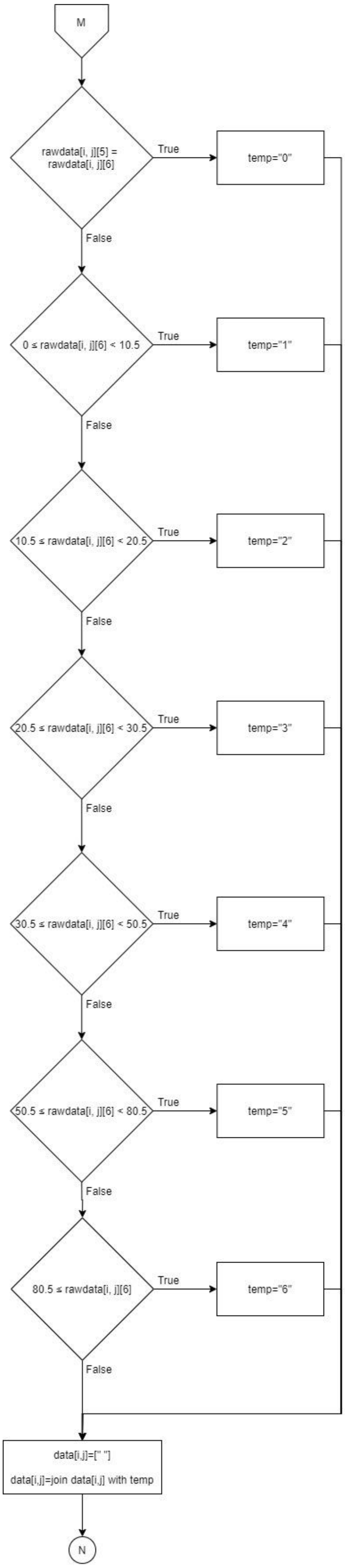


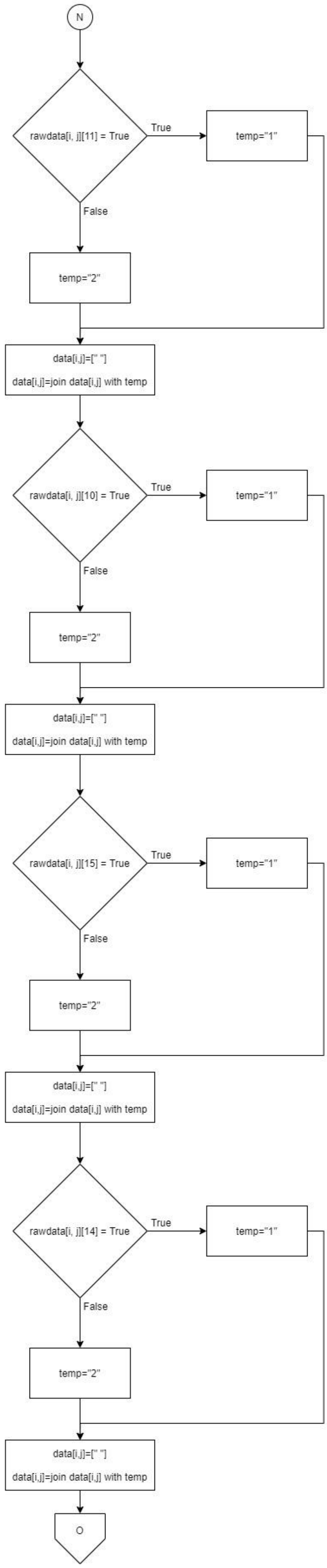

I
rawdata[i, j][9] = Null
True
temp="SN"
False
rawdata[i, j][9] = "conflict"
temp="C"
rawdata[i, j][9] = "potential conflict"
temp="P"
rawdata[i, j][9] = "coordination conflict"
temp="CC"
rawdata[i, j][9] = "potential coordination conflict"
temp="CP"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][8] = "same track"
temp="S"
rawdata[i, j][8] = "crossing track"
temp="C"
rawdata[i, j][8] = "opposite track"
temp="O"
E

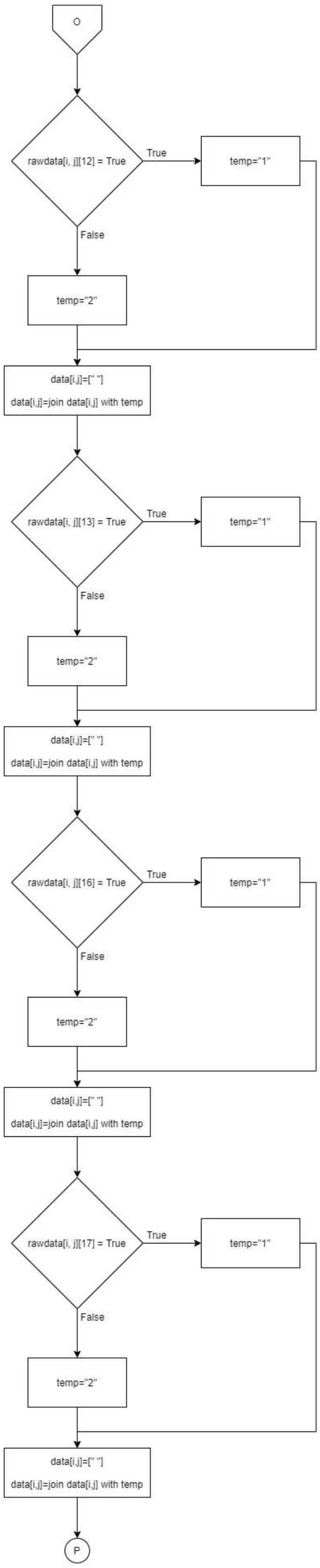

E
0 ≤ rawdata[i, j][3] < 10.5
True
temp="1"
False
10.5 ≤ rawdata[i, j][3] < 20.5
temp="2"
20.5 ≤ rawdata[i, j][3] < 30.5
temp="3"
30.5 ≤ rawdata[i, j][3] < 50.5
temp="4"
50.5 ≤ rawdata[i, j][3] < 80.5
temp="5"
80.5 ≤ rawdata[i, j][3]
temp="6"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][7] = "faster"
temp="1"
rawdata[i, j][7] = "same"
temp="2"
rawdata[i, j][7] = "slower"
temp="3"
K

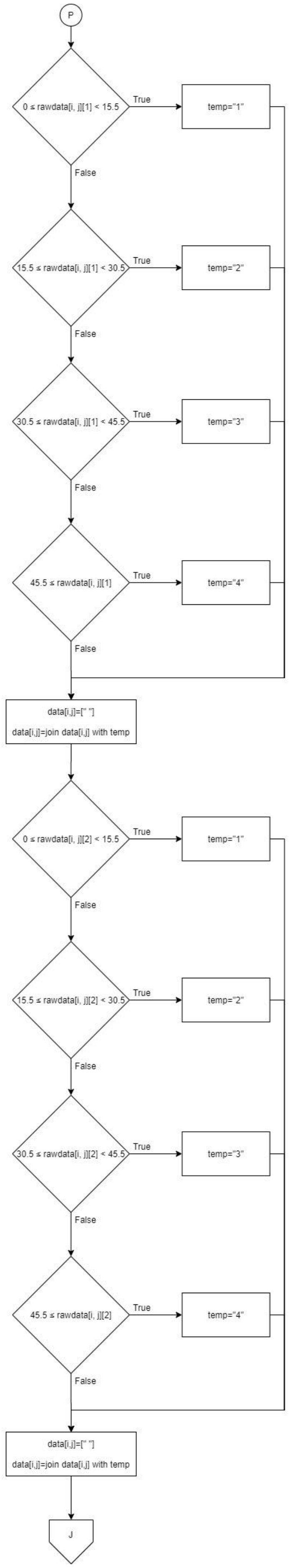

K
0° ≤ rawdata[i, j][4] < 20.5°
True
temp="1"
False
20.5° ≤ rawdata[i, j][4] < 45°
True
temp="2"
False
45° ≤ rawdata[i, j][4] < 90.5°
True
temp="3"
False
90.5° ≤ rawdata[i, j][4] ≤ 135°
True
temp="4"
False
135° < rawdata[i, j][4] < 159.5°
True
temp="5"
False
159.5° ≤ rawdata[i, j][4] ≤ 180°
True
temp="6"
False
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
L


L
rawdata[i, j][5] = rawdata[i, j][6]
True
temp="0"
False
0 ≤ rawdata[i, j][5] < 10.5
True
temp="1"
False
10.5 ≤ rawdata[i, j][5] < 20.5
True
temp="2"
False
20.5 ≤ rawdata[i, j][5] < 30.5
True
temp="3"
False
30.5 ≤ rawdata[i, j][5] < 50.5
True
temp="4"
False
50.5 ≤ rawdata[i, j][5] < 80.5
True
temp="5"
False
80.5 ≤ rawdata[i, j][5]
True
temp="6"
False
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
M

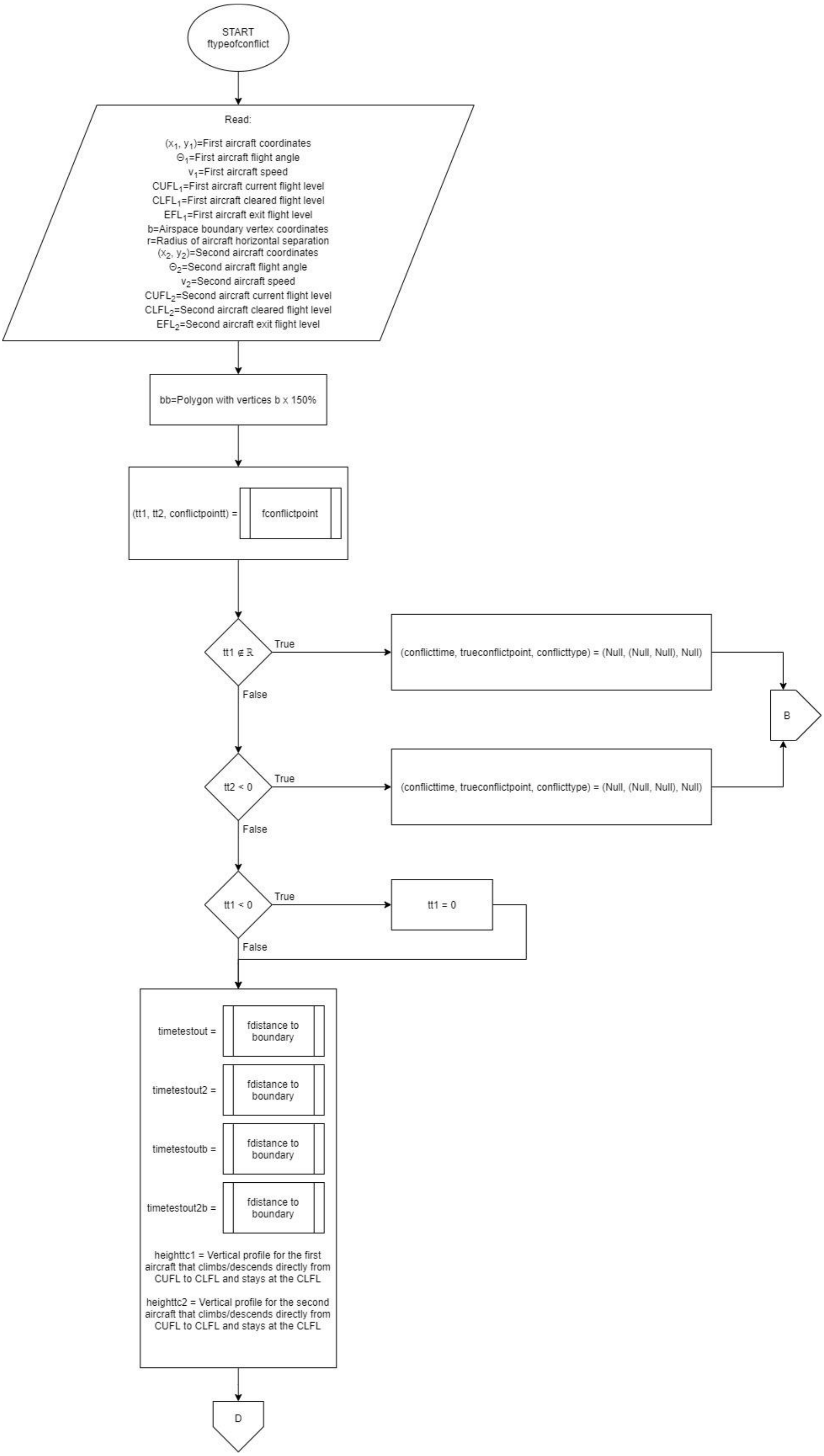

M
rawdata[i, j][5] = rawdata[i, j][6]
True
temp="0"
False
0 ≤ rawdata[i, j][6] < 10.5
True
temp="1"
False
10.5 ≤ rawdata[i, j][6] < 20.5
True
temp="2"
False
20.5 ≤ rawdata[i, j][6] < 30.5
True
temp="3"
False
30.5 ≤ rawdata[i, j][6] < 50.5
True
temp="4"
False
50.5 ≤ rawdata[i, j][6] < 80.5
True
temp="5"
False
80.5 ≤ rawdata[i, j][6]
True
temp="6"
False
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
N

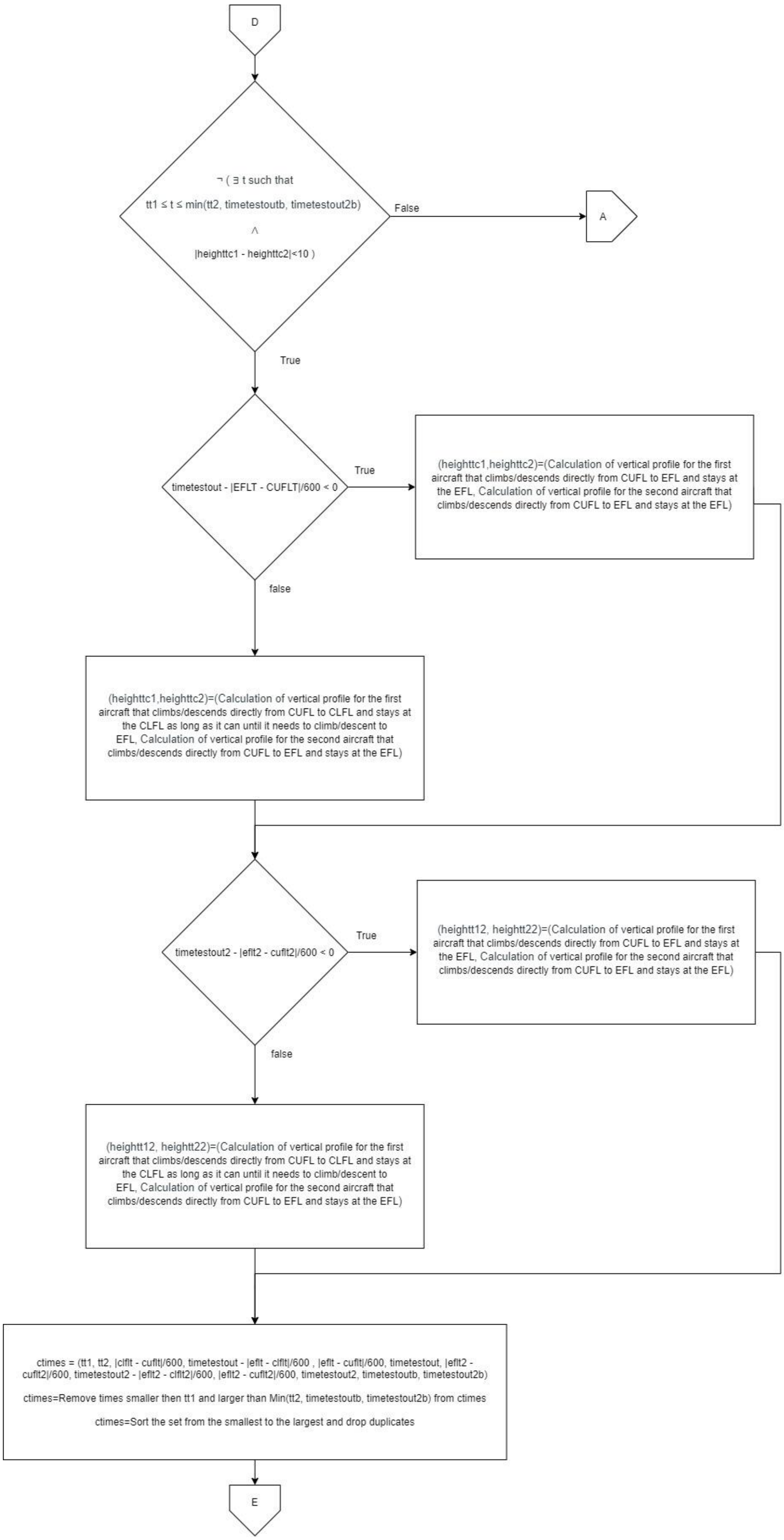

N
rawdata[i, j][11] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][10] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][15] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][14] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
O


O
rawdata[i, j][12] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][13] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][16] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
rawdata[i, j][17] = True
True
temp="1"
False
temp="2"
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
P

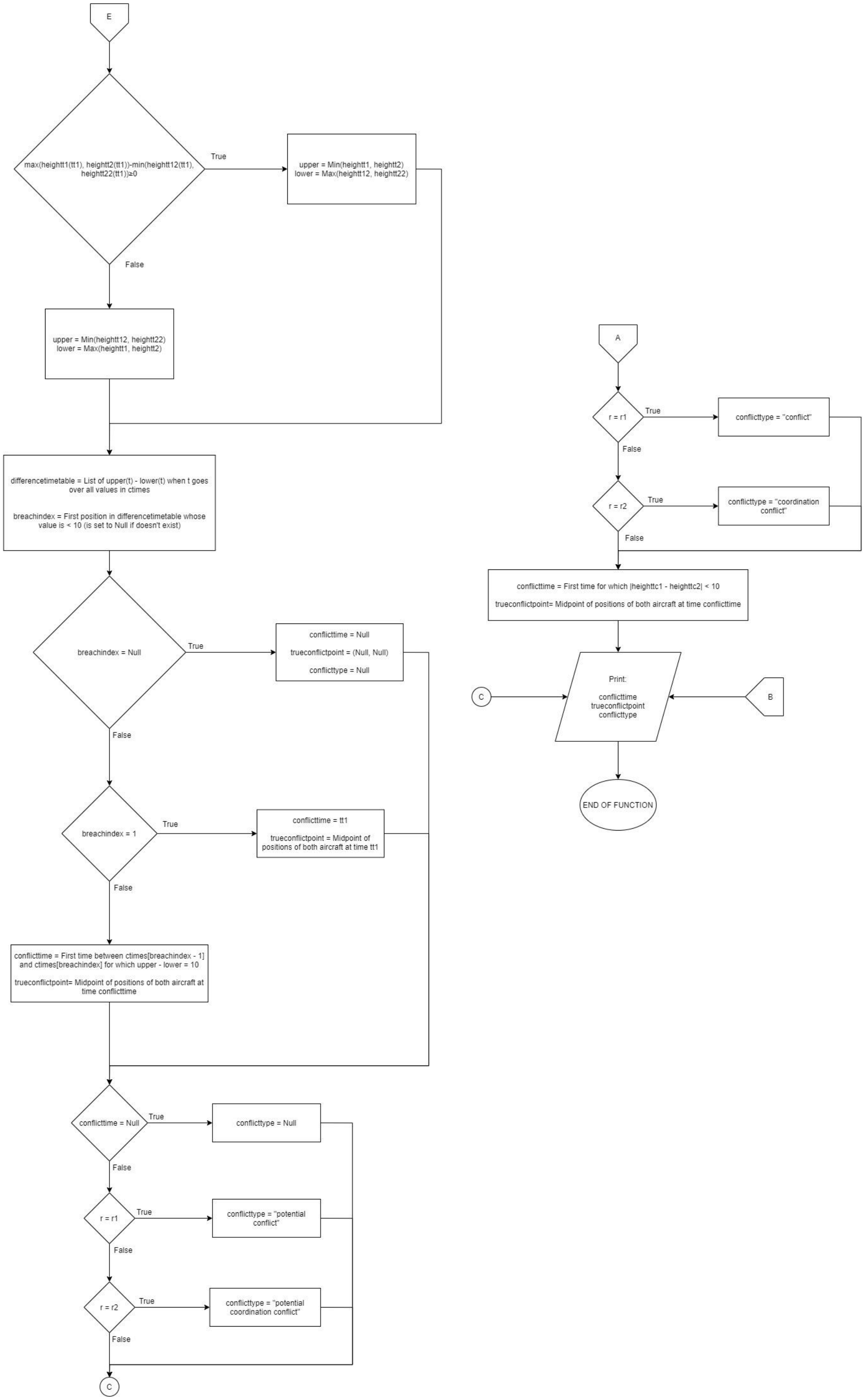

P
0 ≤ rawdata[i, j][1] < 15.5
True
temp="1"
False
15.5 ≤ rawdata[i, j][1] < 30.5
True
temp="2"
False
30.5 ≤ rawdata[i, j][1] < 45.5
True
temp="3"
False
45.5 ≤ rawdata[i, j][1]
True
temp="4"
False
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
0 ≤ rawdata[i, j][2] < 15.5
True
temp="1"
False
15.5 ≤ rawdata[i, j][2] < 30.5
True
temp="2"
False
30.5 ≤ rawdata[i, j][2] < 45.5
True
temp="3"
False
45.5 ≤ rawdata[i, j][2]
True
temp="4"
False
data[i,j]=[" "]
data[i,j]=join data[i,j] with temp
J

The typeofconflict function

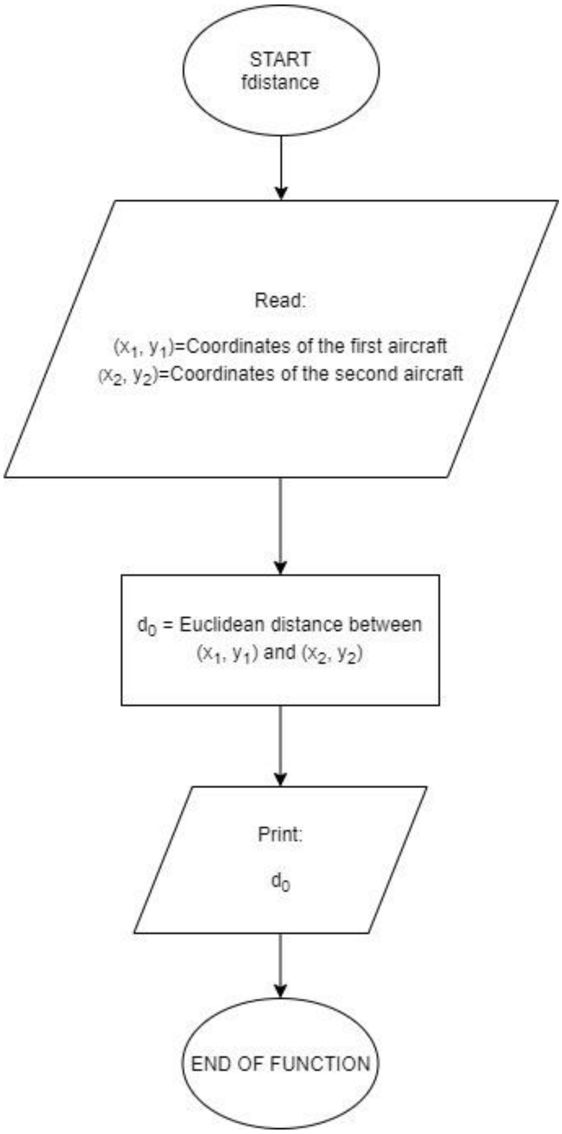




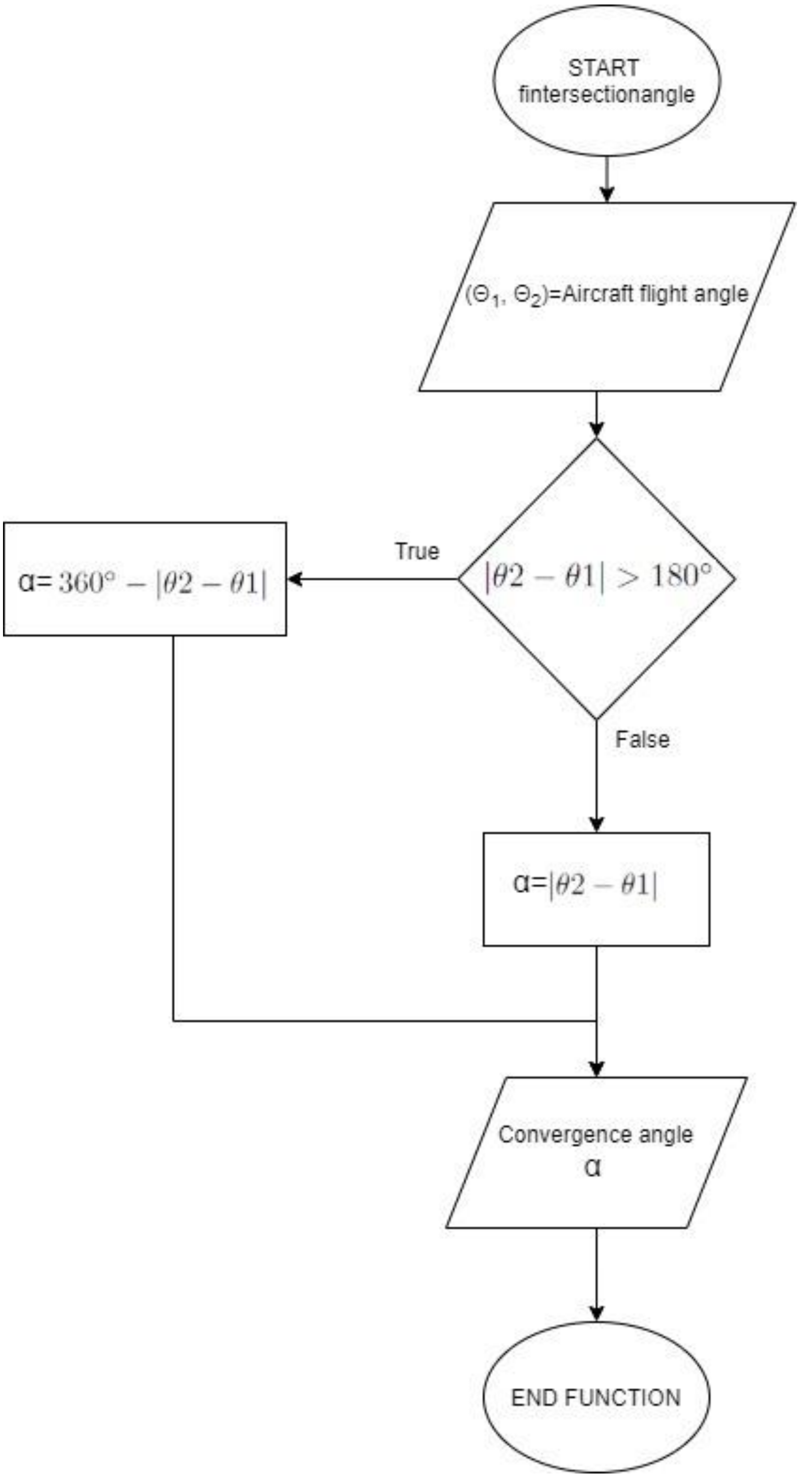
D
¬ (∃ t such that
tt1 ≤ t ≤ min(tt2, timetestoutb, timetestout2b)
∧
|heighttc1 - heighttc2|<10)
False
A
True
timetestout - |EFLT - CUFLT|/600 < 0
True
(heighttc1,heighttc2)=(Calculation of vertical profile for the first aircraft that climbs/descends directly from CUFL to EFL and stays at the EFL, Calculation of vertical profile for the second aircraft that climbs/descends directly from CUFL to EFL and stays at the EFL)
false
(heighttc1,heighttc2)=(Calculation of vertical profile for the first aircraft that climbs/descends directly from CUFL to CLFL and stays at the CLFL as long as it can until it needs to climb/descent to EFL, Calculation of vertical profile for the second aircraft that climbs/descends directly from CUFL to EFL and stays at the EFL)
timetestout2 - |eflt2 - cuflt2|/600 < 0
True
(heightt12, heightt22)=(Calculation of vertical profile for the first aircraft that climbs/descends directly from CUFL to EFL and stays at the EFL, Calculation of vertical profile for the second aircraft that climbs/descends directly from CUFL to EFL and stays at the EFL)
false
(heightt12, heightt22)=(Calculation of vertical profile for the first aircraft that climbs/descends directly from CUFL to CLFL and stays at the CLFL as long as it can until it needs to climb/descent to EFL, Calculation of vertical profile for the second aircraft that climbs/descends directly from CUFL to EFL and stays at the EFL)
ctimes = (tt1, tt2, |clflt - cuflt|/600, timetestout - |eflt - clflt|/600 , |eflt - cuflt|/600, timetestout, |eflt2 - cuflt2|/600, timetestout2 - |eflt2 - clflt2|/600, |eflt2 - cuflt2|/600, timetestout2, timetestoutb, timetestout2b)
ctimes=Remove times smaller then tt1 and larger than Min(tt2, timetestoutb, timetestout2b) from ctimes
ctimes=Sort the set from the smallest to the largest and drop duplicates
E


E
max(heightt1(tt1), heightt2(tt1))-min(heightt12(tt1), heightt22(tt1))≥0
True
upper = Min(heightt1, heightt2)
lower = Max(heightt12, heightt22)
False
upper = Min(heightt12, heightt22)
lower = Max(heightt1, heightt2)
differencetimetable = List of upper(t) - lower(t) when t goes over all values in ctimes
breachindex = First position in differencetimetable whose value is < 10 (is set to Null if doesn't exist)
breachindex = Null
True
conflicttime = Null
trueconflictpoint = (Null, Null)
conflicttype = Null
False
breachindex = 1
True
conflicttime = tt1
trueconflictpoint = Midpoint of positions of both aircraft at time tt1
False
conflicttime = First time between ctimes[breachindex - 1] and ctimes[breachindex] for which upper - lower = 10
trueconflictpoint= Midpoint of positions of both aircraft at time conflicttime
conflicttime = Null
True
conflicttype = Null
False
r = r1
True
conflicttype = "potential conflict"
False
r = r2
True
conflicttype = "potential coordination conflict"
False
C
A
r = r1
True
conflicttype = "conflict"
False
r = r2
True
conflicttype = "coordination conflict"
False
conflicttime = First time for which |heighttc1 - heighttc2| < 10
trueconflictpoint= Midpoint of positions of both aircraft at time conflicttime
Print:
conflicttime
trueconflictpoint
conflicttype
C
B
END OF FUNCTION

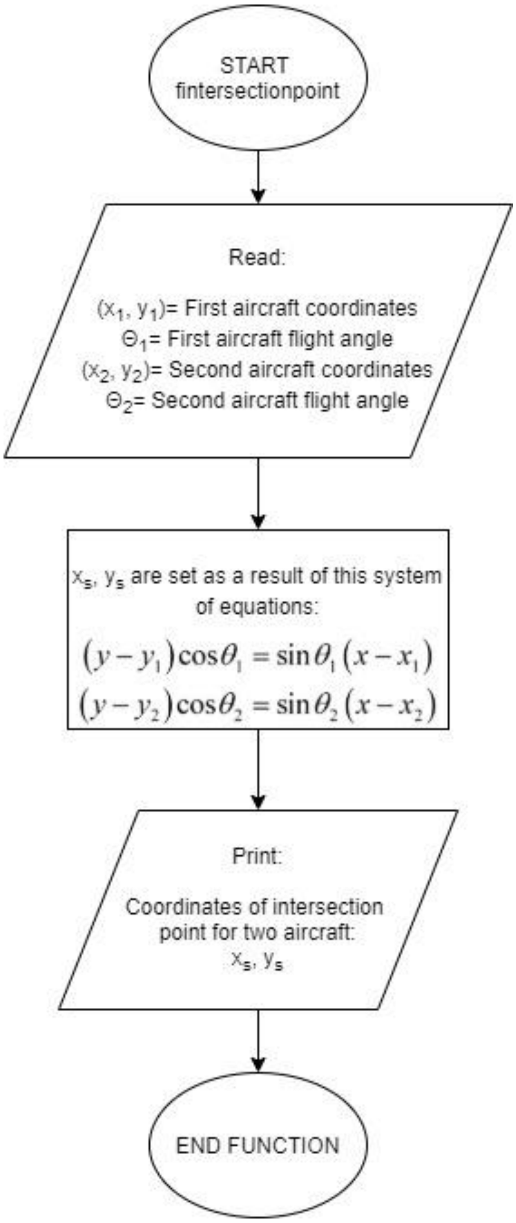
The fdistance function



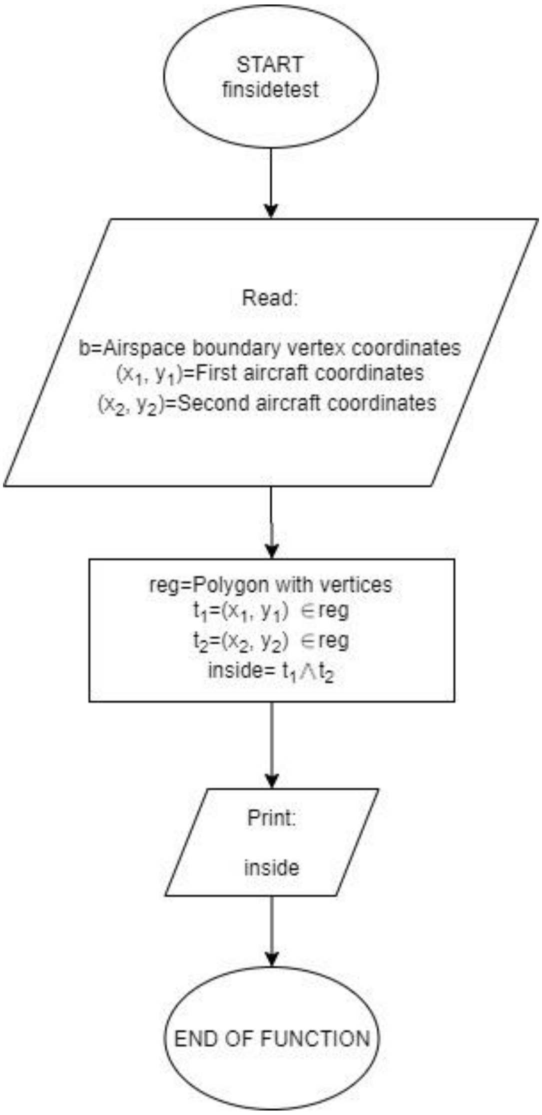
The fintersectionangle function



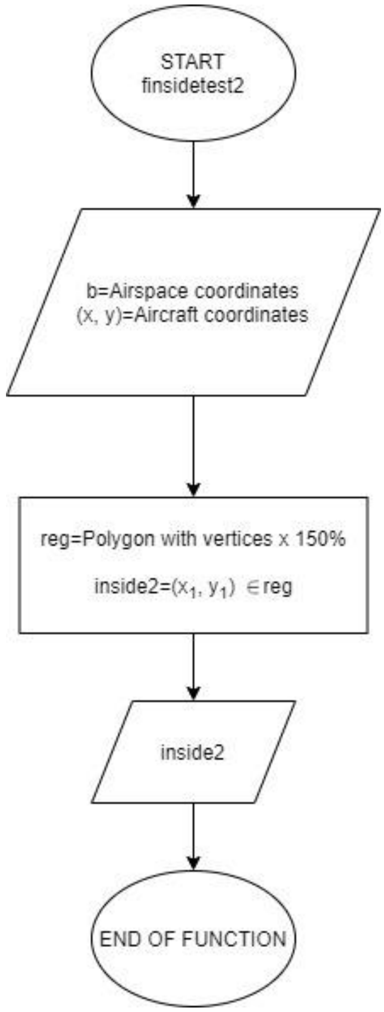
The fintersectionpoint function



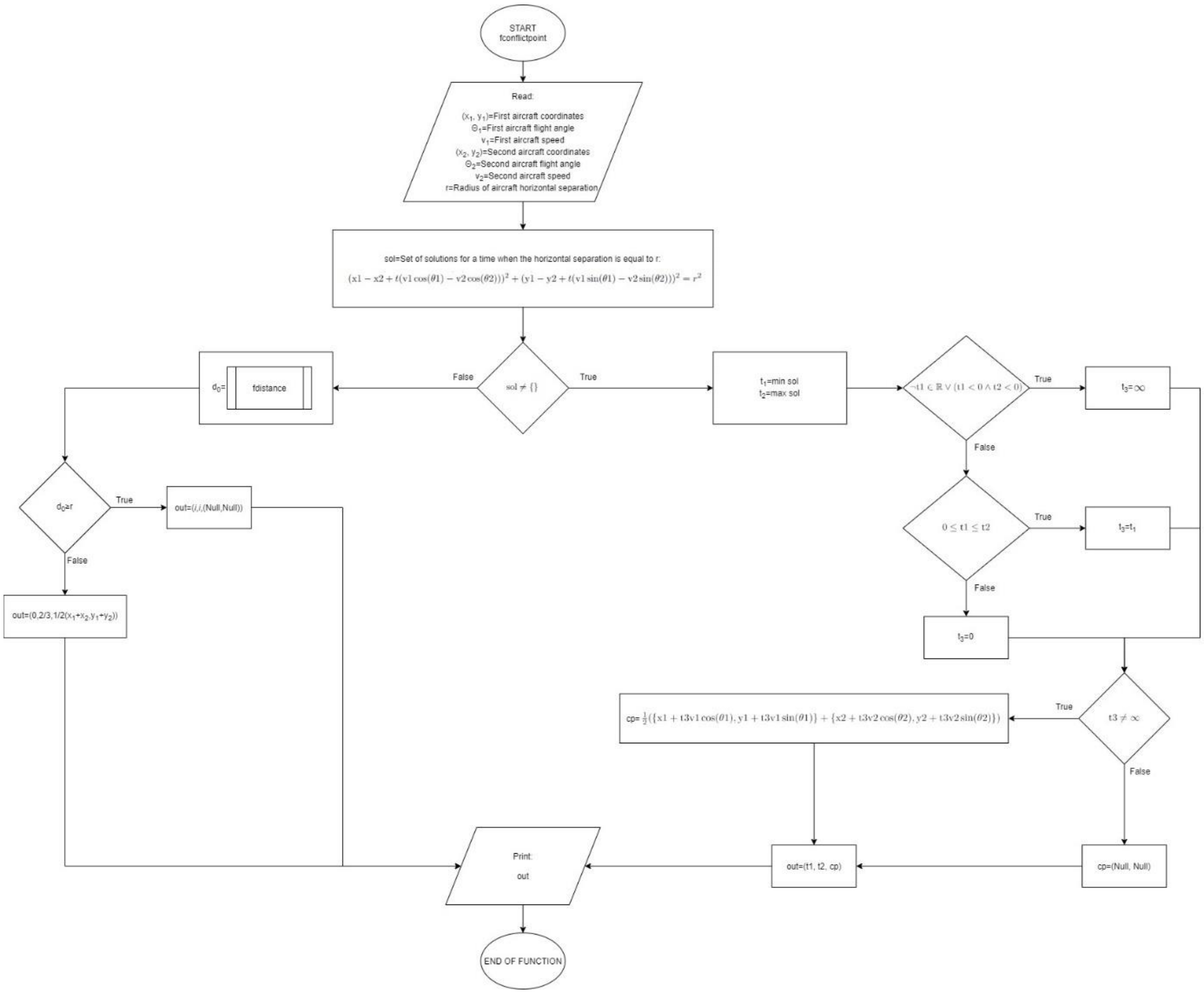
The finsidetest function



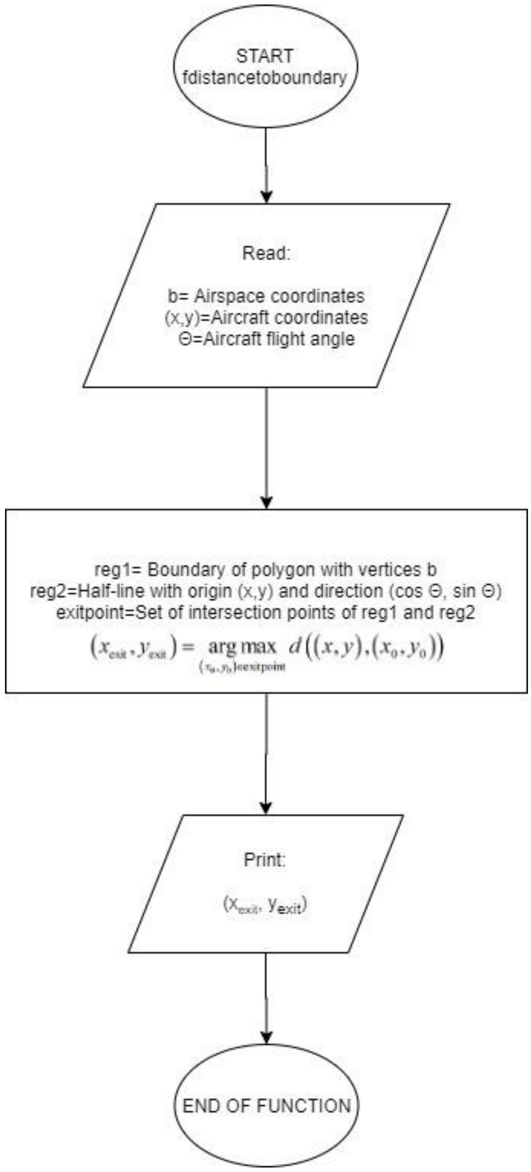
The finsidetest2 function



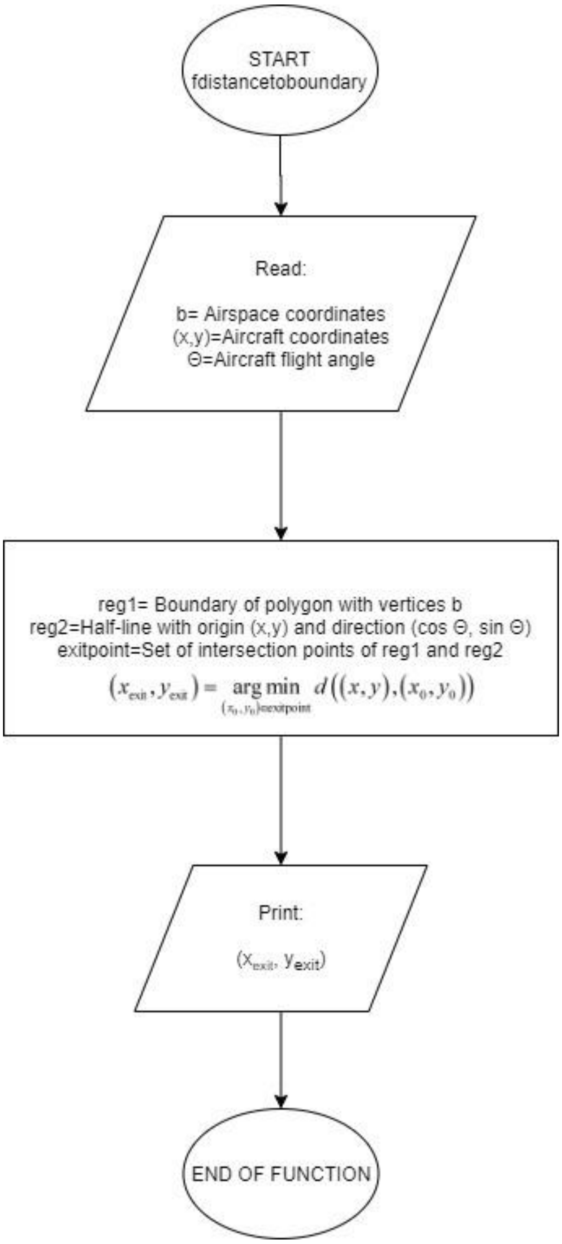
The fconflictpoint function



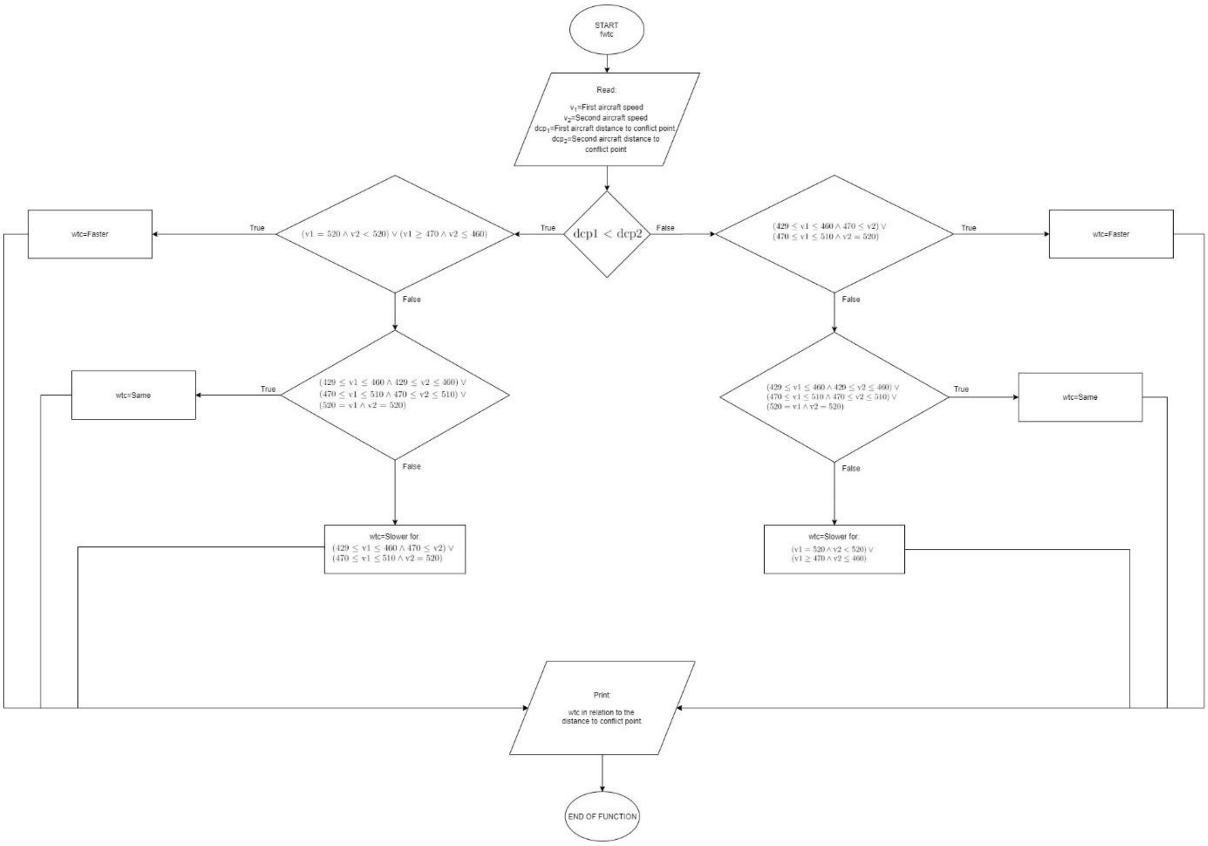
The fdistancetoboundary function



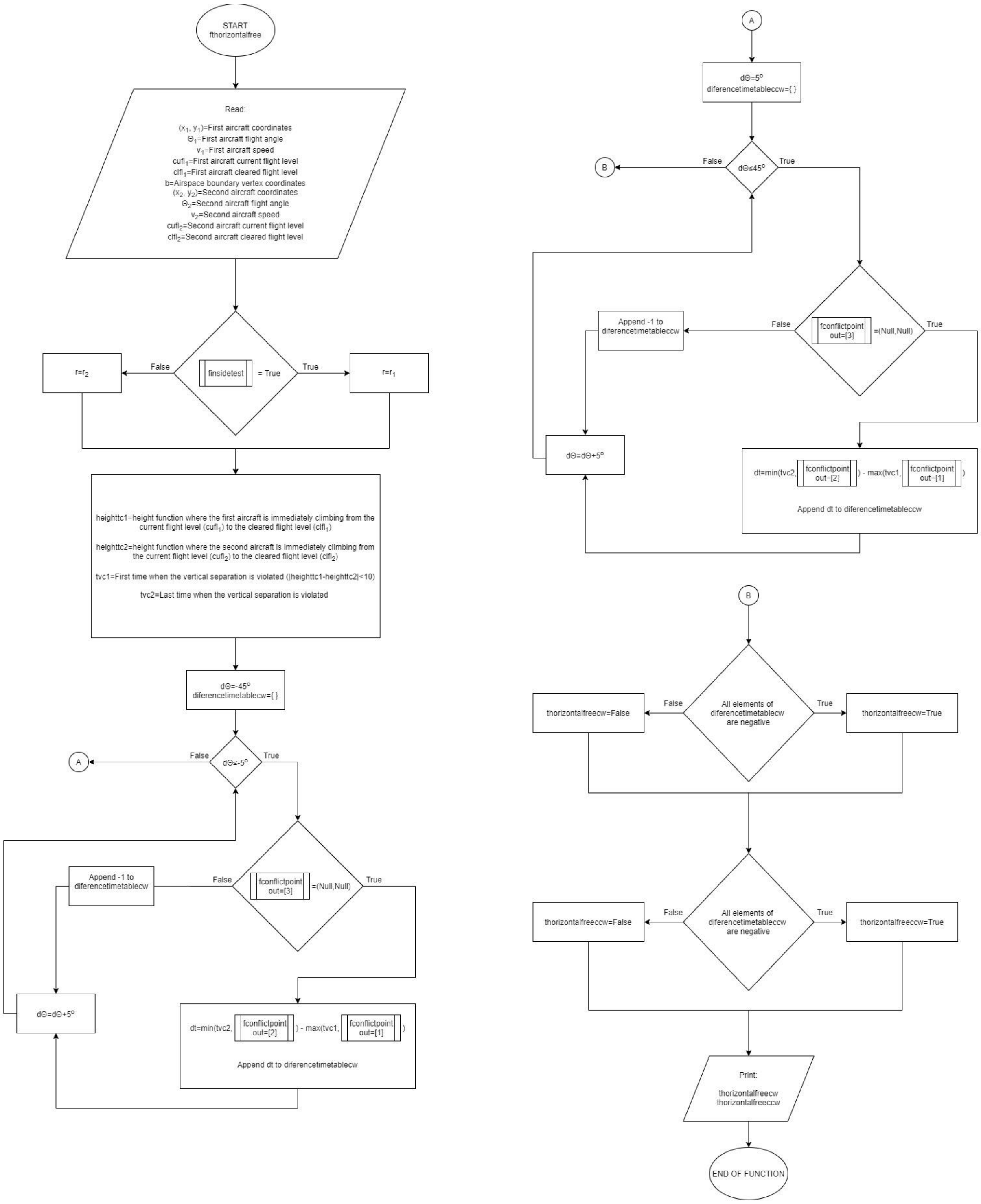
The fdistanceclosertoboundary function



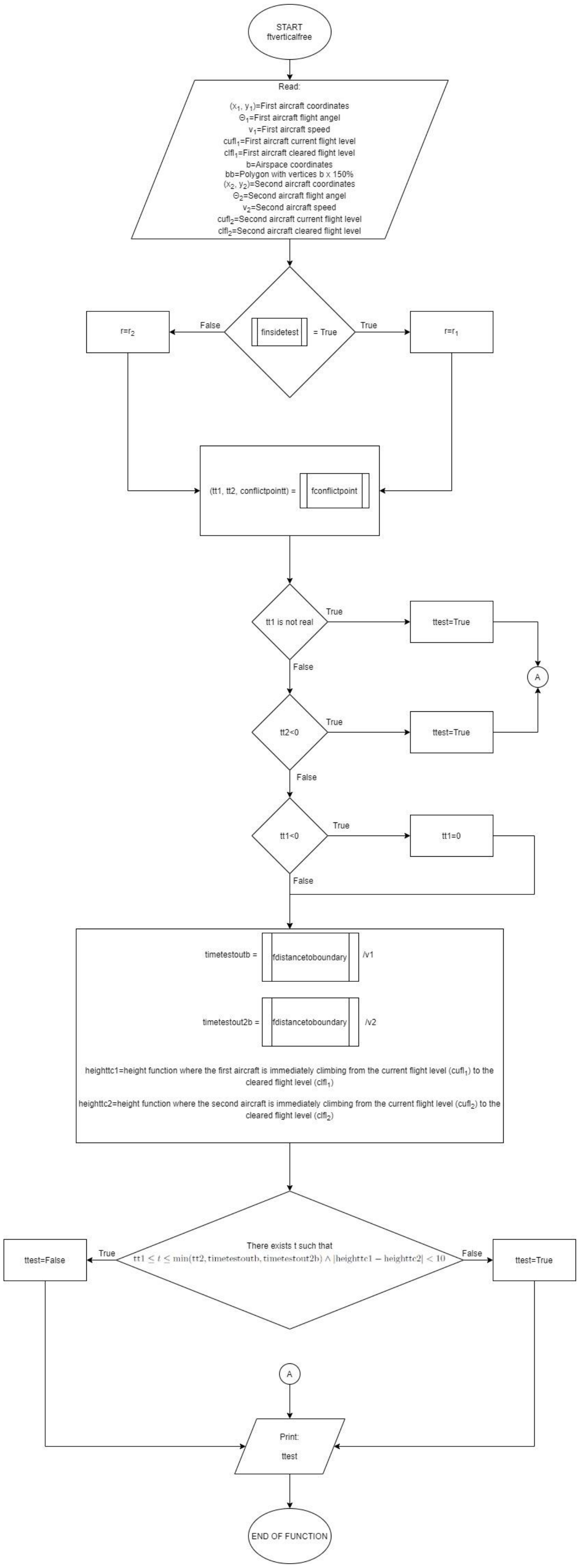
The fwtc function



The fthorizontalfree function



The ftverticalfree function



Appendix 5 – Python code of the Merge Sort algorithm

```
def customLessThanRobust(leftTmp, rightTmp):
    flag = 1
    msg_str = "What is more complex: " + str(leftTmp) + " or " + str(rightTmp)+ " =>"
    while (flag):
        answer = input(msg_str)
        if (answer==str(leftTmp)):
            #print("da")
            flag = 0
            return False
        elif (answer==str(rightTmp)):
            #print("ne")
            flag = 0
            return True
        else:
            print("Input error: Type either " + str(leftTmp) + " or " + str(rightTmp))
            flag = 1


def mergeSortCustom(alist):

   #print("Splitting ",alist)

   if len(alist)>1:
       mid = len(alist)//2
       lefthalf = alist[:mid]
       righthalf = alist[mid:]

       #recursion
       mergeSortCustom(lefthalf)
       mergeSortCustom(righthalf)

       i=0
       j=0
       k=0

       while i < len(lefthalf) and j < len(righthalf):
           if customLessThanRobust(lefthalf[i], righthalf[j]):
               alist[k]=lefthalf[i]
               i=i+1
           else:
               alist[k]=righthalf[j]
               j=j+1
           k=k+1

       while i < len(lefthalf):
           alist[k]=lefthalf[i]
           i=i+1
           k=k+1

       while j < len(righthalf):
           alist[k]=righthalf[j]
           j=j+1
           k=k+1

   #print("Merging ",alist)

#alist = [54,26,93,17,77,31,44,55,20]
alist = ['A1','A2','A3','A4','A5','A6','A7','A8','A9','A10',
         'B1','B2','B3','B4','B5','B6','B7','B8','B9','B10',
         'C1','C2','C3','C4','C5','C6','C7','C8','C9','C10']
print(alist)
mergeSortCustom(alist)
print(alist)
```

Appendix 6 – Data gathering information for all 18 ATCO

Data gathering from ATCO no. 1

Date and time of the experiments:	Candidate:	Years of experience:
23.05.2019. / 10:00 h - 12:09 h	Anonymous no. 1	11

Time required to rank the traffic:	Traffic sample taken and group:
02h09m (10-min break)	A1-C10 / G1

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A5 =>A4
4. What is more complex: A6 or A7 =>A7
5. What is more complex: A5 or A6 =>A5
6. What is more complex: A5 or A7 =>A7
7. What is more complex: A4 or A7 =>A4
8. What is more complex: A1 or A6 =>A6
9. What is more complex: A2 or A6 =>A6
10. What is more complex: A3 or A6 =>A6
11. What is more complex: A8 or A9 =>A9
12. What is more complex: A10 or B1 =>A10

26. What is more complex: A9 or B4 =>A9
27. What is more complex: A9 or B2 =>A9
28. What is more complex: A1 or B5 =>A1
29. What is more complex: A1 or B3 =>A1
30. What is more complex: A1 or B1 =>B1
31. What is more complex: A2 or B1 =>A2
32. What is more complex: A2 or A8 =>A2
33. What is more complex: A2 or A10 =>A2
34. What is more complex: A2 or B4 =>B4
35. What is more complex: A3 or B4 =>B4
36. What is more complex: A6 or B4 =>A6
37. What is more complex: A6 or B2 =>A6

51. What is more complex: B6 or C1 =>C1
52. What is more complex: B8 or C1 =>C1
53. What is more complex: C3 or C4 =>C4
54. What is more complex: C5 or C6 =>C6
55. What is more complex: C3 or C5 =>C3
56. What is more complex: C3 or C6 =>C3
57. What is more complex: C7 or C8 =>C7
58. What is more complex: C9 or C10 =>C10
59. What is more complex: C8 or C9 =>C9
60. What is more complex: C7 or C9 =>C9
61. What is more complex: C5 or C8 =>C5
62. What is more complex: C5 or C7 =>C5

76. What is more complex: C2 or C5 =>C2
77. What is more complex: C2 or C6 =>C2
78. What is more complex: C2 or C3 =>C3
79. What is more complex: B10 or C3 =>B10
80. What is more complex: B10 or C10 =>C10
81. What is more complex: B5 or B7 =>B7
82. What is more complex: B3 or B7 =>B7
83. What is more complex: A1 or B7 =>B7
84. What is more complex: B1 or B7 =>B7
85. What is more complex: A8 or B7 =>B7
86. What is more complex: A10 or B7 =>B7
87. What is more complex: A2 or B7 =>A2

13. What is more complex: A8 or B1 =>A8

14. What is more complex: A8 or A10 =>A10

15. What is more complex: A9 or A10 =>A9

16. What is more complex: B2 or B3 =>B2

17. What is more complex: B4 or B5 =>B4

18. What is more complex: B3 or B5 =>B3

19. What is more complex: B3 or B4 =>B4

20. What is more complex: B2 or B4 =>B2

21. What is more complex: B1 or B5 =>B1

22. What is more complex: B1 or B3 =>B1

23. What is more complex: B1 or B4 =>B4

24. What is more complex: A8 or B4 =>B4

25. What is more complex: A10 or B4 =>B4

38. What is more complex: A6 or A9 =>A9

39. What is more complex: A5 or A9 =>A9

40. What is more complex: A7 or A9 =>A9

41. What is more complex: A4 or A9 =>A4

42. What is more complex: B7 or B8 =>B8

43. What is more complex: B6 or B7 =>B6

44. What is more complex: B6 or B8 =>B8

45. What is more complex: B9 or B10 =>B10

46. What is more complex: C1 or C2 =>C2

47. What is more complex: B9 or C1 =>B9

48. What is more complex: B9 or C2 =>C2

49. What is more complex: B10 or C2 =>B10

50. What is more complex: B7 or C1 =>C1

63. What is more complex: C5 or C9 =>C5

64. What is more complex: C5 or C10 =>C10

65. What is more complex: C6 or C10 =>C10

66. What is more complex: C3 or C10 =>C10

67. What is more complex: C4 or C10 =>C4

68. What is more complex: B7 or C8 =>C8

69. What is more complex: B6 or C8 =>C8

70. What is more complex: B8 or C8 =>C8

71. What is more complex: C1 or C8 =>C8

72. What is more complex: B9 or C8 =>C8

73. What is more complex: C2 or C8 =>C2

74. What is more complex: C2 or C7 =>C2

75. What is more complex: C2 or C9 =>C2

88. What is more complex: A2 or B6 =>B6

89. What is more complex: A3 or B6 =>B6

90. What is more complex: B4 or B6 =>B6

91. What is more complex: B2 or B6 =>B6

92. What is more complex: A6 or B6 =>B6

93. What is more complex: A5 or B6 =>B6

94. What is more complex: A7 or B6 =>B6

95. What is more complex: A9 or B6 =>A9

96. What is more complex: A9 or B8 =>A9

97. What is more complex: A9 or C1 =>C1

98. What is more complex: A4 or C1 =>A4

99. What is more complex: A4 or B9 =>B9

Ranking results:

['B5', 'B3', 'A1', 'B1', **1** 'A8', 'A10', 'B7', 'A2', **2** 'A3', 'B4', 'B2', 'A6', 'A5', 'A7', **3** 'B6', 'B8', 'A9', 'C1', 'A4', 'B9', 'C8', 'C7', 'C9', 'C5', 'C6', **4** 'C2', 'C3', 'B10', 'C10', 'C4' **5**]

Linearly interpolated scores:

1. B5=0.25 (0+1/4)

2. B3=0.5 (0+2/4)

3. A1=0.75 (0+3/4)

16. B8=3.181818 (3+2/11)

17. A9=3.272727 (3+3/11)

18. C1=3.363636 (3+4/11)

4.	B1=1	(0+4/4)	19.	A4=3.454546	(3+5/11)
5.	A8=1.25	(1+1/4)	20.	B9=3.545455	(3+6/11)
6.	A10=1.5	(1+2/4)	21.	C8=3.636364	(3+7/11)
7.	B7=1.75	(1+3/4)	22.	C7=3.727273	(3+8/11)
8.	A2=2	(1+4/4)	23.	C9=3.818182	(3+9/11)
9.	A3=2.166667	(2+1/6)	24.	C5=3.909091	(3+10/11)
10.	B4=2.333333	(2+2/6)	25.	C6=4	(3+11/11)
11.	B2=2.5	(2+3/6)	26.	C2=4.2	(4+1/5)
12.	A6=2.666667	(2+4/6)	27.	C3=4.4	(4+2/5)
13.	A5=2.833333	(2+5/6)	28.	B10=4.6	(4+3/5)
14.	A7=3	(2+6/6)	29.	C10=4.8	(4+4/5)
15.	B6=3.0909091	(3+1/11)	30.	C4=5	(4+5/5)

Comment form the candidate:	**Observations from the moderator:**
Does not need to open a sector, and candidate stated that all the traffic situation seemed easy.	Candidate did not use the ruler.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V2 or V3 =>V2	5.	What is more complex: V4 or V6 =>V6
2.	What is more complex: V1 or V3 =>V3	6.	What is more complex: V1 or V5 =>V5
3.	What is more complex: V5 or V6 =>V6	7.	What is more complex: V3 or V5 =>V5
4.	What is more complex: V4 or V5 =>V4	8.	What is more complex: V2 or V5 =>V5

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V1', **1** 'V3', 'V2', **2** 'V5', **3** 'V4', 'V6' **4**]	1. V1=1 2. V3=1.5 3. V2=2 4. V5=3 5. V4=3.5 6. V6=4

Data gathering from ATCO no. 2

Date and time of the experiments:	Candidate:	Years of experience:
23.05.2019. / 15:40 h - 16:45 h	Anonymous no. 2	30

Time required to rank the traffic:	Traffic sample taken and group:
01h05m (without break)	A1-C10 / G1

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A1
3. What is more complex: A1 or A3 =>A3
4. What is more complex: A4 or A5 =>A4
5. What is more complex: A6 or A7 =>A7
6. What is more complex: A5 or A6 =>A6
7. What is more complex: A4 or A6 =>A4
8. What is more complex: A4 or A7 =>A4
9. What is more complex: A2 or A5 =>A2
10. What is more complex: A2 or A6 =>A2
11. What is more complex: A2 or A7 =>A7
12. What is more complex: A1 or A7 =>A1
13. What is more complex: A1 or A4 =>A4
26. What is more complex: A9 or B2 =>A9
27. What is more complex: A9 or B4 =>A9
28. What is more complex: A9 or B3 =>A9
29. What is more complex: A5 or B5 =>A5
30. What is more complex: A5 or A8 =>A5
31. What is more complex: A5 or B2 =>A5
32. What is more complex: A5 or B4 =>B4
33. What is more complex: A6 or B4 =>B4
34. What is more complex: A2 or B4 =>A2
35. What is more complex: A2 or B3 =>A2
36. What is more complex: A2 or A9 =>A9
37. What is more complex: A7 or A9 =>A9
38. What is more complex: A1 or A9 =>A1
51. What is more complex: B6 or B10 =>B10
52. What is more complex: B7 or B10 =>B10
53. What is more complex: B8 or B10 =>B8
54. What is more complex: C3 or C4 =>C4
55. What is more complex: C5 or C6 =>C6
56. What is more complex: C3 or C5 =>C3
57. What is more complex: C3 or C6 =>C3
58. What is more complex: C7 or C8 =>C8
59. What is more complex: C9 or C10 =>C10
60. What is more complex: C7 or C9 =>C9
61. What is more complex: C8 or C9 =>C9
62. What is more complex: C5 or C7 =>C7
63. What is more complex: C6 or C7 =>C6
76. What is more complex: B10 or C6 =>C6
77. What is more complex: B8 or C6 =>B8
78. What is more complex: B8 or C3 =>C3
79. What is more complex: B5 or C5 =>C5
80. What is more complex: A8 or C5 =>C5
81. What is more complex: B2 or C5 =>C5
82. What is more complex: A5 or C5 =>C5
83. What is more complex: A6 or C5 =>C5
84. What is more complex: B4 or C5 =>B4
85. What is more complex: B4 or C7 =>B4
86. What is more complex: B4 or C8 =>B4
87. What is more complex: B4 or B9 =>B9
88. What is more complex: B3 or B9 =>B9

14. What is more complex: A3 or A4 =>A4
15. What is more complex: A8 or A9 =>A9
16. What is more complex: A10 or B1 =>B1
17. What is more complex: A8 or A10 =>A10
18. What is more complex: A9 or A10 =>A10
19. What is more complex: B2 or B3 =>B3
20. What is more complex: B4 or B5 =>B4
21. What is more complex: B2 or B5 =>B2
22. What is more complex: B2 or B4 =>B4
23. What is more complex: B3 or B4 =>B3
24. What is more complex: A8 or B5 =>A8
25. What is more complex: A8 or B2 =>B2

39. What is more complex: A1 or A10 =>A1
40. What is more complex: A1 or B1 =>A1
41. What is more complex: B7 or B8 =>B8
42. What is more complex: B6 or B7 =>B7
43. What is more complex: B9 or B10 =>B10
44. What is more complex: C1 or C2 =>C2
45. What is more complex: B9 or C1 =>C1
46. What is more complex: B10 or C1 =>B10
47. What is more complex: B10 or C2 =>B10
48. What is more complex: B6 or B9 =>B6
49. What is more complex: B6 or C1 =>B6
50. What is more complex: B6 or C2 =>B6

64. What is more complex: C6 or C8 =>C6
65. What is more complex: C6 or C9 =>C9
66. What is more complex: C3 or C9 =>C9
67. What is more complex: C4 or C9 =>C9
68. What is more complex: B9 or C5 =>B9
69. What is more complex: B9 or C7 =>B9
70. What is more complex: B9 or C8 =>B9
71. What is more complex: B9 or C6 =>C6
72. What is more complex: C1 or C6 =>C6
73. What is more complex: C2 or C6 =>C6
74. What is more complex: B6 or C6 =>C6
75. What is more complex: B7 or C6 =>C6

89. What is more complex: A2 or B9 =>B9
90. What is more complex: A7 or B9 =>B9
91. What is more complex: A9 or B9 =>B9
92. What is more complex: A10 or B9 =>B9
93. What is more complex: B1 or B9 =>B9
94. What is more complex: A1 or B9 =>B9
95. What is more complex: A3 or B9 =>B9
96. What is more complex: A4 or B9 =>B9

Ranking results:

['B5', 'A8', 'B2', 'A5', 'A6', **2** 'C5', 'C7', 'C8', 'B4', 'B3', 'A2', 'A7', 'A9', 'A10', 'B1', 'A1', 'A3', **3 NS** 'A4', 'B9', 'C1', 'C2', 'B6', 'B7', 'B10', 'C6', 'B8', **4** 'C3', 'C4', 'C9', 'C10' **5**]

Linearly interpolated scores:

1. B5=1.2
2. A8=1.4
3. B2=1.6
4. A5=1.8

16. A1=2.916667 (2+11/12)
17. A3=3
18. A4=3.111111 (3+1/9)
19. B9=3.222222 (3+2/9)

5.	A6=2	20.	C1=3.333333 (3+3/9)
6.	C5=2.0833333 (2+1/12)	21.	C2=3.444444 (3+4/9)
7.	C7=2.166667 (2+2/12)	22.	B6=3.555556 (3+5/9)
8.	C8=2.25	23.	B7=3.666667 (3+6/9)
9.	B4=2.333333 (2+4/12)	24.	B10=3.777778 (3+7/9)
10.	B3=2.416667 (2+5/12)	25.	C6=3.888889 (3+8/9)
11.	A2=2.5 (2+6/12)	26.	B8=4
12.	A7=2.583333 (2+7/12)	27.	C3=4.25
13.	A9=2.666667 (2+8/12)	28.	C4=4.5
14.	A10=2.75 (2+9/12)	29.	C9=4.75
15.	B1=2.833333 (2+10/12)	30.	C10=5

Comment form the candidate:	**Observations from the moderator:**
No comment.	Candidate did not use the ruler and answered really fast (average time per the pair of traffic situations was 37.55 seconds). Moderator stopped the candidate few times to elaborate the thinking and decision making process. The candidate elaborated that decision behind the more complex situations is based on more tasks that he needs to do in that particular moment.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V2 or V3 =>V2	5.	What is more complex: V1 or V4 =>V4
2.	What is more complex: V1 or V3 =>V3	6.	What is more complex: V3 or V4 =>V3
3.	What is more complex: V5 or V6 =>V6	7.	What is more complex: V3 or V5 =>V5
4.	What is more complex: V4 or V5 =>V5	8.	What is more complex: V2 or V5 =>V5

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V1', **2** 'V4', 'V3', **3** 'V2', **NS** 'V5', 'V6' **4**]	1. V1=2 4. V2=3.333333

	2.	V4=2.5	5.	V5=3.666667
	3.	V3=3	6.	V6=4

Data gathering from ATCO no. 3

Date and time of the experiments:	**Candidate:**	**Years of experience:**
27.05.2019. / 09:40 h - 11:30 h	Anonymous no. 3	1

Time required to rank the traffic:	**Traffic sample taken and group:**
01h50m (12-min break)	A1-C10 / G1

Merge sort candidate’s answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A3
3. What is more complex: A4 or A5 =>A4
4. What is more complex: A6 or A7 =>A7
5. What is more complex: A5 or A6 =>A6
6. What is more complex: A4 or A6 =>A4
7. What is more complex: A4 or A7 =>A4
8. What is more complex: A1 or A5 =>A1
9. What is more complex: A1 or A6 =>A1
10. What is more complex: A1 or A7 =>A1
11. What is more complex: A1 or A4 =>A4
12. What is more complex: A3 or A4 =>A4
13. What is more complex: A2 or A4 =>A4
26. What is more complex: A9 or B5 =>A9
27. What is more complex: A5 or B2 =>A5
28. What is more complex: A5 or B3 =>B3
29. What is more complex: A6 or B3 =>A6
30. What is more complex: A6 or A8 =>A6
31. What is more complex: A6 or B4 =>A6
32. What is more complex: A6 or B5 =>A6
33. What is more complex: A6 or A9 =>A6
34. What is more complex: A6 or B1 =>A6
35. What is more complex: A6 or A10 =>A6
36. What is more complex: B7 or B8 =>B8
37. What is more complex: B6 or B7 =>B7
38. What is more complex: B9 or B10 =>B10
51. What is more complex: C7 or C8 =>C8
52. What is more complex: C9 or C10 =>C9
53. What is more complex: C7 or C10 =>C10
54. What is more complex: C8 or C10 =>C10
55. What is more complex: C5 or C7 =>C7
56. What is more complex: C6 or C7 =>C6
57. What is more complex: C6 or C8 =>C6
58. What is more complex: C6 or C10 =>C6
59. What is more complex: C6 or C9 =>C9
60. What is more complex: C4 or C9 =>C4
61. What is more complex: B6 or C5 =>C5
62. What is more complex: B7 or C5 =>B7
63. What is more complex: B7 or C7 =>B7
76. What is more complex: B4 or C5 =>C5
77. What is more complex: B5 or C5 =>C5
78. What is more complex: A9 or C5 =>C5
79. What is more complex: B1 or C5 =>C5
80. What is more complex: A10 or C5 =>C5
81. What is more complex: A6 or C5 =>C5
82. What is more complex: A7 or C5 =>C5
83. What is more complex: A1 or C5 =>C5
84. What is more complex: A3 or C5 =>C5
85. What is more complex: A2 or C5 =>C5
86. What is more complex: A4 or C5 =>A4
87. What is more complex: A4 or C7 =>A4
88. What is more complex: A4 or B7 =>A4

14. What is more complex: A8 or A9 =>A9

15. What is more complex: A10 or B1 =>A10

16. What is more complex: A8 or B1 =>B1

17. What is more complex: A9 or B1 =>B1

18. What is more complex: B2 or B3 =>B3

19. What is more complex: B4 or B5 =>B5

20. What is more complex: B2 or B4 =>B4

21. What is more complex: B3 or B4 =>B4

22. What is more complex: A8 or B2 =>A8

23. What is more complex: A8 or B3 =>A8

24. What is more complex: A8 or B4 =>B4

25. What is more complex: A9 or B4 =>A9

39. What is more complex: C1 or C2 =>C1

40. What is more complex: B9 or C2 =>B9

41. What is more complex: B9 or C1 =>B9

42. What is more complex: B6 or C2 =>C2

43. What is more complex: B7 or C2 =>C2

44. What is more complex: B8 or C2 =>B8

45. What is more complex: B8 or C1 =>B8

46. What is more complex: B8 or B9 =>B9

47. What is more complex: C3 or C4 =>C3

48. What is more complex: C5 or C6 =>C6

49. What is more complex: C4 or C5 =>C4

50. What is more complex: C4 or C6 =>C4

64. What is more complex: B7 or C8 =>C8

65. What is more complex: C2 or C8 =>C2

66. What is more complex: C2 or C10 =>C10

67. What is more complex: C1 or C10 =>C10

68. What is more complex: B8 or C10 =>C10

69. What is more complex: B9 or C10 =>C10

70. What is more complex: B10 or C10 =>C10

71. What is more complex: B2 or B6 =>B6

72. What is more complex: A5 or B6 =>B6

73. What is more complex: B3 or B6 =>B6

74. What is more complex: A8 or B6 =>A8

75. What is more complex: A8 or C5 =>C5

89. What is more complex: A4 or C8 =>A4

90. What is more complex: A4 or C2 =>C2

Ranking results:

['B2', 'A5', 'B3', 'B6', 'A8', 'B4', 'B5', 'A9', 'B1', 'A10', 'A6', 'A7', 'A1', **2** 'A3', 'A2', 'C5', 'C7', **3** 'B7', 'C8', 'A4', 'C2', 'C1', 'B8', 'B9', **4** 'B10', **NS** 'C10', 'C6', 'C9', 'C4', 'C3' **5**]

Linearly interpolated scores:

1. B2=1.0769231 (1+1/13)
2. A5=1.153846 (1+2/13)
3. B3=1.230769 (1+3/13)

16. C5=2.75
17. C7=3
18. B7=3.142857 (3+1/7)

4. B6=1.307692 (1+4/13)

5. A8=1.384615 (1+5/13)

6. B4=1.461539 (1+6/13)

7. B5=1.538462 (1+7/13)

8. A9=1.615385 (1+8/13)

9. B1=1.692308 (1+9/13)

10. A10=1.769231 (1+10/13)

11. A6=1.846154 (1+11/13)

12. A7=1.923077 (1+12/13)

13. A1=2

14. A3=2.25

15. A2=2.5

19. C8=3.285714 (3+2/7)

20. A4=3.428571 (3+3/7)

21. C2=3.571429 (3+4/7)

22. C1=3.714286 (3+5/7)

23. B8=3.857143 (3+6/7)

24. B9=4

25. B10=4.166667 (4+1/6)

26. C10=4.333333 (4+2/6)

27. C6=4.5 (4+3/6)

28. C9=4.666667 (4+4/6)

29. C4=4.833333 (4+5/6)

30. C3=5

Comment form the candidate:	**Observations from the moderator:**
Didn’t need to use the ruler because he was evaluating the traffic complexity based on how many SEP tools he would use while working.	Nothing to report.

Validation airspace Merge sort candidates answers:

1. What is more complex: V2 or V3 =>V3
2. What is more complex: V1 or V2 =>V2
3. What is more complex: V5 or V6 =>V6
4. What is more complex: V4 or V5 =>V5
5. What is more complex: V1 or V4 =>V4
6. What is more complex: V2 or V4 =>V4
7. What is more complex: V3 or V4 =>V4

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V1', **1** 'V2', 'V3', **2** 'V4', 'V5', **3** 'V6' **4**]	1. V1=1 2. V2=1.5 3. V3=2 4. V4=2.5 5. V5=3 6. V6=4

Data gathering from ATCO no. 4

Date and time of the experiments:	Candidate:	Years of experience:
29.05.2019. / 12:10 h - 16:00 h	Anonymous no. 4	4

Time required to rank the traffic:	Traffic sample taken and group:
03h50m (30-min break)	A11-C16 / G2

Merge sort candidate’s answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A1
3. What is more complex: A1 or A2 =>A2
4. What is more complex: A4 or A11 =>A4
5. What is more complex: A12 or A13 =>A12
6. What is more complex: A11 or A13 =>A13
7. What is more complex: A4 or A13 =>A4
8. What is more complex: A4 or A12 =>A4
9. What is more complex: A3 or A11 =>A11
10. What is more complex: A1 or A11 =>A1
11. What is more complex: A1 or A13 =>A1
12. What is more complex: A1 or A12 =>A1
13. What is more complex: A1 or A4 =>A4
14. What is more complex: A2 or A4 =>A4

26. What is more complex: A16 or B3 =>A16
27. What is more complex: A16 or B4 =>B4
28. What is more complex: B1 or B4 =>B4
29. What is more complex: A14 or B4 =>A14
30. What is more complex: A3 or B11 =>A3
31. What is more complex: A3 or B2 =>A3
32. What is more complex: A3 or B3 =>A3
33. What is more complex: A3 or A16 =>A16
34. What is more complex: A11 or A16 =>A16
35. What is more complex: A13 or A16 =>A16
36. What is more complex: A12 or A16 =>A12
37. What is more complex: A12 or B1 =>A12
38. What is more complex: A12 or B4 =>B4
39. What is more complex: A1 or B4 =>B4

51. What is more complex: B12 or B15 =>B12
52. What is more complex: B12 or C1 =>C1
53. What is more complex: B14 or C1 =>B14
54. What is more complex: B14 or C2 =>C2
55. What is more complex: B13 or C2 =>C2
56. What is more complex: C3 or C4 =>C3
57. What is more complex: C11 or C12 =>C12
58. What is more complex: C4 or C11 =>C4
59. What is more complex: C4 or C12 =>C4
60. What is more complex: C13 or C14 =>C14
61. What is more complex: C15 or C16 =>C16
62. What is more complex: C13 or C15 =>C15
63. What is more complex: C14 or C15 =>C14
64. What is more complex: C14 or C16 =>C16

76. What is more complex: C1 or C15 =>C1
77. What is more complex: C1 or C12 =>C1
78. What is more complex: C1 or C4 =>C1
79. What is more complex: C1 or C14 =>C14
80. What is more complex: B14 or C14 =>C14
81. What is more complex: B13 or C14 =>C14
82. What is more complex: C2 or C14 =>C14
83. What is more complex: B16 or C14 =>C14
84. What is more complex: B11 or B15 =>B15
85. What is more complex: B2 or B15 =>B15
86. What is more complex: B3 or B15 =>B15
87. What is more complex: A3 or B15 =>B15
88. What is more complex: A11 or B15 =>B15
89. What is more complex: A13 or B15 =>A13

15. What is more complex: A14 or A15 =>A15

16. What is more complex: A16 or B1 =>B1

17. What is more complex: A14 or A16 =>A14

18. What is more complex: A14 or B1 =>A14

19. What is more complex: B2 or B3 =>B3

20. What is more complex: B4 or B11 =>B4

21. What is more complex: B2 or B11 =>B2

22. What is more complex: B2 or B4 =>B4

23. What is more complex: B3 or B4 =>B4

24. What is more complex: A16 or B11 =>A16

25. What is more complex: A16 or B2 =>A16

40. What is more complex: A2 or B4 =>B4

41. What is more complex: A4 or B4 =>A4

42. What is more complex: A4 or A14 =>A4

43. What is more complex: A4 or A15 =>A4

44. What is more complex: B13 or B14 =>B13

45. What is more complex: B12 or B14 =>B14

46. What is more complex: B15 or B16 =>B16

47. What is more complex: C1 or C2 =>C2

48. What is more complex: B15 or C1 =>C1

49. What is more complex: B16 or C1 =>B16

50. What is more complex: B16 or C2 =>B16

65. What is more complex: C11 or C13 =>C13

66. What is more complex: C12 or C13 =>C12

67. What is more complex: C12 or C15 =>C12

68. What is more complex: C12 or C14 =>C14

69. What is more complex: C4 or C14 =>C14

70. What is more complex: C3 or C14 =>C3

71. What is more complex: C3 or C16 =>C3

72. What is more complex: B15 or C11 =>C11

73. What is more complex: B12 or C11 =>B12

74. What is more complex: B12 or C13 =>C13

75. What is more complex: C1 or C13 =>C1

90. What is more complex: A13 or C11 =>C11

91. What is more complex: A16 or C11 =>C11

92. What is more complex: B1 or C11 =>C11

93. What is more complex: A12 or C11 =>A12

94. What is more complex: A12 or B12 =>B12

95. What is more complex: A1 or B12 =>B12

96. What is more complex: A2 or B12 =>B12

97. What is more complex: B4 or B12 =>B4

98. What is more complex: B4 or C13 =>C13

99. What is more complex: A14 or C13 =>C13

100. What is more complex: A15 or C13 =>C13

101. What is more complex: A4 or C13 =>C13

Ranking results:

['B11', 'B2', 'B3', 'A3', 'A11', **1** 'B15', 'A13', 'A16', 'B1', 'C11', 'A12', 'A1', 'A2', **2** 'B12', 'B4', 'A14', 'A15', 'A4', 'C13', 'C15', 'C12', 'C4', 'C1', 'B14', 'B13', **3** 'C2', 'B16', 'C14', **4 NS** 'C16', 'C3' **5**]

Linearly interpolated scores:

1. B11=0.2
2. B2=0.4
3. B3=0.6
4. A3=0.8
5. A11=1

16. A14=2.25 (2+3/12)
17. A15=2.333333 (2+4/12)
18. A4=2.416667 (2+5/12)
19. C13=2.5 (2+6/12)
20. C15=2.583333 (2+7/12)

6. B15=1.125	21. C12=2.666667 (2+8/12)
7. A13=1.25	22. C4=2.75 (2+9/12)
8. A16=1.375	23. C1=2.833333 (2+10/12)
9. B1=1.5	24. B14=2.916667 (2+11/12)
10. C11=1.625	25. B13=3
11. A12=1.75	26. C2=3.333333 (3+1/3)
12. A1=1.875	27. B16=3.666667 (3+2/3)
13. A2=2	28. C14=4
14. B12=2.0833333(2+1/12)	29. C16=4.5
15. B4=2.166667 (2+2/12)	30. C3=5
Comment form the candidate: Candidate stated multiple times that it would be better for him if the experiment was a digital comparison of traffic situations and that he had the use of SEP and QDM tools. Also, candidate is stating that he would put C13 in the category 4 of complexity and that he is baffled why it is in the category 3.	**Observations from the moderator:** Candidate is using the ruler and is taking notes to see what he needs to do in each traffic situation. As time progresses the candidate is becoming inconsistent, probably because of fatigue.
Validation airspace Merge sort candidates answers: 1. What is more complex: V8 or V9 =>V9 2. What is more complex: V7 or V8 =>V8 3. What is more complex: V5 or V6 =>V6 4. What is more complex: V10 or V5 =>V10 5. What is more complex: V10 or V6 =>V6	6. What is more complex: V7 or V5 =>V5 7. What is more complex: V8 or V5 =>V8 8. What is more complex: V8 or V10 =>V10 9. What is more complex: V9 or V10 =>V10
Validation Ranking results: ['V7', **1** 'V5', 'V8', **2** 'V9', **3** 'V10', **4 NS** 'V6' **5**]	**Validation Linearly interpolated scores:** 1. V7=1 4. V9=3 2. V5=1.5 5. V10=4

	3.	V8=2	6.	V6=5

Data gathering from ATCO no. 5

Date and time of the experiments:	Candidate:	Years of experience:
30.05.2019. / 16:40 h - 18:50 h	Anonymous no. 5	6

Time required to rank the traffic:	Traffic sample taken and group:
02h10m (13-min break)	A11-C16 / G2

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A3
3. What is more complex: A4 or A11 =>A4
4. What is more complex: A12 or A13 =>A13
5. What is more complex: A11 or A12 =>A12
6. What is more complex: A4 or A12 =>A4
7. What is more complex: A4 or A13 =>A4
8. What is more complex: A1 or A11 =>A11
9. What is more complex: A3 or A11 =>A3
10. What is more complex: A3 or A12 =>A12
11. What is more complex: A2 or A12 =>A2
12. What is more complex: A2 or A13 =>A13
13. What is more complex: A14 or A15 =>A15
14. What is more complex: A16 or B1 =>A16
26. What is more complex: A14 or B3 =>B3
27. What is more complex: A16 or B3 =>A16
28. What is more complex: A16 or B4 =>A16
29. What is more complex: A1 or B11 =>A1
30. What is more complex: A1 or B2 =>A1
31. What is more complex: A1 or B1 =>B1
32. What is more complex: A11 or B1 =>B1
33. What is more complex: A3 or B1 =>A3
34. What is more complex: A3 or A14 =>A3
35. What is more complex: A3 or B3 =>A3
36. What is more complex: A3 or B4 =>A3
37. What is more complex: A3 or A16 =>A3
38. What is more complex: A3 or A15 =>A15
39. What is more complex: A12 or A15 =>A15
51. What is more complex: B14 or B15 =>B14
52. What is more complex: B14 or C1 =>C1
53. What is more complex: B13 or C1 =>C1
54. What is more complex: B12 or C1 =>B12
55. What is more complex: B12 or C2 =>B12
56. What is more complex: B12 or B16 =>B12
57. What is more complex: C3 or C4 =>C4
58. What is more complex: C11 or C12 =>C11
59. What is more complex: C3 or C12 =>C3
60. What is more complex: C3 or C11 =>C3
61. What is more complex: C13 or C14 =>C14
62. What is more complex: C15 or C16 =>C16
63. What is more complex: C13 or C15 =>C13
64. What is more complex: C13 or C16 =>C16
76. What is more complex: C2 or C15 =>C15
77. What is more complex: B16 or C15 =>B16
78. What is more complex: B16 or C12 =>C12
79. What is more complex: B12 or C12 =>B12
80. What is more complex: B12 or C11 =>B12
81. What is more complex: B12 or C13 =>B12
82. What is more complex: B12 or C14 =>C14
83. What is more complex: B11 or B15 =>B11
84. What is more complex: B11 or B14 =>B14
85. What is more complex: B2 or B14 =>B14
86. What is more complex: A1 or B14 =>A1
87. What is more complex: A1 or B13 =>B13
88. What is more complex: A11 or B13 =>B13
89. What is more complex: B1 or B13 =>B13

15. What is more complex: A14 or B1 =>A14

16. What is more complex: A14 or A16 =>A16

17. What is more complex: A15 or A16 =>A15

18. What is more complex: B2 or B3 =>B3

19. What is more complex: B4 or B11 =>B4

20. What is more complex: B2 or B11 =>B2

21. What is more complex: B2 or B4 =>B4

22. What is more complex: B3 or B4 =>B4

23. What is more complex: B1 or B11 =>B1

24. What is more complex: B1 or B2 =>B1

25. What is more complex: B1 or B3 =>B3

40. What is more complex: A2 or A15 =>A15

41. What is more complex: A13 or A15 =>A15

42. What is more complex: A4 or A15 =>A4

43. What is more complex: B13 or B14 =>B13

44. What is more complex: B12 or B14 =>B12

45. What is more complex: B12 or B13 =>B12

46. What is more complex: B15 or B16 =>B16

47. What is more complex: C1 or C2 =>C2

48. What is more complex: B15 or C1 =>C1

49. What is more complex: B16 or C1 =>B16

50. What is more complex: B16 or C2 =>B16

65. What is more complex: C14 or C16 =>C16

66. What is more complex: C12 or C15 =>C12

67. What is more complex: C12 or C13 =>C13

68. What is more complex: C11 or C13 =>C13

69. What is more complex: C3 or C13 =>C3

70. What is more complex: C3 or C14 =>C3

71. What is more complex: C3 or C16 =>C3

72. What is more complex: B15 or C15 =>C15

73. What is more complex: B14 or C15 =>C15

74. What is more complex: B13 or C15 =>C15

75. What is more complex: C1 or C15 =>C15

90. What is more complex: A14 or B13 =>B13

91. What is more complex: B3 or B13 =>B13

92. What is more complex: B4 or B13 =>B13

93. What is more complex: A16 or B13 =>B13

94. What is more complex: A3 or B13 =>B13

95. What is more complex: A12 or B13 =>B13

96. What is more complex: A2 or B13 =>B13

97. What is more complex: A13 or B13 =>B13

98. What is more complex: A15 or B13 =>A15

99. What is more complex: A15 or C1 =>A15

100. What is more complex: A15 or C2 =>A15

101. What is more complex: A15 or C15 =>C15

102. What is more complex: A4 or C15 =>C15

Ranking results:

['B15', 'B11', 'B2', 'B14', 'A1', 'A11', **2** 'B1', 'A14', 'B3', 'B4', 'A16', 'A3', 'A12', 'A2', 'A13', 'B13', 'C1', 'C2', 'A15', **3** 'A4', **NS** 'C15', 'B16', 'C12', 'C11', **4** 'C13', 'B12', 'C14', 'C16', 'C3', 'C4' **5**]

Linearly interpolated scores:

1. B15=1.166667 (1+1/6)

2. B11=1.333333 (1+2/6)

3. B2=1.5 (1+3/6)

4. B14=1.666667 (1+4/6)

16. B13=2.769231 (2+10/13)

17. C1=2.846154 (2+11/13)

18. C2=2.923077 (2+12/13)

19. A15=3

5.	A1=1.833333 (1+5/6)	20.	A4=3.2
6.	A11=2	21.	C15=3.4
7.	B1=2.0769231 (2+1/13)	22.	B16=3.6
8.	A14=2.153846 (2+2/13)	23.	C12=3.8
9.	B3=2.230769 (2+3/13)	24.	C11=4
10.	B4=2.307692 (2+4/13)	25.	C13=4.166667 (4+1/6)
11.	A16=2.384615 (2+5/13)	26.	B12=4.333333 (4+2/6)
12.	A3=2.461539 (2+6/13)	27.	C14=4.5 (4+3/6)
13.	A12=2.538462 (2+7/13)	28.	C16=4.666667 (4+4/6)
14.	A2=2.615385 (2+8/13)	29.	C3=4.833333 (4+5/6)
15.	A13=2.692308 (2+9/13)	30.	C4=5

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that with time seeing the same traffic situation multiple times it is becoming less complex because the solution to the problem was already found. Also, candidate is stating that he would put B12 in the category 4 of complexity instead of 5.	Candidate is using the ruler only at the beginning (first 5 min).

Validation airspace Merge sort candidates answers:

1.	What is more complex: V8 or V9 =>V9	6.	What is more complex: V7 or V5 =>V5
2.	What is more complex: V7 or V8 =>V8	7.	What is more complex: V8 or V5 =>V8
3.	What is more complex: V5 or V6 =>V6	8.	What is more complex: V8 or V10 =>V8
4.	What is more complex: V10 or V5 =>V10	9.	What is more complex: V8 or V6 =>V6
5.	What is more complex: V10 or V6 =>V6	10.	What is more complex: V9 or V6 =>V6

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V7', **2** 'V5', 'V10', **3** 'V8', **NS** 'V9', **4** 'V6' **5**]	1. V7=2 2. V5=2.5 4. V8=3.5 5. V9=4

	3.	V10=3	6.	V6=5

Data gathering from ATCO no. 6

Date and time of the experiments:	Candidate:	Years of experience:
31.05.2019. / 11:35 h - 13:35 h	Anonymous no. 6	18

Time required to rank the traffic:	Traffic sample taken and group:
02h00m (4-min break)	A11-C16 / G2

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A11 =>A4
4. What is more complex: A12 or A13 =>A12
5. What is more complex: A11 or A13 =>A13
6. What is more complex: A4 or A13 =>A4
7. What is more complex: A4 or A12 =>A4
8. What is more complex: A1 or A11 =>A1
9. What is more complex: A1 or A13 =>A1
10. What is more complex: A1 or A12 =>A12
11. What is more complex: A2 or A12 =>A12
12. What is more complex: A3 or A12 =>A3
13. What is more complex: A3 or A4 =>A4
14. What is more complex: A14 or A15 =>A15

26. What is more complex: A14 or B3 =>A14
27. What is more complex: A14 or B4 =>A14
28. What is more complex: A11 or B11 =>A11
29. What is more complex: A11 or B2 =>B2
30. What is more complex: A13 or B2 =>B2
31. What is more complex: A1 or B2 =>A1
32. What is more complex: A1 or B3 =>A1
33. What is more complex: A1 or B4 =>A1
34. What is more complex: A1 or A14 =>A1
35. What is more complex: A1 or B1 =>A1
36. What is more complex: A1 or A16 =>A1
37. What is more complex: A1 or A15 =>A15
38. What is more complex: A2 or A15 =>A15
39. What is more complex: A12 or A15 =>A15

51. What is more complex: B12 or C2 =>C2
52. What is more complex: B14 or C2 =>C2
53. What is more complex: B13 or C2 =>C2
54. What is more complex: C3 or C4 =>C3
55. What is more complex: C11 or C12 =>C12
56. What is more complex: C4 or C11 =>C4
57. What is more complex: C4 or C12 =>C4
58. What is more complex: C13 or C14 =>C14
59. What is more complex: C15 or C16 =>C15
60. What is more complex: C13 or C16 =>C16
61. What is more complex: C14 or C16 =>C16
62. What is more complex: C11 or C13 =>C13
63. What is more complex: C12 or C13 =>C13
64. What is more complex: C4 or C13 =>C13

76. What is more complex: A11 or B15 =>B15
77. What is more complex: A13 or B15 =>B15
78. What is more complex: B2 or B15 =>B15
79. What is more complex: B3 or B15 =>B15
80. What is more complex: B4 or B15 =>B4
81. What is more complex: B4 or C1 =>B4
82. What is more complex: B4 or C11 =>B4
83. What is more complex: B4 or B12 =>B4
84. What is more complex: B4 or B14 =>B4
85. What is more complex: B4 or C12 =>B4
86. What is more complex: B4 or B13 =>B13
87. What is more complex: A14 or B13 =>B13
88. What is more complex: B1 or B13 =>B13
89. What is more complex: A16 or B13 =>B13

15. What is more complex: A16 or B1 =>A16

16. What is more complex: A14 or B1 =>B1

17. What is more complex: A15 or B1 =>A15

18. What is more complex: A15 or A16 =>A15

19. What is more complex: B2 or B3 =>B3

20. What is more complex: B4 or B11 =>B4

21. What is more complex: B2 or B11 =>B2

22. What is more complex: B2 or B4 =>B4

23. What is more complex: B3 or B4 =>B4

24. What is more complex: A14 or B11 =>A14

25. What is more complex: A14 or B2 =>A14

40. What is more complex: A3 or A15 =>A15

41. What is more complex: A4 or A15 =>A4

42. What is more complex: B13 or B14 =>B13

43. What is more complex: B12 or B14 =>B14

44. What is more complex: B15 or B16 =>B16

45. What is more complex: C1 or C2 =>C2

46. What is more complex: B15 or C1 =>C1

47. What is more complex: B16 or C1 =>B16

48. What is more complex: B16 or C2 =>B16

49. What is more complex: B12 or B15 =>B12

50. What is more complex: B12 or C1 =>B12

65. What is more complex: C3 or C13 =>C13

66. What is more complex: B15 or C11 =>C11

67. What is more complex: C1 or C11 =>C11

68. What is more complex: B12 or C11 =>B12

69. What is more complex: B12 or C12 =>C12

70. What is more complex: B14 or C12 =>C12

71. What is more complex: B13 or C12 =>B13

72. What is more complex: B13 or C4 =>C4

73. What is more complex: C2 or C4 =>C4

74. What is more complex: B16 or C4 =>C4

75. What is more complex: B11 or B15 =>B15

90. What is more complex: A1 or B13 =>B13

91. What is more complex: A2 or B13 =>B13

92. What is more complex: A12 or B13 =>B13

93. What is more complex: A3 or B13 =>B13

94. What is more complex: A15 or B13 =>B13

95. What is more complex: A4 or B13 =>A4

96. What is more complex: A4 or C2 =>C2

Ranking results:

['B11', 'A11', **1** 'A13', 'B2', 'B3', 'B15', 'C1', 'C11', 'B12', 'B14', 'C12', 'B4', 'A14', 'B1', 'A16', 'A1', 'A2', 'A12', **2** 'A3', 'A15', 'B13', 'A4', 'C2', 'B16', **3** 'C4', 'C3', 'C13', 'C14', 'C16', 'C15' **4**]

Linearly interpolated scores:

1. B11=0.5
2. A11=1
3. A13=1.0625
4. B2=1.125
5. B3=1.1875
6. B15=1.25
7. C1=1.3125
16. A1=1.875
17. A2=1.9375
18. A12=2
19. A3=2.166667 (2+1/6)
20. A15=2.333333 (2+2/6)
21. B13=2.5 (2+3/6)
22. A4=2.666667 (2+4/6)

8.	C11=1.375	23.	C2=2.833333 (2+5/6)
9.	B12=1.4375	24.	B16=3
10.	B14=1.5	25.	C4=3.166667 (3+1/6)
11.	C12=1.5625	26.	C3=3.333333 (3+2/6)
12.	B4=1.625	27.	C13=3.5 (3+3/6)
13.	A14=1.6875	28.	C14=3.666667 (3+4/6)
14.	B1=1.75	29.	C16=3.833333 (3+5/6)
15.	A16=1.8125	30.	C15=4

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that with time when he is seeing the same traffic situation multiple times it is becoming less complex because he already found a solution to the problem. Also candidate is stating that he would put A13 and B2 in the category 3 of complexity instead of 2.	Candidate does not wish to open a sector.

Validation airspace Merge sort candidates answers:

1.	What is more complex: V8 or V9 =>V8	6.	What is more complex: V7 or V5 =>V5
2.	What is more complex: V7 or V9 =>V9	7.	What is more complex: V9 or V5 =>V5
3.	What is more complex: V5 or V6 =>V6	8.	What is more complex: V8 or V5 =>V8
4.	What is more complex: V10 or V5 =>V10	9.	What is more complex: V8 or V10 =>V8
5.	What is more complex: V10 or V6 =>V6	10.	What is more complex: V8 or V6 =>V6

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V7', **2** 'V9', 'V5', 'V10', **3** 'V8', 'V6' **4**]	1. V7=2 4. V10=3 2. V9=2.333333 5. V8=3.5 3. V5=2.666667 6. V6=4

Data gathering from ATCO no. 7

Date and time of the experiments:	**Candidate:**	**Years of experience:**
04.06.2019. / 09:59 h - 13:08 h	Anonymous no. 7	27

Time required to rank the traffic:	**Traffic sample taken and group:**
03h09m (40-min break)	A17-C12 / G3

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A17 =>A4
4. What is more complex: A18 or A19 =>A19
5. What is more complex: A17 or A18 =>A18
6. What is more complex: A4 or A18 =>A4
7. What is more complex: A4 or A19 =>A4
8. What is more complex: A1 or A17 =>A1
9. What is more complex: A1 or A18 =>A1
10. What is more complex: A1 or A19 =>A1
11. What is more complex: A1 or A4 =>A4
12. What is more complex: A2 or A4 =>A4
13. What is more complex: A3 or A4 =>A4
14. What is more complex: A20 or A21 =>A20

26. What is more complex: A22 or B17 =>B17
27. What is more complex: B1 or B17 =>B17
28. What is more complex: A17 or A21 =>A21
29. What is more complex: A18 or A21 =>A18
30. What is more complex: A18 or A20 =>A18
31. What is more complex: A18 or B3 =>A18
32. What is more complex: A18 or B2 =>A18
33. What is more complex: A18 or A22 =>A22
34. What is more complex: A19 or A22 =>A22
35. What is more complex: A1 or A22 =>A22
36. What is more complex: A2 or A22 =>A22
37. What is more complex: A3 or A22 =>A3
38. What is more complex: A3 or B1 =>B1
39. What is more complex: A4 or B1 =>A4

51. What is more complex: B20 or B21 =>B21
52. What is more complex: C3 or C4 =>C4
53. What is more complex: C17 or C18 =>C18
54. What is more complex: C3 or C17 =>C3
55. What is more complex: C3 or C18 =>C3
56. What is more complex: C19 or C20 =>C19
57. What is more complex: C21 or C22 =>C22
58. What is more complex: C20 or C21 =>C20
59. What is more complex: C20 or C22 =>C22
60. What is more complex: C19 or C22 =>C22
61. What is more complex: C17 or C21 =>C21
62. What is more complex: C18 or C21 =>C21
63. What is more complex: C3 or C21 =>C21
64. What is more complex: C4 or C21 =>C21

76. What is more complex: A21 or B18 =>B18
77. What is more complex: A20 or B18 =>B18
78. What is more complex: B3 or B18 =>B18
79. What is more complex: B2 or B18 =>B18
80. What is more complex: A18 or B18 =>B18
81. What is more complex: A19 or B18 =>B18
82. What is more complex: A1 or B18 =>B18
83. What is more complex: A2 or B18 =>B18
84. What is more complex: A22 or B18 =>B18
85. What is more complex: A3 or B18 =>B18
86. What is more complex: B1 or B18 =>B18
87. What is more complex: B17 or B18 =>B18
88. What is more complex: A4 or B18 =>A4
89. What is more complex: A4 or B19 =>A4

15. What is more complex: A22 or B1 =>B1

16. What is more complex: A21 or A22 =>A22

17. What is more complex: A20 or A22 =>A22

18. What is more complex: B2 or B3 =>B2

19. What is more complex: B4 or B17 =>B4

20. What is more complex: B3 or B17 =>B17

21. What is more complex: B2 or B17 =>B17

22. What is more complex: A21 or B3 =>B3

23. What is more complex: A20 or B3 =>B3

24. What is more complex: A22 or B3 =>A22

25. What is more complex: A22 or B2 =>A22

40. What is more complex: A4 or B17 =>A4

41. What is more complex: A4 or B4 =>B4

42. What is more complex: B19 or B20 =>B20

43. What is more complex: B18 or B19 =>B19

44. What is more complex: B21 or B22 =>B22

45. What is more complex: C1 or C2 =>C2

46. What is more complex: B21 or C1 =>C1

47. What is more complex: B22 or C1 =>B22

48. What is more complex: B22 or C2 =>B22

49. What is more complex: B18 or B21 =>B21

50. What is more complex: B19 or B21 =>B21

65. What is more complex: B18 or C17 =>C17

66. What is more complex: B19 or C17 =>C17

67. What is more complex: B20 or C17 =>C17

68. What is more complex: B21 or C17 =>C17

69. What is more complex: C1 or C17 =>C17

70. What is more complex: C2 or C17 =>C2

71. What is more complex: C2 or C18 =>C2

72. What is more complex: C2 or C3 =>C3

73. What is more complex: B22 or C3 =>B22

74. What is more complex: B22 or C4 =>C4

75. What is more complex: A17 or B18 =>B18

90. What is more complex: A4 or B20 =>B20

91. What is more complex: B4 or B20 =>B20

Ranking results:

['A17', **1** 'A21', 'A20', 'B3', 'B2', 'A18', 'A19', 'A1', **2** 'A2', 'A22', 'A3', 'B1', 'B17', 'B18', 'B19', 'A4', 'B4', **NS** 'B20', 'B21', 'C1', 'C17', **3** 'C18', 'C2', 'C3', 'B22', **4** 'C4', 'C21', 'C20', 'C19', 'C22' **5**]

Linearly interpolated scores:

1. A17=1
2. A21=1.142857 (1+1/7)
3. A20=1.285714 (1+2/7)
4. B3=1.428571 (1+3/7)
5. B2=1.571429 (1+4/7)
6. A18=1.714286 (1+5/7)

16. A4=2.615385 (2+8/13)
17. B4=2.692308 (2+9/13)
18. B20=2.769231 (2+10/13)
19. B21=2.846154 (2+11/13)
20. C1=2.923077 (2+12/13)
21. C17=3

7.	A19=1.857143 (1+6/7)	22.	C18=3.25
8.	A1=2	23.	C2=3.5
9.	A2=2.0769231 (2+1/13)	24.	C3=3.75
10.	A22=2.153846 (2+2/13)	25.	B22=4
11.	A3=2.230769 (2+3/13)	26.	C4=4.2
12.	B1=2.307692 (2+4/13)	27.	C21=4.4
13.	B17=2.384615 (2+5/13)	28.	C20=4.6
14.	B18=2.461539 (2+6/13)	29.	C19=4.8
15.	B19=2.538462 (2+7/13)	30.	C22=5

Comment form the candidate:	**Observations from the moderator:**
No comments from the candidate.	No comments.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V8 or V9 =>V9	6.	What is more complex: V7 or V5 =>V5
2.	What is more complex: V7 or V8 =>V8	7.	What is more complex: V8 or V5 =>V5
3.	What is more complex: V5 or V6 =>V6	8.	What is more complex: V9 or V5 =>V9
4.	What is more complex: V10 or V5 =>V10	9.	What is more complex: V9 or V10 =>V9
5.	What is more complex: V10 or V6 =>V6	10.	What is more complex: V9 or V6 =>V6

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V7', **1** 'V8', **3** NS 'V5', 'V10', **4** 'V9', 'V6' **5**]	1. V7=1 2. V8=3 3. V5=3.5 4. V10=4 5. V9=4.5 6. V6=5

Data gathering from ATCO no. 8

Date and time of the experiments:	Candidate:	Years of experience:
04.06.2019. / 14:00 h - 16:35 h	Anonymous no. 8	12

Time required to rank the traffic:	Traffic sample taken and group:
02h35m (without break)	A17-C22 / G3

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A1
3. What is more complex: A1 or A3 =>A3
4. What is more complex: A4 or A17 =>A4
5. What is more complex: A18 or A19 =>A19
6. What is more complex: A17 or A18 =>A17
7. What is more complex: A17 or A19 =>A17
8. What is more complex: A2 or A18 =>A2
9. What is more complex: A2 or A19 =>A2
10. What is more complex: A2 or A17 =>A17
11. What is more complex: A1 or A17 =>A17
12. What is more complex: A3 or A17 =>A3
13. What is more complex: A3 or A4 =>A4
14. What is more complex: A20 or A21 =>A20

26. What is more complex: A21 or B4 =>B4
27. What is more complex: A20 or B4 =>B4
28. What is more complex: A22 or B4 =>A22
29. What is more complex: A22 or B17 =>B17
30. What is more complex: A18 or B3 =>B3
31. What is more complex: A19 or B3 =>B3
32. What is more complex: A2 or B3 =>B3
33. What is more complex: A1 or B3 =>A1
34. What is more complex: A1 or B2 =>A1
35. What is more complex: A1 or B1 =>B1
36. What is more complex: A17 or B1 =>B1
37. What is more complex: A3 or B1 =>A3
38. What is more complex: A3 or A21 =>A3
39. What is more complex: A3 or A20 =>A3

51. What is more complex: B22 or C1 =>B22
52. What is more complex: B20 or B21 =>B21
53. What is more complex: B18 or B21 =>B21
54. What is more complex: B19 or B21 =>B21
55. What is more complex: C3 or C4 =>C4
56. What is more complex: C17 or C18 =>C18
57. What is more complex: C3 or C17 =>C3
58. What is more complex: C3 or C18 =>C18
59. What is more complex: C4 or C18 =>C4
60. What is more complex: C19 or C20 =>C19
61. What is more complex: C21 or C22 =>C21
62. What is more complex: C20 or C22 =>C22
63. What is more complex: C19 or C22 =>C19
64. What is more complex: C19 or C21 =>C19

76. What is more complex: C2 or C17 =>C2
77. What is more complex: C2 or C22 =>C22
78. What is more complex: C1 or C22 =>C22
79. What is more complex: B22 or C22 =>B22
80. What is more complex: B22 or C3 =>B22
81. What is more complex: B22 or C18 =>B22
82. What is more complex: B22 or C4 =>C4
83. What is more complex: A18 or B20 =>B20
84. What is more complex: A19 or B20 =>B20
85. What is more complex: A2 or B20 =>B20
86. What is more complex: B3 or B20 =>B20
87. What is more complex: B2 or B20 =>B20
88. What is more complex: A1 or B20 =>B20
89. What is more complex: A17 or B20 =>B20

15. What is more complex: A22 or B1 =>A22

16. What is more complex: A21 or B1 =>A21

17. What is more complex: A21 or A22 =>A22

18. What is more complex: A20 or A22 =>A22

19. What is more complex: B2 or B3 =>B2

20. What is more complex: B4 or B17 =>B17

21. What is more complex: B3 or B4 =>B4

22. What is more complex: B2 or B4 =>B4

23. What is more complex: B1 or B3 =>B1

24. What is more complex: B1 or B2 =>B1

25. What is more complex: B1 or B4 =>B4

40. What is more complex: A3 or B4 =>B4

41. What is more complex: A4 or B4 =>A4

42. What is more complex: A4 or A22 =>A4

43. What is more complex: A4 or B17 =>A4

44. What is more complex: B19 or B20 =>B19

45. What is more complex: B18 or B20 =>B18

46. What is more complex: B18 or B19 =>B19

47. What is more complex: B21 or B22 =>B22

48. What is more complex: C1 or C2 =>C1

49. What is more complex: B21 or C2 =>C2

50. What is more complex: B22 or C2 =>B22

65. What is more complex: C17 or C20 =>C17

66. What is more complex: C17 or C22 =>C22

67. What is more complex: C3 or C22 =>C3

68. What is more complex: C3 or C21 =>C21

69. What is more complex: C18 or C21 =>C21

70. What is more complex: C4 or C21 =>C21

71. What is more complex: B20 or C20 =>C20

72. What is more complex: B18 or C20 =>C20

73. What is more complex: B19 or C20 =>C20

74. What is more complex: B21 or C20 =>C20

75. What is more complex: C2 or C20 =>C2

90. What is more complex: B1 or B20 =>B1

91. What is more complex: B1 or B18 =>B18

92. What is more complex: A21 or B18 =>A21

93. What is more complex: A21 or B19 =>A21

94. What is more complex: A21 or B21 =>B21

95. What is more complex: A20 or B21 =>B21

96. What is more complex: A3 or B21 =>B21

97. What is more complex: B4 or B21 =>B4

98. What is more complex: B4 or C20 =>C20

99. What is more complex: A22 or C20 =>C20

100. What is more complex: B17 or C20 =>B17

101. What is more complex: B17 or C17 =>B17

102. What is more complex: B17 or C2 =>B17

103. What is more complex: B17 or C1 =>B17

104. What is more complex: B17 or C22 =>C22

105. What is more complex: A4 or C22 =>C22

Ranking results:

['A18', 'A19', 'A2', 'B3', **1** 'B2', 'A1', 'A17', 'B20', 'B1', 'B18', 'B19', 'A21', 'A20', 'A3', 'B21', **2** 'B4', 'A22', 'C20', 'C17', 'C2', 'C1', **3** 'B17', 'A4', 'C22', **4** 'C3', 'C18', 'B22', 'C4', 'C21', 'C19' **5**]

Linearly interpolated scores:

1.	A18=0.25	16.	B4=2.166667 (2+1/6)
2.	A19=0.5	17.	A22=2.333333 (2+2/6)
3.	A2=0.75	18.	C20=2.5 (2+3/6)
4.	B3=1	19.	C17=2.666667 (2+4/6)
5.	B2=1.0909091 (1+1/11)	20.	C2=2.833333 (2+5/6)
6.	A1=1.181818 (1+2/11)	21.	C1=3
7.	A17=1.272727 (1+3/11)	22.	B17=3.333333 (3+1/3)
8.	B20=1.363636 (1+4/11)	23.	A4=3.666667 (3+2/3)
9.	B1=1.454546 (1+5/11)	24.	C22=4
10.	B18=1.545455 (1+6/11)	25.	C3=4.166667 (4+1/6)
11.	B19=1.636364 (1+7/11)	26.	C18=4.333333 (4+2/6)
12.	A21=1.727273 (1+8/11)	27.	B22=4.5 (4+3/6)
13.	A20=1.818182 (1+9/11)	28.	C4=4.666667 (4+4/6)
14.	A3=1.909091 (1+10/11)	29.	C21=4.833333 (4+5/6)
15.	B21=2	30.	C19=5

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that he believes a mark 5 of complexity is for avoiding the weather, military zones and turbulence.	Candidate does not wish to open a sector.

Validation airspace Merge sort candidate's answers:

1. What is more complex: V8 or V9 =>V9
2. What is more complex: V7 or V8 =>V7
3. What is more complex: V7 or V9 =>V9
4. What is more complex: V5 or V6 =>V6
5. What is more complex: V10 or V5 =>V10
6. What is more complex: V10 or V6 =>V6
7. What is more complex: V8 or V5 =>V8
8. What is more complex: V8 or V10 =>V8
9. What is more complex: V8 or V6 =>V6
10. What is more complex: V7 or V6 =>V6
11. What is more complex: V9 or V6 =>V6

Validation Ranking results:	Validation Linearly interpolated scores:
['V5', 'V10', **2** 'V8', 'V7', **3** 'V9', 'V6' **4**]	1. V5=1.5 2. V10=2 3. V8=2.5 4. V7=3 5. V9=3.5 6. V6=4

Data gathering from ATCO no. 9

Date and time of the experiments:	Candidate:	Years of experience:
06.06.2019. / 10:26 h - 13:45 h	Anonymous no. 9	22

Time required to rank the traffic:	Traffic sample taken and group:
03h19m (30-min break)	A17-C22 / G3

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A3
3. What is more complex: A4 or A17 =>A4
4. What is more complex: A18 or A19 =>A19
5. What is more complex: A17 or A18 =>A17
6. What is more complex: A17 or A19 =>A19
7. What is more complex: A4 or A19 =>A4
8. What is more complex: A1 or A18 =>A1
9. What is more complex: A1 or A17 =>A1
10. What is more complex: A1 or A19 =>A19
11. What is more complex: A3 or A19 =>A19
12. What is more complex: A2 or A19 =>A19
13. What is more complex: A20 or A21 =>A21
14. What is more complex: A22 or B1 =>B1
26. What is more complex: A22 or B17 =>A22
27. What is more complex: A18 or B2 =>B2
28. What is more complex: A17 or B2 =>B2
29. What is more complex: A1 or B2 =>A1
30. What is more complex: A1 or B3 =>A1
31. What is more complex: A1 or A20 =>A20
32. What is more complex: A3 or A20 =>A20
33. What is more complex: A2 or A20 =>A2
34. What is more complex: A2 or A21 =>A2
35. What is more complex: A2 or B4 =>B4
36. What is more complex: A19 or B4 =>B4
37. What is more complex: A4 or B4 =>A4
38. What is more complex: A4 or B17 =>A4
39. What is more complex: A4 or A22 =>A4
51. What is more complex: B19 or C2 =>C2
52. What is more complex: B18 or C2 =>C2
53. What is more complex: C3 or C4 =>C3
54. What is more complex: C17 or C18 =>C18
55. What is more complex: C4 or C17 =>C17
56. What is more complex: C3 or C17 =>C17
57. What is more complex: C19 or C20 =>C19
58. What is more complex: C21 or C22 =>C22
59. What is more complex: C20 or C21 =>C21
60. What is more complex: C19 or C21 =>C21
61. What is more complex: C4 or C20 =>C20
62. What is more complex: C3 or C20 =>C3
63. What is more complex: C3 or C19 =>C3
64. What is more complex: C3 or C21 =>C21
76. What is more complex: B22 or C3 =>C3
77. What is more complex: A18 or B21 =>B21
78. What is more complex: A17 or B21 =>B21
79. What is more complex: B2 or B21 =>B21
80. What is more complex: B3 or B21 =>B21
81. What is more complex: A1 or B21 =>B21
82. What is more complex: A3 or B21 =>B21
83. What is more complex: A20 or B21 =>B21
84. What is more complex: A21 or B21 =>B21
85. What is more complex: A2 or B21 =>B21
86. What is more complex: A19 or B21 =>A19
87. What is more complex: A19 or B20 =>A19
88. What is more complex: A19 or B19 =>A19
89. What is more complex: A19 or B18 =>A19

15. What is more complex: A20 or A22 =>A22

16. What is more complex: A21 or A22 =>A22

17. What is more complex: B2 or B3 =>B3

18. What is more complex: B4 or B17 =>B17

19. What is more complex: B2 or B4 =>B4

20. What is more complex: B3 or B4 =>B4

21. What is more complex: A20 or B2 =>A20

22. What is more complex: A20 or B3 =>A20

23. What is more complex: A20 or B4 =>B4

24. What is more complex: A21 or B4 =>B4

25. What is more complex: A22 or B4 =>A22

40. What is more complex: A4 or B1 =>A4

41. What is more complex: B19 or B20 =>B19

42. What is more complex: B18 or B20 =>B18

43. What is more complex: B18 or B19 =>B18

44. What is more complex: B21 or B22 =>B22

45. What is more complex: C1 or C2 =>C1

46. What is more complex: B21 or C2 =>C2

47. What is more complex: B22 or C2 =>B22

48. What is more complex: B22 or C1 =>B22

49. What is more complex: B20 or B21 =>B20

50. What is more complex: B20 or C2 =>C2

65. What is more complex: C17 or C21 =>C21

66. What is more complex: C18 or C21 =>C21

67. What is more complex: B21 or C4 =>C4

68. What is more complex: B20 or C4 =>C4

69. What is more complex: B19 or C4 =>C4

70. What is more complex: B18 or C4 =>C4

71. What is more complex: C2 or C4 =>C4

72. What is more complex: C1 or C4 =>C4

73. What is more complex: B22 or C4 =>B22

74. What is more complex: B22 or C20 =>B22

75. What is more complex: B22 or C19 =>B22

90. What is more complex: A19 or C2 =>C2

91. What is more complex: B4 or C2 =>C2

92. What is more complex: B17 or C2 =>C2

93. What is more complex: A22 or C2 =>C2

94. What is more complex: B1 or C2 =>C2

95. What is more complex: A4 or C2 =>A4

96. What is more complex: A4 or C1 =>A4

97. What is more complex: A4 or C4 =>A4

98. What is more complex: A4 or C20 =>C20

Ranking results:

['A18', 'A17', 'B2', 'B3', 'A1', 'A3', **1** 'A20', 'A21', 'A2', 'B21', 'B20', 'B19', 'B18', 'A19', **2** 'B4', 'B17', 'A22', 'B1', 'C2', 'C1', **3 NS** 'C4', 'A4', 'C20', 'C19', 'B22', 'C3', **4** 'C17', 'C18', 'C21', 'C22' **5**]

Linearly interpolated scores:

1. A18=0.166667 (0+1/6)
2. A17=0.333333 (0+2/6)
3. B2=0.5 (0+3/6)
4. B3=0.666667 (0+4/6)
5. A1=0.833333 (0+5/6)
6. A3=1

16. B17=2.333333 (2+2/6)
17. A22=2.5 (2+3/6)
18. B1=2.666667 (2+4/6)
19. C2=2.833333 (2+5/6)
20. C1=3
21. C4=4.166667 (4+1/6)

7.	A20=1.125	(1+1/8)	22.	A4=4.333333	(4+2/6)
8.	A21=1.25	(1+2/8)	23.	C20=4.5	(4+3/6)
9.	A2=1.375	(1+3/8)	24.	C19=4.666667	(4+4/6)
10.	B21=1.5	(1+4/8)	25.	B22=4.833333	(4+5/6)
11.	B20=1.625	(1+5/8)	26.	C3=4	
12.	B19=1.75	(1+6/8)	27.	C17=4.25	
13.	B18=1.875	(1+7/8)	28.	C18=4.5	
14.	A19=2		29.	C21=4.75	
15.	B4=2.166667	(2+1/6)	30.	C22=5	

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that when she saw the final ranking in front of her that everything seemed easier and that she would give lower complexity marks because she saw them a lot of times now and she knows how to solve them and they appear a lot easier.	No comment.

Validation airspace Merge sort candidate's answers:

1. What is more complex: V8 or V9 =>V8
2. What is more complex: V7 or V9 =>V9
3. What is more complex: V5 or V6 =>V6
4. What is more complex: V10 or V5 =>V5
5. What is more complex: V7 or V10 =>V10
6. What is more complex: V9 or V10 =>V10
7. What is more complex: V8 or V10 =>V10

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V7', **1** 'V9', 'V8', **2** 'V10', **3 NS** 'V5', **4** 'V6' **5**]	1. V7=1 2. V9=1.5 3. V8=2 4. V10=3 5. V5=4 6. V6=5

Data gathering from ATCO no. 10

Date and time of the experiments:	Candidate:	Years of experience:
14.06.2019. / 09:21 h - 11:30 h	Anonymous no. 10	7

Time required to rank the traffic:	Traffic sample taken and group:
02h09m (15-min break)	A23-C28 / G4

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A23 =>A4
4. What is more complex: A24 or A25 =>A25
5. What is more complex: A23 or A24 =>A23
6. What is more complex: A23 or A25 =>A25
7. What is more complex: A4 or A25 =>A4
8. What is more complex: A1 or A24 =>A1
9. What is more complex: A1 or A23 =>A23
10. What is more complex: A2 or A23 =>A2
11. What is more complex: A2 or A25 =>A25
12. What is more complex: A3 or A25 =>A25
13. What is more complex: A26 or A27 =>A27
14. What is more complex: A28 or B1 =>B1

26. What is more complex: B1 or B3 =>B3
27. What is more complex: A27 or B3 =>A27
28. What is more complex: A27 or B2 =>A27
29. What is more complex: A27 or B4 =>A27
30. What is more complex: A24 or A28 =>A28
31. What is more complex: A1 or A28 =>A28
32. What is more complex: A23 or A28 =>A28
33. What is more complex: A2 or A28 =>A2
34. What is more complex: A2 or B23 =>A2
35. What is more complex: A2 or A26 =>A2
36. What is more complex: A2 or B1 =>A2
37. What is more complex: A2 or B3 =>A2
38. What is more complex: A2 or B2 =>A2
39. What is more complex: A2 or B4 =>B4

51. What is more complex: B27 or C1 =>B27
52. What is more complex: B25 or B28 =>B28
53. What is more complex: B24 or B28 =>B28
54. What is more complex: B26 or B28 =>B26
55. What is more complex: B26 or C2 =>B26
56. What is more complex: B26 or C1 =>B26
57. What is more complex: B26 or B27 =>B27
58. What is more complex: C3 or C4 =>C4
59. What is more complex: C23 or C24 =>C24
60. What is more complex: C3 or C23 =>C3
61. What is more complex: C3 or C24 =>C3
62. What is more complex: C25 or C26 =>C26
63. What is more complex: C27 or C28 =>C28
64. What is more complex: C25 or C27 =>C25

76. What is more complex: C2 or C27 =>C27
77. What is more complex: C1 or C27 =>C27
78. What is more complex: B26 or C27 =>B26
79. What is more complex: B26 or C25 =>C25
80. What is more complex: B27 or C25 =>B27
81. What is more complex: B27 or C23 =>B27
82. What is more complex: B27 or C24 =>B27
83. What is more complex: B27 or C3 =>B27
84. What is more complex: B27 or C4 =>C4
85. What is more complex: A24 or B25 =>B25
86. What is more complex: A1 or B25 =>B25
87. What is more complex: A23 or B25 =>A23
88. What is more complex: A23 or B24 =>B24
89. What is more complex: A28 or B24 =>B24

15. What is more complex: A26 or A28 =>A26

16. What is more complex: A26 or B1 =>B1

17. What is more complex: A27 or B1 =>A27

18. What is more complex: B2 or B3 =>B2

19. What is more complex: B4 or B23 =>B4

20. What is more complex: B3 or B23 =>B3

21. What is more complex: B3 or B4 =>B4

22. What is more complex: B2 or B4 =>B4

23. What is more complex: A28 or B23 =>B23

24. What is more complex: A26 or B23 =>A26

25. What is more complex: A26 or B3 =>B3

40. What is more complex: A3 or B4 =>A3

41. What is more complex: A3 or A27 =>A27

42. What is more complex: A25 or A27 =>A27

43. What is more complex: A4 or A27 =>A4

44. What is more complex: B25 or B26 =>B26

45. What is more complex: B24 or B25 =>B24

46. What is more complex: B24 or B26 =>B26

47. What is more complex: B27 or B28 =>B27

48. What is more complex: C1 or C2 =>C1

49. What is more complex: B28 or C2 =>C2

50. What is more complex: B27 or C2 =>B27

65. What is more complex: C25 or C28 =>C28

66. What is more complex: C26 or C28 =>C26

67. What is more complex: C23 or C27 =>C23

68. What is more complex: C23 or C25 =>C23

69. What is more complex: C23 or C28 =>C28

70. What is more complex: C24 or C28 =>C28

71. What is more complex: C3 or C28 =>C28

72. What is more complex: C4 or C28 =>C28

73. What is more complex: B25 or C27 =>C27

74. What is more complex: B24 or C27 =>C27

75. What is more complex: B28 or C27 =>C27

90. What is more complex: B23 or B24 =>B24

91. What is more complex: A26 or B24 =>B24

92. What is more complex: B1 or B24 =>B24

93. What is more complex: B3 or B24 =>B24

94. What is more complex: B2 or B24 =>B24

95. What is more complex: A2 or B24 =>B24

96. What is more complex: B4 or B24 =>B4

97. What is more complex: B4 or B28 =>B28

98. What is more complex: A3 or B28 =>B28

99. What is more complex: A25 or B28 =>B28

100. What is more complex: A27 or B28 =>B28

101. What is more complex: A4 or B28 =>A4

102. What is more complex: A4 or C2 =>A4

103. What is more complex: A4 or C1 =>A4

104. What is more complex: A4 or C27 =>A4

105. What is more complex: A4 or B26 =>A4

106. What is more complex: A4 or C25 =>A4

107. What is more complex: A4 or C23 =>A4

108. What is more complex: A4 or C24 =>A4

109. What is more complex: A4 or C3 =>A4

110. What is more complex: A4 or B27 =>B27

Ranking results:

['A24', 'A1', 'B25', 'A23', 'A28', 'B23', 'A26', 'B1', 'B3', 'B2', **1** 'A2', 'B24', **2** 'B4', 'A3', 'A25', 'A27', 'B28', 'C2', 'C1', **3** 'C27', 'B26', 'C25', 'C23', 'C24', 'C3', 'A4', 'B27', **4 NS** 'C4', 'C28', 'C26' **5**]

Linearly interpolated scores:

1.	A24=0.1		16.	A27=2.571429	(2+4/7)
2.	A1=0.2		17.	B28=2.714286	(2+5/7)
3.	B25=0.3		18.	C2=2.857143	(2+6/7)
4.	A23=0.4		19.	C1=3	
5.	A28=0.5		20.	C27=3.125	(3+1/8)
6.	B23=0.6		21.	B26=3.25	(3+2/8)
7.	A26=0.7		22.	C25=3.375	(3+3/8)
8.	B1=0.8		23.	C23=3.5	(3+4/8)
9.	B3=0.9		24.	C24=3.625	(3+5/8)
10.	B2=1		25.	C3=3.75	(3+6/8)
11.	A2=1.5		26.	A4=3.875	(3+7/8)
12.	B24=2		27.	B27=4	
13.	B4=2.142857	(2+1/7)	28.	C4=4.333333	(4+1/3)
14.	A3=2.285714	(2+2/7)	29.	C28=4.666667	(4+2/3)
15.	A25=2.428571	(2+3/7)	30.	C26=5	

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that with time the same traffic is becoming less complex.	Candidate does not wish to open a sector on validation airspace.

<table>
<tr><td colspan="2">Validation airspace Merge sort candidate’s answers:
1. What is more complex: V14 or V15 =>V15
2. What is more complex: V13 or V14 =>V13
3. What is more complex: V13 or V15 =>V15
4. What is more complex: V17 or V18 =>V18
5. What is more complex: V16 or V17 =>V16
6. What is more complex: V16 or V18 =>V16
7. What is more complex: V14 or V17 =>V17
8. What is more complex: V13 or V17 =>V17
9. What is more complex: V15 or V17 =>V17</td></tr>
<tr><td>Validation Ranking results:
['V14', 'V13', 2 'V15', 3 'V17', 'V18', 'V16' 4]</td><td>Validation Linearly interpolated scores:
1. V14=1.5
2. V13=2
3. V15=3
4. V17=3.333333
5. V18=3.666667
6. V16=4</td></tr>
</table>

Data gathering from ATCO no. 11

Date and time of the experiments:	Candidate:	Years of experience:
21.06.2019. / 12:14 h - 15:35 h	Anonymous no. 11	8

Time required to rank the traffic:	Traffic sample taken and group:
03h21m (26-min break)	A23-C28 / G4

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A1
3. What is more complex: A1 or A2 =>A2
4. What is more complex: A4 or A23 =>A4
5. What is more complex: A24 or A25 =>A25
6. What is more complex: A23 or A24 =>A23
7. What is more complex: A23 or A25 =>A25
8. What is more complex: A4 or A25 =>A4
9. What is more complex: A3 or A24 =>A24
10. What is more complex: A1 or A24 =>A1
11. What is more complex: A1 or A23 =>A1
12. What is more complex: A1 or A25 =>A25
13. What is more complex: A2 or A25 =>A2
14. What is more complex: A2 or A4 =>A4
26. What is more complex: B1 or B2 =>B1
27. What is more complex: B1 or B3 =>B3
28. What is more complex: A27 or B3 =>A27
29. What is more complex: A27 or B4 =>B4
30. What is more complex: A3 or A28 =>A3
31. What is more complex: A3 or A26 =>A26
32. What is more complex: A24 or A26 =>A26
33. What is more complex: A23 or A26 =>A23
34. What is more complex: A23 or B2 =>A23
35. What is more complex: A23 or B1 =>B1
36. What is more complex: A1 or B1 =>A1
37. What is more complex: A1 or B3 =>A1
38. What is more complex: A1 or A27 =>A27
39. What is more complex: A25 or A27 =>A27
51. What is more complex: B26 or B28 =>B26
52. What is more complex: B26 or C1 =>C1
53. What is more complex: B25 or C1 =>C1
54. What is more complex: B24 or C1 =>B24
55. What is more complex: B24 or C2 =>C2
56. What is more complex: C3 or C4 =>C4
57. What is more complex: C23 or C24 =>C24
58. What is more complex: C3 or C23 =>C3
59. What is more complex: C3 or C24 =>C3
60. What is more complex: C25 or C26 =>C26
61. What is more complex: C27 or C28 =>C28
62. What is more complex: C25 or C27 =>C27
63. What is more complex: C26 or C27 =>C26
64. What is more complex: C26 or C28 =>C26
76. What is more complex: C2 or C23 =>C2
77. What is more complex: C2 or C25 =>C25
78. What is more complex: B27 or C25 =>B27
79. What is more complex: B27 or C27 =>B27
80. What is more complex: B27 or C24 =>C24
81. What is more complex: A28 or B28 =>B28
82. What is more complex: A3 or B28 =>B28
83. What is more complex: A24 or B28 =>B28
84. What is more complex: A26 or B28 =>A26
85. What is more complex: A26 or B26 =>B26
86. What is more complex: B2 or B26 =>B26
87. What is more complex: A23 or B26 =>B26
88. What is more complex: B1 or B26 =>B26
89. What is more complex: B3 or B26 =>B3

15. What is more complex: A26 or A27 =>A27
16. What is more complex: A28 or B1 =>B1
17. What is more complex: A26 or A28 =>A26
18. What is more complex: A26 or B1 =>B1
19. What is more complex: A27 or B1 =>A27
20. What is more complex: B2 or B3 =>B3
21. What is more complex: B4 or B23 =>B23
22. What is more complex: B2 or B4 =>B4
23. What is more complex: B3 or B4 =>B4
24. What is more complex: A28 or B2 =>B2
25. What is more complex: A26 or B2 =>B2
40. What is more complex: A2 or A27 =>A2
41. What is more complex: A2 or B4 =>B4
42. What is more complex: A4 or B4 =>B4
43. What is more complex: B25 or B26 =>B25
44. What is more complex: B24 or B26 =>B24
45. What is more complex: B24 or B25 =>B24
46. What is more complex: B27 or B28 =>B27
47. What is more complex: C1 or C2 =>C2
48. What is more complex: B28 or C1 =>C1
49. What is more complex: B27 or C1 =>B27
50. What is more complex: B27 or C2 =>B27
65. What is more complex: C23 or C25 =>C25
66. What is more complex: C24 or C25 =>C24
67. What is more complex: C24 or C27 =>C24
68. What is more complex: C24 or C28 =>C28
69. What is more complex: C3 or C28 =>C28
70. What is more complex: C4 or C28 =>C28
71. What is more complex: B28 or C23 =>C23
72. What is more complex: B26 or C23 =>C23
73. What is more complex: B25 or C23 =>C23
74. What is more complex: C1 or C23 =>C23
75. What is more complex: B24 or C23 =>C23
90. What is more complex: B3 or B25 =>B25
91. What is more complex: A1 or B25 =>A1
92. What is more complex: A1 or C1 =>C1
93. What is more complex: A25 or C1 =>A25
94. What is more complex: A25 or B24 =>B24
95. What is more complex: A27 or B24 =>A27
96. What is more complex: A27 or C23 =>C23
97. What is more complex: A2 or C23 =>C23
98. What is more complex: A4 or C23 =>A4
99. What is more complex: A4 or C2 =>C2
100. What is more complex: B4 or C2 =>C2
101. What is more complex: B23 or C2 =>C2

Ranking results:

['A28', 'A3', 'A24', 'B28', 'A26', 'B2', 'A23', 'B1', **1** 'B26', 'B3', 'B25', 'A1', 'C1', 'A25', 'B24', 'A27', 'A2', **2** 'C23', 'A4', 'B4', 'B23', 'C2', **3** 'C25', 'C27', 'B27', 'C24', **NS 4** 'C3', 'C4', 'C28', 'C26' **5**]

Linearly interpolated scores:

1. A28=0.125 (0+1/8)
2. A3=0.25 (0+2/8)
3. A24=0.375 (0+3/8)
4. B28=0.5 (0+4/8)
5. A26=0.625 (0+5/8)
16. A27=1.888889 (1+8/9)
17. A2=2
18. C23=2.2
19. A4=2.4
20. B4=2.6

6.	B2=0.75	(0+6/8)	21.	B23=2.8
7.	A23=0.875	(0+7/8)	22.	C2=3
8.	B1=1		23.	C25=3.25
9.	B26=1.111111	(1+1/9)	24.	C27=3.5
10.	B3=1.222222	(1+2/9)	25.	B27=3.75
11.	B25=1.333333	(1+3/9)	26.	C24=4
12.	A1=1.444444	(1+4/9)	27.	C3=4.25
13.	C1=1.555556	(1+5/9)	28.	C4=4.5
14.	A25=1.666667	(1+6/9)	29.	C28=4.75
15.	B24=1.777778	(1+7/9)	30.	C26=5

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that this is interesting method and approach for determining complexity.	Candidate is using the ruler one quarter of a time. Candidate is showing signs of fatigue and is making questionable complexity rankings at the end. For example, A1 more complex then B25 and long thinking between A1 and C1 and as a result candidate is unsure if C1 is more complex then A1.

Validation airspace Merge sort candidate's answers:

1. What is more complex: V14 or V15 =>V14
2. What is more complex: V13 or V15 =>V15
3. What is more complex: V17 or V18 =>V18
4. What is more complex: V16 or V17 =>V16
5. What is more complex: V16 or V18 =>V18
6. What is more complex: V13 or V17 =>V17
7. What is more complex: V15 or V17 =>V15
8. What is more complex: V15 or V16 =>V16
9. What is more complex: V14 or V16 =>V16

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V13', **2** 'V17', 'V15', 'V14', **3 NS** 'V16', 'V18' **5**]	1. V13=2 2. V17=2.333333 4. V14=3 5. V16=4.5

	3.	V15=2.666667	6.	V18=5

Data gathering from ATCO no. 12

Date and time of the experiments:	Candidate:	Years of experience:
24.06.2019. / 10:00 h - 12:09 h	Anonymous no. 12	9

Time required to rank the traffic:	Traffic sample taken and group:
02h17m (18-min break)	A23-C28 / G4

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A1
3. What is more complex: A1 or A2 =>A2
4. What is more complex: A4 or A23 =>A4
5. What is more complex: A24 or A25 =>A25
6. What is more complex: A23 or A24 =>A24
7. What is more complex: A4 or A24 =>A4
8. What is more complex: A4 or A25 =>A4
9. What is more complex: A3 or A23 =>A23
10. What is more complex: A1 or A23 =>A23
11. What is more complex: A2 or A23 =>A2
12. What is more complex: A2 or A24 =>A24
13. What is more complex: A26 or A27 =>A27
14. What is more complex: A28 or B1 =>B1
26. What is more complex: A27 or B2 =>A27
27. What is more complex: A27 or B3 =>A27
28. What is more complex: A27 or B4 =>B4
29. What is more complex: A3 or A28 =>A28
30. What is more complex: A1 or A28 =>A28
31. What is more complex: A23 or A28 =>A28
32. What is more complex: A2 or A28 =>A28
33. What is more complex: A24 or A28 =>A28
34. What is more complex: A25 or A28 =>A25
35. What is more complex: A25 or B1 =>A25
36. What is more complex: A25 or A26 =>A26
37. What is more complex: A4 or A26 =>A4
38. What is more complex: A4 or B23 =>A4
39. What is more complex: A4 or B2 =>A4
51. What is more complex: B25 or B28 =>B28
52. What is more complex: B24 or B28 =>B24
53. What is more complex: B24 or B27 =>B24
54. What is more complex: B24 or C2 =>B24
55. What is more complex: B24 or C1 =>B24
56. What is more complex: C3 or C4 =>C3
57. What is more complex: C23 or C24 =>C23
58. What is more complex: C4 or C24 =>C4
59. What is more complex: C4 or C23 =>C4
60. What is more complex: C25 or C26 =>C26
61. What is more complex: C27 or C28 =>C28
62. What is more complex: C25 or C27 =>C25
63. What is more complex: C25 or C28 =>C25
64. What is more complex: C24 or C27 =>C24
76. What is more complex: C1 or C23 =>C23
77. What is more complex: B24 or C23 =>B24
78. What is more complex: B24 or C4 =>C4
79. What is more complex: A3 or B26 =>B26
80. What is more complex: A1 or B26 =>A1
81. What is more complex: A1 or B25 =>B25
82. What is more complex: A23 or B25 =>B25
83. What is more complex: A2 or B25 =>B25
84. What is more complex: A24 or B25 =>B25
85. What is more complex: A28 or B25 =>B25
86. What is more complex: B1 or B25 =>B25
87. What is more complex: A25 or B25 =>A25
88. What is more complex: A25 or B28 =>B28
89. What is more complex: A26 or B28 =>B28

15. What is more complex: A26 or A28 =>A26

16. What is more complex: A26 or B1 =>A26

17. What is more complex: B2 or B3 =>B3

18. What is more complex: B4 or B23 =>B4

19. What is more complex: B2 or B23 =>B2

20. What is more complex: B2 or B4 =>B4

21. What is more complex: B3 or B4 =>B4

22. What is more complex: A28 or B23 =>B23

23. What is more complex: B1 or B23 =>B23

24. What is more complex: A26 or B23 =>B23

25. What is more complex: A27 or B23 =>A27

40. What is more complex: A4 or B3 =>A4

41. What is more complex: A4 or A27 =>A4

42. What is more complex: A4 or B4 =>B4

43. What is more complex: B25 or B26 =>B25

44. What is more complex: B24 or B26 =>B24

45. What is more complex: B24 or B25 =>B24

46. What is more complex: B27 or B28 =>B27

47. What is more complex: C1 or C2 =>C1

48. What is more complex: B28 or C2 =>C2

49. What is more complex: B27 or C2 =>C2

50. What is more complex: B26 or B28 =>B28

65. What is more complex: C24 or C28 =>C28

66. What is more complex: C23 or C28 =>C28

67. What is more complex: C4 or C28 =>C28

68. What is more complex: C3 or C28 =>C28

69. What is more complex: B26 or C27 =>C27

70. What is more complex: B25 or C27 =>C27

71. What is more complex: B28 or C27 =>C27

72. What is more complex: B27 or C27 =>C27

73. What is more complex: C2 or C27 =>C2

74. What is more complex: C2 or C24 =>C24

75. What is more complex: C1 or C24 =>C1

90. What is more complex: B23 or B28 =>B23

91. What is more complex: B23 or B27 =>B27

92. What is more complex: B2 or B27 =>B27

93. What is more complex: B3 or B27 =>B27

94. What is more complex: A27 or B27 =>B27

95. What is more complex: A4 or B27 =>B27

96. What is more complex: B4 or B27 =>B4

97. What is more complex: B4 or C27 =>C27

Ranking results:

['A3', 'B26', **1** 'A1', 'A23', 'A2', 'A24', 'A28', 'B1', 'B25', **2** 'A25', 'A26', 'B28', 'B23', 'B2', 'B3', 'A27', 'A4', **3 NS** 'B27', 'B4', 'C27', 'C2', 'C24', 'C1', **4** 'C23', 'B24', 'C4', 'C3', 'C28', 'C25', 'C26' **5**]

Linearly interpolated scores:

1. A3=0.5
2. B26=1
3. A1=1.142857 (1+1/7)
4. A23=1.285714 (1+2/7)
5. A2=1.428571 (1+3/7)
6. A24=1.571429 (1+4/7)

16. A27=2.875 (2+7/8)
17. A4=3
18. B27=3.166667 (3+1/6)
19. B4=3.333333 (3+2/6)
20. C27=3.5 (3+3/6)
21. C2=3.666667 (3+4/6)

7.	A28=1.714286 (1+5/7)	22.	C24=3.833333 (3+5/6)
8.	B1=1.857143 (1+6/7)	23.	C1=4
9.	B25=2	24.	C23=4.142857 (4+1/7)
10.	A25=2.125 (2+1/8)	25.	B24=4.285714 (4+2/7)
11.	A26=2.25 (2+2/8)	26.	C4=4.428571 (4+3/7)
12.	B28=2.375 (2+3/8)	27.	C3=4.571429 (4+4/7)
13.	B23=2.5 (2+4/8)	28.	C28=4.714286 (4+5/7)
14.	B2=2.625 (2+5/8)	29.	C25=4.857143 (4+6/7)
15.	B3=2.75 (2+6/8)	30.	C26=5

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that he would switch C27 with C25.	Candidate is using the ruler just to see direction of a flight.

Validation airspace Merge sort candidate's answers:

1. What is more complex: V14 or V15 =>V15
2. What is more complex: V13 or V14 =>V13
3. What is more complex: V13 or V15 =>V15
4. What is more complex: V17 or V18 =>V18
5. What is more complex: V16 or V17 =>V16
6. What is more complex: V16 or V18 =>V18
7. What is more complex: V14 or V17 =>V17
8. What is more complex: V13 or V17 =>V17
9. What is more complex: V15 or V17 =>V17

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V14', 'V13', **2** 'V15', **3 NS** 'V17', 'V16', **4** 'V18' **5**]	1. V14=1.5 2. V13=2 3. V15=3 4. V17=3.5 5. V16=4 6. V18=5

Data gathering from ATCO no. 13

Date and time of the experiments:	**Candidate:**	**Years of experience:**
27.06.2019. / 10:50 h - 13:10 h	Anonymous no. 13	9

Time required to rank the traffic:	**Traffic sample taken and group:**
02h20m (5-min break)	A29-C34 / G5

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A3
3. What is more complex: A4 or A29 =>A4
4. What is more complex: A30 or A31 =>A30
5. What is more complex: A29 or A31 =>A29
6. What is more complex: A29 or A30 =>A30
7. What is more complex: A4 or A30 =>A4
8. What is more complex: A1 or A31 =>A1
9. What is more complex: A1 or A29 =>A29
10. What is more complex: A3 or A29 =>A3
11. What is more complex: A3 or A30 =>A30
12. What is more complex: A2 or A30 =>A2
13. What is more complex: A2 or A4 =>A4
14. What is more complex: A32 or A33 =>A33

26. What is more complex: B1 or B4 =>B4
27. What is more complex: A34 or B4 =>A34
28. What is more complex: A34 or B29 =>B29
29. What is more complex: A33 or B29 =>B29
30. What is more complex: A31 or B2 =>A31
31. What is more complex: A31 or B3 =>A31
32. What is more complex: A31 or A32 =>A31
33. What is more complex: A31 or B1 =>A31
34. What is more complex: A31 or B4 =>B4
35. What is more complex: A1 or B4 =>B4
36. What is more complex: A29 or B4 =>B4
37. What is more complex: A3 or B4 =>B4
38. What is more complex: A30 or B4 =>B4
39. What is more complex: A2 or B4 =>B4

51. What is more complex: B30 or B34 =>B34
52. What is more complex: B31 or B34 =>B31
53. What is more complex: B31 or C1 =>B31
54. What is more complex: B31 or C2 =>C2
55. What is more complex: B32 or C2 =>B32
56. What is more complex: B32 or B33 =>B32
57. What is more complex: C3 or C4 =>C3
58. What is more complex: C29 or C30 =>C29
59. What is more complex: C4 or C30 =>C4
60. What is more complex: C4 or C29 =>C4
61. What is more complex: C31 or C32 =>C31
62. What is more complex: C33 or C34 =>C33
63. What is more complex: C32 or C34 =>C32
64. What is more complex: C32 or C33 =>C32

76. What is more complex: B33 or C34 =>C34
77. What is more complex: B32 or C34 =>C34
78. What is more complex: B2 or B30 =>B30
79. What is more complex: B3 or B30 =>B30
80. What is more complex: A32 or B30 =>B30
81. What is more complex: B1 or B30 =>B30
82. What is more complex: A31 or B30 =>A31
83. What is more complex: A31 or B34 =>A31
84. What is more complex: A31 or C1 =>C1
85. What is more complex: A1 or C1 =>C1
86. What is more complex: A29 or C1 =>C1
87. What is more complex: A3 or C1 =>C1
88. What is more complex: A30 or C1 =>C1
89. What is more complex: A2 or C1 =>C1

15. What is more complex: A34 or B1 =>A34

16. What is more complex: A32 or B1 =>B1

17. What is more complex: A33 or B1 =>A33

18. What is more complex: A33 or A34 =>A33

19. What is more complex: B2 or B3 =>B3

20. What is more complex: B4 or B29 =>B29

21. What is more complex: B2 or B4 =>B4

22. What is more complex: B3 or B4 =>B4

23. What is more complex: A32 or B2 =>A32

24. What is more complex: A32 or B3 =>A32

25. What is more complex: A32 or B4 =>B4

40. What is more complex: A4 or B4 =>A4

41. What is more complex: A4 or A34 =>A4

42. What is more complex: A4 or A33 =>A4

43. What is more complex: A4 or B29 =>A4

44. What is more complex: B31 or B32 =>B32

45. What is more complex: B30 or B31 =>B31

46. What is more complex: B33 or B34 =>B33

47. What is more complex: C1 or C2 =>C2

48. What is more complex: B34 or C1 =>C1

49. What is more complex: B33 or C1 =>B33

50. What is more complex: B33 or C2 =>B33

65. What is more complex: C30 or C34 =>C30

66. What is more complex: C30 or C33 =>C33

67. What is more complex: C29 or C33 =>C29

68. What is more complex: C29 or C32 =>C32

69. What is more complex: C4 or C32 =>C32

70. What is more complex: C3 or C32 =>C32

71. What is more complex: B30 or C34 =>C34

72. What is more complex: B34 or C34 =>C34

73. What is more complex: C1 or C34 =>C34

74. What is more complex: B31 or C34 =>C34

75. What is more complex: C2 or C34 =>C34

90. What is more complex: B4 or C1 =>B4

91. What is more complex: B4 or B31 =>B4

92. What is more complex: B4 or C2 =>C2

93. What is more complex: A34 or C2 =>C2

94. What is more complex: A33 or C2 =>C2

95. What is more complex: B29 or C2 =>C2

96. What is more complex: A4 or C2 =>A4

97. What is more complex: A4 or B33 =>A4

98. What is more complex: A4 or B32 =>A4

99. What is more complex: A4 or C34 =>C34

Ranking results:

['B2', 'B3', **1** 'A32', 'B1', 'B30', 'B34', **2** 'A31', 'A1', 'A29', 'A3', 'A30', 'A2', **NS** 'C1', 'B31', 'B4', 'A34', 'A33', 'B29', **3** 'C2', 'B33', 'B32', 'A4', **4** 'C34', 'C30', 'C33', 'C29', 'C4', 'C3', 'C32', 'C31' **5**]

Linearly interpolated scores:

1. B2=0.5
2. B3=1
3. A32=1.25
4. B1=1.5
5. B30=1.75
6. B34=2

16. A34=2.833333 (2+10/12)
17. A33=2.916667 (2+11/12)
18. B29=3
19. C2=3.25
20. B33=3.5
21. B32=3.75

7.	A31=2.083333	(2+1/12)	22.	A4=4	
8.	A1=2.166667	(2+2/12)	23.	C34=4.125	(4+1/8)
9.	A29=2.25	(2+3/12)	24.	C30=4.25	(4+2/8)
10.	A3=2.333333	(2+4/12)	25.	C33=4.375	(4+3/8)
11.	A30=2.416667	(2+5/12)	26.	C29=4.5	(4+4/8)
12.	A2=2.5	(2+6/12)	27.	C4=4.625	(4+5/8)
13.	C1=2.583333	(2+7/12)	28.	C3=4.75	(4+6/8)
14.	B31=2.666667	(2+8/12)	29.	C32=4.875	(4+7/8)
15.	B4=2.75	(2+9/12)	30.	C31=5	

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that the aircraft should have 1-minute vector instead of fix 4 NM vector. Candidate would switch V20 with V19.	Nothing to report.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V20 or V21 =>V21	6.	What is more complex: V19 or V23 =>V19
2.	What is more complex: V19 or V20 =>V20	7.	What is more complex: V19 or V22 =>V19
3.	What is more complex: V23 or V24 =>V24	8.	What is more complex: V19 or V24 =>V24
4.	What is more complex: V22 or V23 =>V22	9.	What is more complex: V20 or V24 =>V24
5.	What is more complex: V22 or V24 =>V24	10.	What is more complex: V21 or V24 =>V24

Validation Ranking results:

['V23', 'V22', **2** 'V19', 'V20', 'V21', **3 NS** 'V24' **4**]

Validation Linearly interpolated scores:

1.	V23=1.5	4.	V20=2.666667
2.	V22=2	5.	V21=3
3.	V19=2.333333	6.	V24=4

Data gathering from ATCO no. 14

<table>
<tr><td colspan="2">Date and time of the experiments:

03.07.2019. / 10:12 h - 12:55 h</td><td colspan="2">Candidate:

Anonymous no. 14</td><td>Years of experience:

23</td></tr>
<tr><td colspan="3">Time required to rank the traffic:

02h43m (15-min break)</td><td colspan="2">Traffic sample taken and group:

A29-C34 / G5</td></tr>
</table>

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A1
3. What is more complex: A1 or A3 =>A3
4. What is more complex: A4 or A29 =>A4
5. What is more complex: A30 or A31 =>A31
6. What is more complex: A29 or A30 =>A30
7. What is more complex: A4 or A30 =>A4
8. What is more complex: A4 or A31 =>A4
9. What is more complex: A2 or A29 =>A29
10. What is more complex: A1 or A29 =>A1
11. What is more complex: A1 or A30 =>A1
12. What is more complex: A1 or A31 =>A31
13. What is more complex: A3 or A31 =>A31
14. What is more complex: A32 or A33 =>A32
26. What is more complex: A33 or B29 =>A33
27. What is more complex: A33 or B4 =>A33
28. What is more complex: A33 or B2 =>A33
29. What is more complex: A2 or B3 =>B3
30. What is more complex: A29 or B3 =>B3
31. What is more complex: A30 or B3 =>A30
32. What is more complex: A30 or B1 =>A30
33. What is more complex: A30 or B29 =>A30
34. What is more complex: A30 or B4 =>B4
35. What is more complex: A1 or B4 =>B4
36. What is more complex: A3 or B4 =>B4
37. What is more complex: A31 or B4 =>A31
38. What is more complex: A31 or B2 =>A31
39. What is more complex: A31 or A33 =>A33
51. What is more complex: B31 or C2 =>C2
52. What is more complex: B32 or C2 =>B32
53. What is more complex: B32 or C1 =>B32
54. What is more complex: B32 or B33 =>B32
55. What is more complex: C3 or C4 =>C3
56. What is more complex: C29 or C30 =>C30
57. What is more complex: C4 or C29 =>C4
58. What is more complex: C4 or C30 =>C30
59. What is more complex: C3 or C30 =>C30
60. What is more complex: C31 or C32 =>C32
61. What is more complex: C33 or C34 =>C33
62. What is more complex: C31 or C34 =>C31
63. What is more complex: C31 or C33 =>C33
64. What is more complex: C32 or C33 =>C32
76. What is more complex: B33 or C29 =>B33
77. What is more complex: B33 or C34 =>C34
78. What is more complex: B32 or C34 =>C34
79. What is more complex: A2 or B30 =>A2
80. What is more complex: A2 or B34 =>B34
81. What is more complex: A29 or B34 =>B34
82. What is more complex: B3 or B34 =>B34
83. What is more complex: B1 or B34 =>B34
84. What is more complex: B29 or B34 =>B34
85. What is more complex: A30 or B34 =>B34
86. What is more complex: A1 or B34 =>B34
87. What is more complex: A3 or B34 =>B34
88. What is more complex: B4 or B34 =>B34
89. What is more complex: B2 or B34 =>B34

15. What is more complex: A34 or B1 =>A34
16. What is more complex: A33 or B1 =>A33
17. What is more complex: A33 or A34 =>A34
18. What is more complex: A32 or A34 =>A34
19. What is more complex: B2 or B3 =>B2
20. What is more complex: B4 or B29 =>B4
21. What is more complex: B3 or B29 =>B29
22. What is more complex: B2 or B29 =>B2
23. What is more complex: B2 or B4 =>B2
24. What is more complex: B1 or B3 =>B1
25. What is more complex: B1 or B29 =>B29

40. What is more complex: A4 or A33 =>A4
41. What is more complex: A4 or A32 =>A32
42. What is more complex: B31 or B32 =>B32
43. What is more complex: B30 or B31 =>B31
44. What is more complex: B33 or B34 =>B33
45. What is more complex: C1 or C2 =>C1
46. What is more complex: B34 or C2 =>C2
47. What is more complex: B33 or C2 =>B33
48. What is more complex: B33 or C1 =>B33
49. What is more complex: B30 or B34 =>B34
50. What is more complex: B31 or B34 =>B31

65. What is more complex: C29 or C34 =>C34
66. What is more complex: C4 or C34 =>C4
67. What is more complex: C4 or C31 =>C4
68. What is more complex: C4 or C33 =>C4
69. What is more complex: C4 or C32 =>C32
70. What is more complex: C3 or C32 =>C3
71. What is more complex: B30 or C29 =>C29
72. What is more complex: B34 or C29 =>C29
73. What is more complex: B31 or C29 =>C29
74. What is more complex: C2 or C29 =>C29
75. What is more complex: C1 or C29 =>C29

90. What is more complex: A31 or B34 =>A31
91. What is more complex: A31 or B31 =>A31
92. What is more complex: A31 or C2 =>C2
93. What is more complex: A33 or C2 =>C2
94. What is more complex: A4 or C2 =>C2
95. What is more complex: A32 or C2 =>A32
96. What is more complex: A32 or C1 =>C1
97. What is more complex: A34 or C1 =>A34
98. What is more complex: A34 or C29 =>C29

Ranking results:

['B30', **1** 'A2', 'A29', 'B3', 'B1', 'B29', 'A30', 'A1', 'A3', 'B4', 'B2', 'B34', 'B31', **2** 'A31', 'A33', 'A4', 'C2', 'A32', 'C1', 'A34', 'C29', **3 NS** 'B33', 'B32', 'C34', **4** 'C31', 'C33', 'C4', 'C32', 'C3', 'C30' **5**]

Linearly interpolated scores:

1. B30=1
2. A2=1.083333 (1+1/12)
3. A29=1.166667 (1+2/12)
4. B3=1.25 (1+3/12)
5. B1=1.333333 (1+4/12)
6. B29=1.416667 (1+5/12)

16. A4=2.375 (2+3/8)
17. C2=2.5 (2+4/8)
18. A32=2.625 (2+5/8)
19. C1=2.75 (2+6/8)
20. A34=2.875 (2+7/8)
21. C29=3

7.	A30=1.5	(1+6/12)	22.	B33=3.333333	(3+1/3)
8.	A1=1.583333	(1+7/12)	23.	B32=3.666667	(3+2/3)
9.	A3=1.666667	(1+8/12)	24.	C34=4	
10.	B4=1.75	(1+9/12)	25.	C31=4.166667	(4+1/6)
11.	B2=1.833333	(1+10/12)	26.	C33=4.333333	(4+2/6)
12.	B34=1.916667	(1+11/12)	27.	C4=4.5	(4+3/6)
13.	B31=2		28.	C32=4.666667	(4+4/6)
14.	A31=2.125	(2+1/8)	29.	C3=4.833333	(4+5/6)
15.	A33=2.25	(2+2/8)	30.	C30=5	

Comment form the candidate:	**Observations from the moderator:**
Candidate would switch A32 to the category 1. A32 is more complex then C2 because it has more FL to cover and green aircraft to the right need to be in constant watch for their speeds.	Candidate is pressing with her fingers on the aircraft out of habit to see the prolonged track of the aircraft. Also, candidate is not using the ruler.

Validation airspace Merge sort candidate's answers:

1. What is more complex: V20 or V21 =>V21
2. What is more complex: V19 or V20 =>V19
3. What is more complex: V19 or V21 =>V21
4. What is more complex: V23 or V24 =>V24
5. What is more complex: V22 or V23 =>V22
6. What is more complex: V22 or V24 =>V22
7. What is more complex: V20 or V23 =>V23
8. What is more complex: V19 or V23 =>V19
9. What is more complex: V19 or V24 =>V24
10. What is more complex: V21 or V24 =>V24

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V20', **1** 'V23', 'V19', **2** 'V21', **3 NS** 'V24', 'V22' **4**]	1. 'V20=1 2. V23=1.5 3. V19=2 4. V21=3 5. V24=3.5 6. V22=4

Data gathering from ATCO no. 15

Date and time of the experiments:	**Candidate:**	**Years of experience:**
04.07.2019. / 08:55 h - 10:50 h	Anonymous no. 15	28

Time required to rank the traffic:	**Traffic sample taken and group:**
01h55m (15-min break)	A29-C34 / G5

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A3
3. What is more complex: A4 or A29 =>A4
4. What is more complex: A30 or A31 =>A31
5. What is more complex: A29 or A30 =>A30
6. What is more complex: A4 or A30 =>A4
7. What is more complex: A4 or A31 =>A4
8. What is more complex: A1 or A29 =>A29
9. What is more complex: A3 or A29 =>A3
10. What is more complex: A3 or A30 =>A30
11. What is more complex: A2 or A30 =>A30
12. What is more complex: A32 or A33 =>A33
13. What is more complex: A34 or B1 =>A34
14. What is more complex: A32 or B1 =>A32
26. What is more complex: A34 or B3 =>A34
27. What is more complex: A34 or B2 =>A34
28. What is more complex: A1 or B29 =>B29
29. What is more complex: A29 or B29 =>B29
30. What is more complex: A3 or B29 =>A3
31. What is more complex: A3 or B1 =>A3
32. What is more complex: A3 or A32 =>A3
33. What is more complex: A3 or A33 =>A3
34. What is more complex: A3 or B4 =>A3
35. What is more complex: A3 or B3 =>A3
36. What is more complex: A3 or B2 =>A3
37. What is more complex: A3 or A34 =>A34
38. What is more complex: A2 or A34 =>A34
39. What is more complex: A30 or A34 =>A34
51. What is more complex: B32 or B33 =>B33
52. What is more complex: C3 or C4 =>C4
53. What is more complex: C29 or C30 =>C30
54. What is more complex: C3 or C29 =>C29
55. What is more complex: C4 or C29 =>C29
56. What is more complex: C31 or C32 =>C32
57. What is more complex: C33 or C34 =>C34
58. What is more complex: C31 or C33 =>C33
59. What is more complex: C32 or C33 =>C32
60. What is more complex: C32 or C34 =>C32
61. What is more complex: C3 or C31 =>C31
62. What is more complex: C4 or C31 =>C31
63. What is more complex: C29 or C31 =>C29
64. What is more complex: C29 or C33 =>C29
76. What is more complex: B29 or B34 =>B34
77. What is more complex: B1 or B34 =>B34
78. What is more complex: A32 or B34 =>B34
79. What is more complex: A33 or B34 =>B34
80. What is more complex: B4 or B34 =>B34
81. What is more complex: B3 or B34 =>B34
82. What is more complex: B2 or B34 =>B34
83. What is more complex: A3 or B34 =>B34
84. What is more complex: A2 or B34 =>B34
85. What is more complex: A30 or B34 =>B34
86. What is more complex: A31 or B34 =>B34
87. What is more complex: A34 or B34 =>B34
88. What is more complex: A4 or B34 =>B34

15. What is more complex: A32 or A34 =>A34

16. What is more complex: A33 or A34 =>A34

17. What is more complex: B2 or B3 =>B2

18. What is more complex: B4 or B29 =>B4

19. What is more complex: B3 or B29 =>B3

20. What is more complex: B3 or B4 =>B3

21. What is more complex: B1 or B29 =>B1

22. What is more complex: B1 or B4 =>B4

23. What is more complex: A32 or B4 =>B4

24. What is more complex: A33 or B4 =>B4

25. What is more complex: A34 or B4 =>A34

40. What is more complex: A31 or A34 =>A34

41. What is more complex: A4 or A34 =>A4

42. What is more complex: B31 or B32 =>B32

43. What is more complex: B30 or B31 =>B31

44. What is more complex: B33 or B34 =>B33

45. What is more complex: C1 or C2 =>C2

46. What is more complex: B34 or C1 =>C1

47. What is more complex: B33 or C1 =>C1

48. What is more complex: B30 or B34 =>B30

49. What is more complex: B30 or B33 =>B33

50. What is more complex: B31 or B33 =>B33

65. What is more complex: C29 or C34 =>C29

66. What is more complex: C29 or C32 =>C29

67. What is more complex: B34 or C3 =>C3

68. What is more complex: B30 or C3 =>C3

69. What is more complex: B31 or C3 =>C3

70. What is more complex: B32 or C3 =>C3

71. What is more complex: B33 or C3 =>C3

72. What is more complex: C1 or C3 =>C3

73. What is more complex: C2 or C3 =>C3

74. What is more complex: A1 or B34 =>B34

75. What is more complex: A29 or B34 =>B34

Ranking results:

['A1', 'A29', 'B29', 'B1', 'A32', 'A33', **2** 'B4', 'B3', 'B2', 'A3', 'A2', 'A30', 'A31', 'A34', 'A4', **3** 'B34', 'B30', 'B31', 'B32', 'B33', 'C1', **4 NS** 'C2', 'C3', 'C4', 'C31', 'C33', 'C34', 'C32', 'C29', 'C30' **5**]

Linearly interpolated scores:

1. A1=1.166667 (1+1/6)
2. A29=1.333333 (1+2/6)
3. B29=1.5 (1+3/6)
4. B1=1.666667 (1+4/6)
5. A32=1.833333 (1+5/6)
6. A33=2

16. B34=3.166667 (3+1/6)
17. B30=3.333333 (3+2/6)
18. B31=3.5 (3+3/6)
19. B32=3.666667 (3+4/6)
20. B33=3.833333 (3+5/6)
21. C1=4

7.	B4=2.111111 (2+1/9)	22.	C2=4.111111 (4+1/9)
8.	B3=2.222222 (2+2/9)	23.	C3=4.222222 (4+2/9)
9.	B2=2.333333 (2+3/9)	24.	C4=4.333333 (4+3/9)
10.	A3=2.444444 (2+4/9)	25.	C31=4.444444 (4+4/9)
11.	A2=2.555556 (2+5/9)	26.	C33=4.555556 (4+5/9)
12.	A30=2.666667 (2+6/9)	27.	C34=4.666667 (4+6/9)
13.	A31=2.777778 (2+7/9)	28.	C32=4.777778 (4+7/9)
14.	A34=2.888889 (2+8/9)	29.	C29=4.888889 (4+8/9)
15.	A4=3	30.	C30=5

Comment form the candidate:	**Observations from the moderator:**
Candidate would switch C29 with C30 and B34 on top of mark 4 and B30 on the mark 3.	Candidate was asked in the middle of the ranking what she would give to the traffic A34, A3 and she said it would be 3 or 4 score of complexity. Candidate stated that she does not see the solution to the problems in the C29.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V20 or V21 =>v21	6.	What is more complex: V22 or V23 =>V23
2.	What is more complex: V20 or V21 =>V21	7.	What is more complex: V20 or V22 =>V22
3.	What is more complex: V19 or V20 =>V19	8.	What is more complex: V19 or V22 =>V22
4.	What is more complex: V19 or V21 =>V21	9.	What is more complex: V21 or V22 =>V21
5.	What is more complex: V23 or V24 =>V24	10.	What is more complex: V21 or V23 =>V23

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V20', 'V19', **2 NS** 'V22', 'V21', **4** 'V23', 'V24' **5**]	1. V20=1.5 2. V19=2 3. V22=3.5 4. V21=4 5. V23=4.5 6. V24=5

Data gathering from ATCO no. 16

Date and time of the experiments:	**Candidate:**	**Years of experience:**
04.07.2019. / 13:25 h - 15:12 h	Anonymous no. 16	21

Time required to rank the traffic:	**Traffic sample taken and group:**
02h13m (5-min break)	A35-C40 / G6

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A35 =>A4
4. What is more complex: A36 or A37 =>A37
5. What is more complex: A35 or A36 =>A35
6. What is more complex: A35 or A37 =>A37
7. What is more complex: A4 or A37 =>A4
8. What is more complex: A1 or A36 =>A1
9. What is more complex: A1 or A35 =>A35
10. What is more complex: A2 or A35 =>A35
11. What is more complex: A3 or A35 =>A3
12. What is more complex: A3 or A37 =>A3
13. What is more complex: A3 or A4 =>A4
14. What is more complex: A38 or A39 =>A38

26. What is more complex: A39 or B35 =>A39
27. What is more complex: A39 or B4 =>B4
28. What is more complex: A38 or B4 =>B4
29. What is more complex: A40 or B4 =>A40
30. What is more complex: A36 or B3 =>B3
31. What is more complex: A1 or B3 =>B3
32. What is more complex: A2 or B3 =>A2
33. What is more complex: A2 or B2 =>A2
34. What is more complex: A2 or B1 =>B1
35. What is more complex: A35 or B1 =>A35
36. What is more complex: A35 or B35 =>A35
37. What is more complex: A35 or A39 =>A39
38. What is more complex: A37 or A39 =>A37
39. What is more complex: A37 or A38 =>A38

51. What is more complex: B36 or C1 =>B36
52. What is more complex: B36 or B39 =>B36
53. What is more complex: B36 or C2 =>C2
54. What is more complex: B37 or C2 =>C2
55. What is more complex: C3 or C4 =>C3
56. What is more complex: C35 or C36 =>C36
57. What is more complex: C4 or C35 =>C4
58. What is more complex: C4 or C36 =>C4
59. What is more complex: C37 or C38 =>C37
60. What is more complex: C39 or C40 =>C40
61. What is more complex: C38 or C39 =>C39
62. What is more complex: C37 or C39 =>C37
63. What is more complex: C37 or C40 =>C40
64. What is more complex: C35 or C38 =>C38

76. What is more complex: B36 or C39 =>C39
77. What is more complex: B37 or C39 =>C39
78. What is more complex: C2 or C39 =>C39
79. What is more complex: B40 or C39 =>C39
80. What is more complex: A36 or B38 =>B38
81. What is more complex: A1 or B38 =>A1
82. What is more complex: A1 or C35 =>C35
83. What is more complex: B3 or C35 =>C35
84. What is more complex: B2 or C35 =>C35
85. What is more complex: A2 or C35 =>C35
86. What is more complex: B1 or C35 =>B1
87. What is more complex: B1 or C1 =>C1
88. What is more complex: B35 or C1 =>C1
89. What is more complex: A35 or C1 =>C1

15. What is more complex: A40 or B1 =>A40

16. What is more complex: A39 or B1 =>A39

17. What is more complex: A39 or A40 =>A40

18. What is more complex: A38 or A40 =>A40

19. What is more complex: B2 or B3 =>B2

20. What is more complex: B4 or B35 =>B4

21. What is more complex: B3 or B35 =>B35

22. What is more complex: B2 or B35 =>B35

23. What is more complex: B1 or B3 =>B1

24. What is more complex: B1 or B2 =>B1

25. What is more complex: B1 or B35 =>B35

40. What is more complex: A3 or A38 =>A38

41. What is more complex: A4 or A38 =>A38

42. What is more complex: B37 or B38 =>B37

43. What is more complex: B36 or B38 =>B36

44. What is more complex: B36 or B37 =>B37

45. What is more complex: B39 or B40 =>B40

46. What is more complex: C1 or C2 =>C2

47. What is more complex: B39 or C1 =>B39

48. What is more complex: B39 or C2 =>C2

49. What is more complex: B40 or C2 =>B40

50. What is more complex: B38 or C1 =>C1

65. What is more complex: C36 or C38 =>C36

66. What is more complex: C36 or C39 =>C36

67. What is more complex: C36 or C37 =>C37

68. What is more complex: C4 or C37 =>C4

69. What is more complex: C4 or C40 =>C40

70. What is more complex: C3 or C40 =>C3

71. What is more complex: B38 or C35 =>C35

72. What is more complex: C1 or C35 =>C1

73. What is more complex: C1 or C38 =>C38

74. What is more complex: B39 or C38 =>C38

75. What is more complex: B36 or C38 =>B36

90. What is more complex: A39 or C1 =>C1

91. What is more complex: A37 or C1 =>C1

92. What is more complex: A3 or C1 =>C1

93. What is more complex: A4 or C1 =>A4

94. What is more complex: A4 or B39 =>A4

95. What is more complex: A4 or C38 =>C38

96. What is more complex: A38 or C38 =>C38

97. What is more complex: B4 or C38 =>C38

98. What is more complex: A40 or C38 =>C38

Ranking results:

['A36', 'B38', 'A1', 'B3', 'B2', 'A2', **1** 'C35', 'B1', 'B35', 'A35', 'A39', 'A37', 'A3', **2** 'C1', 'B39', 'A4', 'A38', 'B4', 'A40', 'C38', 'B36', 'B37', **3 NS** 'C2', 'B40', **4** 'C39', 'C36', 'C37', 'C4', 'C40', 'C3' **5**]

Linearly interpolated scores:

1. A36=0.166667 (0+1/6)
2. B38=0.333333 (0+2/6)
3. A1=0.5 (0+3/6)
4. B3=0.666667 (0+4/6)
5. B2=0.833333 (0+5/6)
6. A2=1

16. A4=2.333333 (2+3/9)
17. A38=2.444444 (2+4/9)
18. B4=2.555556 (2+5/9)
19. A40=2.666667 (2+6/9)
20. C38=2.777778 (2+7/9)
21. B36=2.888889 (2+8/9)

7.	C35=1.142857	(1+1/7)	22.	B37=3	
8.	B1=1.285714	(1+2/7)	23.	C2=3.5	
9.	B35=1.428571	(1+3/7)	24.	B40=4	
10.	A35=1.571429	(1+4/7)	25.	C39=4.166667	(4+1/6)
11.	A39=1.714286	(1+5/7)	26.	C36=4.333333	(4+2/6)
12.	A37=1.857143	(1+6/7)	27.	C37=4.5	(4+3/6)
13.	A3=2		28.	C4=4.666667	(4+4/6)
14.	C1=2.111111	(2+1/9)	29.	C40=4.833333	(4+5/6)
15.	B39=2.222222	(2+2/9)	30.	C3=5	

Comment form the candidate:	**Observations from the moderator:**
Candidate would switch A35 in the category of 1.	Candidate would give for the traffic situation B40 mark between 3 and 4 (asked in the middle of the ranking).

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V26 or V27 =>V27	5.	What is more complex: V25 or V28 =>V28
2.	What is more complex: V25 or V26 =>V26	6.	What is more complex: V26 or V28 =>V28
3.	What is more complex: V29 or V30 =>V30	7.	What is more complex: V27 or V28 =>V27
4.	What is more complex: V28 or V29 =>V29	8.	What is more complex: V27 or V29 =>V29

Validation Ranking results:	**Validation Linearly interpolated scores:**
['V25', **1** 'V26', **2** 'V28', **3 NS** 'V27', **4** 'V29', 'V30' **5**]	1. V25=1 2. V26=2 3. V28=3 4. V27=4 5. V29=4.5 6. V30=5

Data gathering from ATCO no. 17

Date and time of the experiments:	Candidate:	Years of experience:
12.07.2019. / 16:05 h - 19:00 h	Anonymous no. 17	24

Time required to rank the traffic:	Traffic sample taken and group:
02h55m (5-min break)	A35-C40 / G6

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A3
2. What is more complex: A1 or A2 =>A2
3. What is more complex: A4 or A35 =>A4
4. What is more complex: A36 or A37 =>A37
5. What is more complex: A35 or A36 =>A35
6. What is more complex: A35 or A37 =>A35
7. What is more complex: A1 or A36 =>A36
8. What is more complex: A2 or A36 =>A36
9. What is more complex: A3 or A36 =>A36
10. What is more complex: A38 or A39 =>A38
11. What is more complex: A40 or B1 =>A40
12. What is more complex: A39 or B1 =>A39
13. What is more complex: A39 or A40 =>A40
14. What is more complex: A38 or A40 =>A40
26. What is more complex: A1 or B1 =>B1
27. What is more complex: A2 or B1 =>A2
28. What is more complex: A2 or B2 =>A2
29. What is more complex: A2 or A39 =>A39
30. What is more complex: A3 or A39 =>A39
31. What is more complex: A36 or A39 =>A39
32. What is more complex: A37 or A39 =>A39
33. What is more complex: A35 or A39 =>A39
34. What is more complex: A4 or A39 =>A4
35. What is more complex: A4 or B35 =>A4
36. What is more complex: A4 or B3 =>A4
37. What is more complex: A4 or B4 =>A4
38. What is more complex: A4 or A38 =>A4
39. What is more complex: A4 or A40 =>A40
51. What is more complex: C3 or C4 =>C4
52. What is more complex: C35 or C36 =>C36
53. What is more complex: C3 or C35 =>C3
54. What is more complex: C3 or C36 =>C3
55. What is more complex: C37 or C38 =>C37
56. What is more complex: C39 or C40 =>C40
57. What is more complex: C38 or C39 =>C39
58. What is more complex: C37 or C39 =>C37
59. What is more complex: C37 or C40 =>C40
60. What is more complex: C35 or C38 =>C38
61. What is more complex: C36 or C38 =>C38
62. What is more complex: C3 or C38 =>C3
63. What is more complex: C3 or C39 =>C3
64. What is more complex: C3 or C37 =>C3
76. What is more complex: A1 or C35 =>C35
77. What is more complex: B1 or C35 =>C35
78. What is more complex: B2 or C35 =>C35
79. What is more complex: A2 or C35 =>C35
80. What is more complex: A3 or C35 =>C35
81. What is more complex: A36 or C35 =>C35
82. What is more complex: A37 or C35 =>C35
83. What is more complex: A35 or C35 =>C35
84. What is more complex: A39 or C35 =>C35
85. What is more complex: B35 or C35 =>C35
86. What is more complex: B3 or C35 =>C35
87. What is more complex: B4 or C35 =>C35
88. What is more complex: A38 or C35 =>C35
89. What is more complex: A4 or C35 =>A4

15. What is more complex: B2 or B3 =>B3

16. What is more complex: B4 or B35 =>B4

17. What is more complex: B2 or B35 =>B35

18. What is more complex: B3 or B35 =>B3

19. What is more complex: B3 or B4 =>B4

20. What is more complex: B1 or B2 =>B2

21. What is more complex: A39 or B2 =>A39

22. What is more complex: A39 or B35 =>B35

23. What is more complex: A38 or B35 =>A38

24. What is more complex: A38 or B3 =>A38

25. What is more complex: A38 or B4 =>A38

40. What is more complex: B37 or B38 =>B38

41. What is more complex: B36 or B37 =>B36

42. What is more complex: B36 or B38 =>B38

43. What is more complex: B39 or B40 =>B40

44. What is more complex: C1 or C2 =>C2

45. What is more complex: B39 or C1 =>B39

46. What is more complex: B39 or C2 =>B39

47. What is more complex: B37 or C1 =>C1

48. What is more complex: B36 or C1 =>B36

49. What is more complex: B36 or C2 =>C2

50. What is more complex: B38 or C2 =>C2

65. What is more complex: C3 or C40 =>C40

66. What is more complex: C4 or C40 =>C40

67. What is more complex: B37 or C35 =>B37

68. What is more complex: B37 or C36 =>C36

69. What is more complex: C1 or C36 =>C36

70. What is more complex: B36 or C36 =>C36

71. What is more complex: B38 or C36 =>C36

72. What is more complex: C2 or C36 =>C36

73. What is more complex: B39 or C36 =>C36

74. What is more complex: B40 or C36 =>B40

75. What is more complex: B40 or C38 =>C38

90. What is more complex: A4 or B37 =>A4

91. What is more complex: A4 or C1 =>A4

92. What is more complex: A4 or B36 =>A4

93. What is more complex: A4 or B38 =>A4

94. What is more complex: A4 or C2 =>C2

95. What is more complex: A40 or C2 =>C2

Ranking results:

['A1', 'B1', 'B2', 'A2', **1** 'A3', 'A36', 'A37', 'A35', **2** 'A39', 'B35', 'B3', 'B4', 'A38', 'C35', 'B37', 'C1', 'B36', 'B38', **3** 'A4', 'A40', 'C2', 'B39', 'C36', 'B40', **NS 4** 'C38', 'C39', 'C37', 'C3', 'C4', 'C40' **5**]

Linearly interpolated scores:

1. A1=0.25
2. B1=0.5
3. B2=0.75
4. A2=1
5. A3=1.25
6. A36=1.5

16. C1=2.8
17. B36=2.9
18. B38=3
19. A4=3.166667 (3+1/6)
20. A40=3.333333 (3+2/6)
21. C2=3.5 (3+3/6)

7.	A37=1.75	22.	B39=3.666667 (3+4/6)
8.	A35=2	23.	C36=3.833333 (3+5/6)
9.	A39=2.1	24.	B40=4
10.	B35=2.2	25.	C38=4.166667 (4+1/6)
11.	B3=2.3	26.	C39=4.333333 (4+2/6)
12.	B4=2.4	27.	C37=4.5 (4+3/6)
13.	A38=2.5	28.	C3=4.666667 (4+4/6)
14.	C35=2.6	29.	C4=4.833333 (4+5/6)
15.	B37=2.7	30.	C40=5

Comment form the candidate:	**Observations from the moderator:**
No comments from the candidate.	Candidate is using the ruler. When asked in the middle of the ranking to score the traffic A40 candidate sad he would give the score of 4.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V26 or V27 =>V26	5.	What is more complex: V25 or V28 =>V28
2.	What is more complex: V25 or V27 =>V27	6.	What is more complex: V27 or V28 =>V27
3.	What is more complex: V29 or V30 =>V30	7.	What is more complex: V27 or V29 =>V29
4.	What is more complex: V28 or V29 =>V29	8.	What is more complex: V26 or V29 =>V29

Validation Ranking results:

['V25', **1** 'V28', **2** 'V27', **3 NS** 'V26', 'V29', **4** 'V30' **5**]

Validation Linearly interpolated scores:

1.	V25=1	4.	V26=3.5
2.	V28=2	5.	V29=4
3.	V27=3	6.	V30=5

Data gathering from ATCO no. 18

Date and time of the experiments:	**Candidate:**	**Years of experience:**
15.07.2019. / 10:24 h - 11:29 h	Anonymous no. 18	12

Time required to rank the traffic:	**Traffic sample taken and group:**
01h05m (5-min break)	A35-C40 / G6

Merge sort candidate's answers:

1. What is more complex: A2 or A3 =>A2
2. What is more complex: A1 or A3 =>A1
3. What is more complex: A1 or A2 =>A1
4. What is more complex: A4 or A35 =>A4
5. What is more complex: A36 or A37 =>A37
6. What is more complex: A35 or A36 =>A36
7. What is more complex: A4 or A36 =>A4
8. What is more complex: A4 or A37 =>A4
9. What is more complex: A3 or A35 =>A3
10. What is more complex: A3 or A36 =>A3
11. What is more complex: A3 or A37 =>A3
12. What is more complex: A3 or A4 =>A4
13. What is more complex: A2 or A4 =>A4
14. What is more complex: A1 or A4 =>A4

26. What is more complex: B1 or B2 =>B1
27. What is more complex: B1 or B3 =>B3
28. What is more complex: A39 or B3 =>A39
29. What is more complex: A39 or B4 =>B4
30. What is more complex: A40 or B4 =>B4
31. What is more complex: A38 or B4 =>A38
32. What is more complex: A35 or B35 =>B35
33. What is more complex: A36 or B35 =>B35
34. What is more complex: A37 or B35 =>B35
35. What is more complex: A3 or B35 =>B35
36. What is more complex: A2 or B35 =>B35
37. What is more complex: A1 or B35 =>B35
38. What is more complex: A4 or B35 =>B35
39. What is more complex: B37 or B38 =>B38

51. What is more complex: C35 or C36 =>C36
52. What is more complex: C4 or C35 =>C4
53. What is more complex: C4 or C36 =>C4
54. What is more complex: C37 or C38 =>C37
55. What is more complex: C39 or C40 =>C40
56. What is more complex: C38 or C39 =>C38
57. What is more complex: C38 or C40 =>C40
58. What is more complex: C37 or C40 =>C40
59. What is more complex: C35 or C39 =>C39
60. What is more complex: C36 or C39 =>C39
61. What is more complex: C4 or C39 =>C4
62. What is more complex: C4 or C38 =>C38
63. What is more complex: C3 or C38 =>C3
64. What is more complex: C3 or C37 =>C37

76. What is more complex: C2 or C3 =>C3
77. What is more complex: A35 or C35 =>C35
78. What is more complex: A36 or C35 =>C35
79. What is more complex: A37 or C35 =>C35
80. What is more complex: A3 or C35 =>C35
81. What is more complex: A2 or C35 =>C35
82. What is more complex: A1 or C35 =>C35
83. What is more complex: A4 or C35 =>C35
84. What is more complex: B35 or C35 =>C35
85. What is more complex: B2 or C35 =>C35
86. What is more complex: B1 or C35 =>C35
87. What is more complex: B3 or C35 =>C35
88. What is more complex: A39 or C35 =>C35
89. What is more complex: A40 or C35 =>C35

15. What is more complex: A38 or A39 =>A38

16. What is more complex: A40 or B1 =>A40

17. What is more complex: A39 or B1 =>A39

18. What is more complex: A39 or A40 =>A40

19. What is more complex: A38 or A40 =>A38

20. What is more complex: B2 or B3 =>B3

21. What is more complex: B4 or B35 =>B4

22. What is more complex: B2 or B35 =>B2

23. What is more complex: B2 or B4 =>B4

24. What is more complex: B3 or B4 =>B4

25. What is more complex: B1 or B35 =>B1

40. What is more complex: B36 or B37 =>B37

41. What is more complex: B39 or B40 =>B40

42. What is more complex: C1 or C2 =>C2

43. What is more complex: B39 or C1 =>C1

44. What is more complex: B40 or C1 =>C1

45. What is more complex: B36 or B39 =>B36

46. What is more complex: B36 or B40 =>B36

47. What is more complex: B36 or C1 =>C1

48. What is more complex: B37 or C1 =>C1

49. What is more complex: B38 or C1 =>C1

50. What is more complex: C3 or C4 =>C3

65. What is more complex: B39 or C35 =>B39

66. What is more complex: B39 or C36 =>C36

67. What is more complex: B40 or C36 =>B40

68. What is more complex: B40 or C39 =>C39

69. What is more complex: B36 or C39 =>C39

70. What is more complex: B37 or C39 =>C39

71. What is more complex: B38 or C39 =>C39

72. What is more complex: C1 or C39 =>C39

73. What is more complex: C2 or C39 =>C2

74. What is more complex: C2 or C4 =>C2

75. What is more complex: C2 or C38 =>C2

90. What is more complex: B4 or C35 =>B4

91. What is more complex: B4 or B39 =>B4

92. What is more complex: B4 or C36 =>C36

93. What is more complex: A38 or C36 =>C36

Ranking results:

['A35', 'A36', 'A37', 'A3', 'A2', 'A1', **1** 'A4', 'B35', 'B2', 'B1', **2** 'B3', 'A39', **NS** 'A40', 'C35', **3** 'B39', 'B4', 'A38', 'C36', 'B40', 'B36', 'B37', **4** 'B38', 'C1', 'C39', 'C4', 'C38', 'C2', 'C3', 'C37', 'C40' **5**]

Linearly interpolated scores:

1. A35=0.166667 (0+1/6)
2. A36=0.333333 (0+2/6)
3. A37=0.5 (0+3/6)
4. A3=0.666667 (0+4/6)
5. A2=0.833333 (0+5/6)
6. A1=1

16. B4=3.285714 (3+2/7)
17. A38=3.428571 (3+3/7)
18. C36=3.571429 (3+4/7)
19. B40=3.714286 (3+5/7)
20. B36=3.857143 (3+6/7)
21. B37=4

7.	A4=1.25	(1+1/4)	22.	B38=4.111111	(4+1/9)
8.	B35=1.5	(1+2/4)	23.	C1=4.222222	(4+2/9)
9.	B2=1.75	(1+3/4)	24.	C39=4.333333	(4+3/9)
10.	B1=2		25.	C4=4.444444	(4+4/9)
11.	B3=2.25	(2+1/4)	26.	C38=4.555556	(4+5/9)
12.	A39=2.5	(2+2/4)	27.	C2=4.666667	(4+6/9)
13.	A40=2.75	(2+3/4)	28.	C3=4.777778	(4+7/9)
14.	C35=3		29.	C37=4.888889	(4+8/9)
15.	B39=3.142857	(3+1/7)	30.	C40=5	

Comment form the candidate:	**Observations from the moderator:**
Candidate stated that C39 is similar to a pattern of avoiding the weather.	When asked in the middle of the ranking B40 and B39 would give the score of 5.

Validation airspace Merge sort candidate's answers:

1.	What is more complex: V26 or V27 =>V27	5.	What is more complex: V25 or V28 =>V28
2.	What is more complex: V25 or V26 =>V26	6.	What is more complex: V26 or V28 =>V26
3.	What is more complex: V29 or V30 =>V30	7.	What is more complex: V26 or V29 =>V29
4.	What is more complex: V28 or V29 =>V29	8.	What is more complex: V27 or V29 =>V29

Validation Ranking results:

['V25', **1** 'V28', 'V26', **3 NS** 'V27', 'V29', **4** 'V30' **5**]

Validation Linearly interpolated scores:

1.	V25=1	4.	V27=3.5
2.	V28=2.5	5.	V29=4
3.	V26=3	6.	V30=5

Appendix 7 – Python code of model development

```
# Basic data analysis modules
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.special import expit
# Machine learning models and evaluation
from sklearn.linear_model import LogisticRegression, LinearRegression, BayesianRidge
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.pipeline import Pipeline
from sklearn.metrics import brier_score_loss
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import SpectralClustering
# Theano is needed for defining pymc3 models
import theano
theano.config.warn.round=False

import theano.tensor as t

def tinvlogit(x):
    return t.exp(x) / (1 + t.exp(x))
# Turns off SettingWithCopyWarning warning, which I believe is safe and suppreses the output
pd.options.mode.chained_assignment = None  # default='warn'

pd.read_csv('data/features_count_v2.csv').head(10)

pd.read_csv('data/features_count_v2.csv')\
              .pivot('situation','code','count')\
              .fillna(0.0).head()
# TODO: features_count_v4.csv holds both training situations (A1-A30,B1-B30,C1-C30) and validation situations (V1-V28)
pd.read_csv('data/features_count_v4.csv')\
              .pivot('situation','code','count')\
              .fillna(0.0).loc[['V'+str(x) for x in range(1,29)]].head()
pd.read_excel(r'data/aircraft_count_recoded.xlsx', sheet_name='Sheet1', index_col=0)\
  .rename(columns={'Aircraft in airspace':'count_airspace','Aircraft approaching':'count_approaching'})\
  .rename_axis('situation').head()

pd.read_csv('data/grades.csv').head()

pd.read_csv('data/grades_interpolation.csv').head()

pd.read_csv('data/comparisons.csv').head()

pd.read_csv('data/rankings.csv').head()

pd.read_csv('data/new_sector.csv').head()

pd.read_csv('data/new_sector_interpolated.csv').head()

temp = pd.read_csv('data/features_aircrafts.csv')
temp = pd.pivot_table(temp, index=['situation','aircraft'],columns='variable',values='value',fill_value=0.0)
temp.head(12)
temp = pd.read_csv('data/features_aircrafts_CP.csv')
temp = pd.pivot_table(temp, index=['situation','aircraft'],columns='variable',values='value',fill_value=0.0)
temp.head()
# Features v1 (logistic regression) with the original set of 69 task types and numerical features (all features)
features1 = pd.read_csv('data/features_count_v2.csv')\
              .pivot('situation','code','count')\
              .fillna(0.0).astype(np.float64)
aircraft_count = pd.read_excel(r'data/aircraft_count_recoded.xlsx', sheet_name='Sheet1', index_col=0)
aircraft_count = aircraft_count.rename(columns={'Aircraft in airspace':'count_airspace',
                                                'Aircraft approaching':'count_approaching'})
aircraft_count = aircraft_count.rename_axis('situation')
# Features v2 (logistic regression) all features with aircraft counts

# Using left join because aircraft_count has counts for validation situations as well!
features2 = features1.join(aircraft_count,how='left')

# Add squares of aircraft count!
# features2['count_airspace_square'] = np.square(features2['count_airspace'])
# features2['count_approaching_square'] = np.square(features2['count_approaching'])

# True number of aircraft pairs
features2['count_airspace_square'] = ((features2['count_airspace']*(features2['count_airspace']-1))/2).astype(np.int32)
features2['count_approaching_square'] = ((features2['count_approaching']*(features2['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features2['count_airspace']+features2['count_approaching']
features2['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
```

```python
# Features v3 (logistic regression) with a square root of all features and aircraft counts
features3 = features1.transform(np.sqrt).join(aircraft_count,how='left')

# Add squares of aircraft count!
# features3['count_airspace_square'] = np.square(features3['count_airspace'])
# features3['count_approaching_square'] = np.square(features3['count_approaching'])

# True number of aircraft pairs
features3['count_airspace_square'] = ((features3['count_airspace']*(features3['count_airspace']-1))/2).astype(np.int32)
features3['count_approaching_square'] = ((features3['count_approaching']*(features3['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features3['count_airspace']+features3['count_approaching']
features3['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
grades = pd.read_csv('data/grades.csv')#,index_col=1)
grades_validation = pd.read_csv('data/grades_validation_recoded.csv')#,index_col=1)
grades_interpolation = pd.read_csv('data/grades_interpolation.csv')#,index_col=1)
grades_interpolation_validation = pd.read_csv('data/grades_interpolation_validation_recoded.csv')#,index_col=1)
# Features v4 (linear regression) all features

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1,how='left')
features4 = temp[temp.columns.difference(['grade'])] # exclude target variable
# Features v5 (linear regression) all features with aircraft counts

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1,how='left')
features5 = temp[temp.columns.difference(['grade'])] # exclude target variable

features5 = features5.join(aircraft_count,how='left')

# Add squares of aircraft count!
# features5['count_airspace_square'] = np.square(features5['count_airspace'])
# features5['count_approaching_square'] = np.square(features5['count_approaching'])

# True number of aircraft pairs
features5['count_airspace_square'] = ((features5['count_airspace']*(features5['count_airspace']-1))/2).astype(np.int32)
features5['count_approaching_square'] = ((features5['count_approaching']*(features5['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features5['count_airspace']+features5['count_approaching']
features5['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
# Features v6 (linear regression) sqrt of all features with aircraft counts

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1,how='left')
features6 = temp[temp.columns.difference(['grade'])] # exclude target variable

features6 = features6.transform(np.sqrt).join(aircraft_count,how='left') # with sqrt of all other features

# Add squares of aircraft count!
# features6['count_airspace_square'] = np.square(features6['count_airspace'])
# features6['count_approaching_square'] = np.square(features6['count_approaching'])

# True number of aircraft pairs
features6['count_airspace_square'] = ((features6['count_airspace']*(features6['count_airspace']-1))/2).astype(np.int32)
features6['count_approaching_square'] = ((features6['count_approaching']*(features6['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features6['count_airspace']+features6['count_approaching']
features6['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
# Features v7 (linear regression) only task types with aircraft counts

task_types = ['CCC','CCO','CCS','CPC','CPO','CPS','CC','CO','CS','ER','FT','IC','PC','PO','PS','SI','SN','SP']

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1[task_types],how='left')
features7 = temp[temp.columns.difference(['grade'])] # exclude target variable

features7 = features7.join(aircraft_count.astype(float),how='left')

# Add squares of aircraft count!
# features7['count_airspace_square'] = np.square(features7['count_airspace'])
# features7['count_approaching_square'] = np.square(features7['count_approaching'])

# True number of aircraft pairs
features7['count_airspace_square'] = ((features7['count_airspace']*(features7['count_airspace']-1))/2).astype(np.int32)
features7['count_approaching_square'] = ((features7['count_approaching']*(features7['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features7['count_airspace']+features7['count_approaching']
features7['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
```

```
# Features v8 (linear regression) all numerical features without task types and with aircraft counts

task_types = ['CCC','CCO','CCS','CPC','CPO','CPS','CC','CO','CS','ER','FT','IC','PC','PO','PS','SI','SN','SP']

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1[features1.columns.difference(task_types)],how='left')
features8 = temp[temp.columns.difference(['grade'])] # exclude target variable

features8 = features8.join(aircraft_count.astype(float),how='left')

# True number of aircraft pairs
features8['count_airspace_square'] = ((features8['count_airspace']*(features8['count_airspace']-1))/2).astype(np.int32)
features8['count_approaching_square'] = ((features8['count_approaching']*(features8['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features8['count_airspace']+features8['count_approaching']
features8['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
# Trying various combinations of features

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1[task_types],how='left')
features_temp = temp[temp.columns.difference(['grade'])] # exclude target variable
features_temp = features_temp.join(aircraft_count.astype(float),how='left')

# Squares of aircraft counts
# features_temp['count_airspace_square'] = np.square(features_temp['count_airspace'])
# features_temp['count_approaching_square'] = np.square(features_temp['count_approaching'])

# True number of aircraft pairs
features_temp['count_airspace_square'] = ((features_temp['count_airspace']*(features_temp['count_airspace']-1))
                                           /2).astype(np.int32)
features_temp['count_approaching_square'] = ((features_temp['count_approaching']*(features_temp['count_approaching']-1))
                                              /2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features_temp['count_airspace']+features_temp['count_approaching']
features_temp['count_square'] = ((temp*(temp-1))/2).astype(np.int32)
features9 = features_temp[['count_airspace','count_approaching','count_airspace_square',
                           'count_approaching_square','count_square']]

features10 = features_temp[['count_airspace_square']]

features11 = features_temp[['count_airspace_square','SN','CO','SP','count_airspace']]

features12 = features_temp[['count_airspace_square','CO','count_airspace','PS','PC']]

features13 = features_temp[['CO','CPS','CPC','CS','CC']]
# New combination of features

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1,how='left')
features14 = temp[temp.columns.difference(['grade'])] # exclude target variable
features14 = features14[['CCC','CCO','CCS','CPC','CPO','CPS','CC','CO','CS','PC','PO','PS','free-left-1st-1',
                         'free-left-1st-2','free-right-1st-1','free-right-1st-2','free-left-2nd-1','free-left-2nd-2',
                         'free-right-2nd-1','free-right-2nd-2','free-above-1st-1','free-above-1st-2','free-below-1st-1',
                         'free-below-1st-2','free-above-2nd-1','free-above-2nd-2','free-below-2nd-1','free-below-2nd-2']]
# New combination of features v15

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features1,how='left')
features15 = temp[temp.columns.difference(['grade'])] # exclude target variable
features15 = features15[['CCC','CCO','CCS','CPC','CPO','CPS','CC','CO','CS','PC','PO','PS','turb-faster',
                         'turb-same','turb-slower','conv-0-20','conv-21-44','conv-45-90','conv-91-135','conv-136-159',
                         'conv-160-180','conflict-1st-0-10','conflict-1st-11-20','conflict-1st-21-30','conflict-1st-31-50',
                         'conflict-1st-51-80','conflict-1st-81','conflict-2nd-0-10','conflict-2nd-11-20','conflict-2nd-21-30',
                         'conflict-2nd-31-50','conflict-2nd-51-80','conflict-2nd-81']]
# New combination of features v16 - like v14 but with aircraft counts

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features2,how='left')
features16 = temp[temp.columns.difference(['grade'])] # exclude target variable
features16 = features16[['CCC','CCO','CCS','CPC','CPO','CPS','CC','CO','CS','PC','PO','PS','free-left-1st-1',
                         'free-left-1st-2','free-right-1st-1','free-right-1st-2','free-left-2nd-1','free-left-2nd-2',
                         'free-right-2nd-1','free-right-2nd-2','free-above-1st-1','free-above-1st-2','free-below-1st-1',
                         'free-below-1st-2','free-above-2nd-1','free-above-2nd-2','free-below-2nd-1','free-below-2nd-2',
                         'count_airspace','count_approaching','count_airspace_square','count_approaching_square',
                         'count_square']]
```

```
# New combination of features v17 - like v15 but with aircraft counts

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features2,how='left')
features17 = temp[temp.columns.difference(['grade'])] # exclude target variable
features17 = features17[['CCC','CCO','CCS','CPC','CPO','CPS','CC','CO','CS','PC','PO','PS','turb-faster',
                         'turb-same','turb-slower','conv-0-20','conv-21-44','conv-45-90','conv-91-135','conv-136-159',
                         'conv-160-180','conflict-1st-0-10','conflict-1st-11-20','conflict-1st-21-30','conflict-1st-31-50',
                         'conflict-1st-51-80','conflict-1st-81','conflict-2nd-0-10','conflict-2nd-11-20','conflict-2nd-21-30',
                         'conflict-2nd-31-50','conflict-2nd-51-80','conflict-2nd-81','count_airspace','count_approaching',
                         'count_airspace_square','count_approaching_square','count_square']]
grade_value = temp['grade'].values # include only target variable
# grade_value[:5]
# Comparisons are used in pairwise modeling for which we used logistic regression
comparisons = pd.read_csv('data/comparisons.csv')
# comparisons.head()
rankings = pd.read_csv('data/rankings.csv')

# We only select rankings of situations A1-A4, B1-B4, C1-C4 which are shared accross all controllers
situations_shared = [letter+str(i) for letter in ['A','B','C'] for i in range(1,5)]
rankings_shared = rankings[rankings['situation'].isin(situations_shared)]

# We recalculate rank of each situation within each controller
rankings_shared['rank_within'] = rankings_shared.groupby('controller_id')['rank'].rank()
# rankings_shared.head(20)
grade_duplicates = grades[['situation','grade']].groupby(['situation','grade'],as_index=False).size().reset_index()
grade_duplicates = grade_duplicates.sort_values(by='situation')
grade_duplicates = grade_duplicates.rename(columns={0:'count'})
# grade_duplicates.head()
grade_duplicates_validation = grades_validation[['situation','grade']].groupby(['situation','grade'],
                                                       as_index=False).size().reset_index()
grade_duplicates_validation = grade_duplicates_validation.sort_values(by='situation')
grade_duplicates_validation = grade_duplicates_validation.rename(columns={0:'count'})
# grade_duplicates_validation.head()
grade_means = grades[['situation','grade']].groupby(['situation'],as_index=False).mean()
grade_means = grade_means.rename(columns={'grade':'grade_mean'})
# grade_means.head()
grade_means_validation = grades_validation[['situation','grade']].groupby(['situation'],as_index=False).mean()
grade_means_validation = grade_means_validation.rename(columns={'grade':'grade_mean'})
# grade_means.head()
grade_means_interpolation = grades_interpolation[['situation','grade']].groupby(['situation'],as_index=False).mean()
grade_means_interpolation = grade_means_interpolation.rename(columns={'grade':'grade_mean_interpolation'})
# grade_means_interpolation.head()
grade_means_interpolation_validation = grades_interpolation_validation[['situation','grade']].groupby(['situation'],
                                                                                    as_index=False).mean()
grade_means_interpolation_validation = grade_means_interpolation_validation.rename(columns={'grade':'grade_mean_interpolation'})
# grade_means_interpolation_validation.head()
# NOTE: The sorting of the graph in the exploratory analysis section is determined by sorting variable here!
grade_duplicates_means = pd.merge(grade_duplicates,grade_means,on='situation',how='inner')
grade_duplicates_means = pd.merge(grade_duplicates_means,grade_means_interpolation,on='situation',how='inner')\
                              .sort_values(by='grade_mean_interpolation')
# grade_duplicates_means.head()
# NOTE: The sorting of the graph in the exploratory analysis section is determined by sorting variable here!
grade_duplicates_means_validation = pd.merge(grade_duplicates_validation,grade_means_validation,on='situation',how='inner')
grade_duplicates_means_validation = pd.merge(grade_duplicates_means_validation,grade_means_interpolation_validation,
                                                 on='situation',how='inner')\
                              .sort_values(by='grade_mean_interpolation')
# grade_duplicates_means_validation.head()
grade_means_sorted = grade_duplicates_means[['situation','grade_mean','grade_mean_interpolation']].drop_duplicates(keep='first')
# grade_means_sorted.head()
grade_means_sorted_validation = grade_duplicates_means_validation[['situation','grade_mean',
                                                                    'grade_mean_interpolation']].drop_duplicates(keep='first')
# grade_means_sorted_validation.head()
# Validation situations

# TODO: features_count_v4.csv holds both training situations (A1-A30,B1-B30,C1-C30) and validation situations (V1-V28)
features_validation = pd.read_csv('data/features_count_v4.csv')\
              .pivot('situation','code','count')\
              .fillna(0.0).loc[['V'+str(x) for x in range(1,29)]].astype(np.float64)

# TODO: These are already loaded in section above. Choose where to load them!
grades_validation = pd.read_csv('data/grades_validation_recoded.csv')#,index_col=1)
grades_interpolation_validation = pd.read_csv('data/grades_interpolation_validation_recoded.csv')#,index_col=1)

# Joining mean grades of each situation with the features describing the situation (tasks)
temp = grades_interpolation_validation[['situation','grade']].groupby(['situation']).mean()
temp = temp.join(features_validation,how='left')
features_validation = temp[temp.columns.difference(['grade'])] # exclude target variable

# Features6 is the best performing feature subset so here we take sqrt of all features as well
features_validation = features_validation.transform(np.sqrt).join(aircraft_count,how='left')
# features_validation = features_validation.join(aircraft_count.astype(float),how='left')

# Add squares of aircraft count!
# features_validation['count_airspace_square'] = np.square(features_validation['count_airspace'])
# features_validation['count_approaching_square'] = np.square(features_validation['count_approaching'])
```

```python
# True number of aircraft pairs
features_validation['count_airspace_square'] = ((features_validation['count_airspace']*
                                                 (features_validation['count_airspace']-1))/2).astype(np.int32)
features_validation['count_approaching_square'] = ((features_validation['count_approaching']*
                                                    (features_validation['count_approaching']-1))/2).astype(np.int32)

# Total number of aircraft pairs, both inside and outside of airspace
temp = features_validation['count_airspace']+features_validation['count_approaching']
features_validation['count_square'] = ((temp*(temp-1))/2).astype(np.int32)

# Features11 is the best performing feature subset and so we will use it on validation features as well
features11_validation = features_validation[['count_airspace_square','SN','CO','SP','count_airspace']]
# Comparison of complexity grades given by different controllers to same traffic situations
fig = plt.figure(figsize=(10,25))

for label_sorted,data,i in zip(['sorted alphabetically','sorted by mean grade'],
                               [grade_duplicates,grade_duplicates_means],
                               [0,1]):

    ax = plt.subplot2grid((1,2), (0,i))

    ax.scatter(data['grade'],data['situation'],cmap=plt.cm.Set1,marker='o',label='individual grades')
    for i,temp in data.iterrows():
        ax.annotate(temp['count'],(temp['grade'],temp['situation']),
                    textcoords='offset points',
                    xytext=(5,-3))
    ax.scatter(grade_means_sorted['grade_mean'],
               grade_means_sorted['situation'],
               color='red',marker='^',label='mean grade')
    ax.scatter(grade_means_sorted['grade_mean_interpolation'],
               grade_means_sorted['situation'],
               color='green',marker='s',label='mean interpolated grade')
    ax.set(title='Complexity grades given to traffic situations\nby controllers '+
                 'and labeled with\nmultiplicity ('+label_sorted+')',
           xlabel='controllers complexity grades',
           ylabel='traffic situations')
    ax.xaxis.set_ticks(range(1,6))
    ax.legend(loc='upper left')

plt.tight_layout();
controllers_grade_validation = grades_interpolation_validation.set_index('situation').join
(grade_means_interpolation_validation.set_index('situation'),how='left')
controllers_grade_validation['diff'] = controllers_grade_validation['grade']
- controllers_grade_validation['grade_mean_interpolation']
# Comparison of complexity grades given by different controllers to same traffic situations
fig = plt.figure(figsize=(10,7))

for label_sorted,data,i in zip(['sorted alphabetically','sorted by mean grade'],
                               [grade_duplicates_validation,grade_duplicates_means_validation],
                               [0,1]):

    ax = plt.subplot2grid((1,2), (0,i))

    ax.scatter(data['grade'],data['situation'],cmap=plt.cm.Set1,marker='o',label='individual grades')
    for i,temp in data.iterrows():
        ax.annotate(temp['count'],(temp['grade'],temp['situation']),
                    textcoords='offset points',
                    xytext=(5,-3))
    ax.scatter(grade_means_sorted_validation['grade_mean'],
               grade_means_sorted_validation['situation'],
               color='red',marker='^',label='mean grade')
    ax.scatter(grade_means_sorted_validation['grade_mean_interpolation'],
               grade_means_sorted_validation['situation'],
               color='green',marker='s',label='mean interpolated grade')
    ax.set(title='Complexity grades given to traffic situations\nby controllers '+
                 'and labeled with\nmultiplicity ('+label_sorted+')',
           xlabel='controllers complexity grades',
           ylabel='traffic situations')
    ax.xaxis.set_ticks(range(1,6))
    ax.legend(loc='upper left')

plt.tight_layout();
controllers_grade = grades_interpolation.set_index('situation').join(grade_means_interpolation.set_index('situation'),
                                                                     how='left')
controllers_grade['diff'] = controllers_grade['grade'] - controllers_grade['grade_mean_interpolation']
# Mean absolute deviations from common mean accross 30 validation situations for each controller
controllers_diff_validation = controllers_grade_validation[['controller_id','diff']].abs().groupby('controller_id').mean()
controllers_diff_validation.sort_values(by='diff')
# Mean absolute deviations from common mean accross 30 situations for each controller
controllers_diff = controllers_grade[['controller_id','diff']].abs().groupby('controller_id').mean()
controllers_diff.sort_values(by='diff')
# Maximum deviations from common mean accross 30 validation situations for each controller
temp = [(row[1],row[0]) for row in controllers_grade_validation[['controller_id','diff']].abs().groupby
        ('controller_id').idxmax().reset_index().values]
controllers_diff_max_validation = controllers_grade_validation.reset_index().set_index(['situation','controller_id']).loc[temp]
controllers_diff_max_validation[['diff']].sort_values(by='diff')
```

```python
# Training on 90 random train situations
model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

features = features6
test_size = 30 # same number of situations as each controller had
n_folds = 100 # number of random subsets

# Training data (training situations)
features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

# Testing data (validation situations)
# features_targets_validation = features_validation.merge(grade_means_sorted_validation.set_index('situation'),on='situation')
# features_aligned_validation = features_targets_validation[features_targets_validation.columns.difference
#"(['grade_mean','grade_mean_interpolation'])]
# grades_aligned_validation = features_targets_validation['grade_mean_interpolation']

complexity_est = np.zeros((len(grade_means_interpolation),n_folds),dtype=np.float64)

# situations in grade_means_interpolation_validation are sorted by mean grade!
for k,situation in enumerate(grade_means_interpolation['situation'].values):

    # TODO: Exclude this particular situation!

    # Training is done on 90 random situations to introduce variation.
    # Testing is done on all 30 validation situations.
    for i in range(n_folds):

        # We use a train_test_split function to select subsets of situations
        situations_train, _ = train_test_split(features.index,test_size=test_size)

        # Assumption is that linear (rather than logistic) model is used
        # Training is on 90 random situations
        features_train_bool = features_aligned.index.isin(situations_train)
        X_train = features_aligned[features_train_bool].values
        y_train = grades_aligned[features_train_bool]

        # Testing is done on a single particular situation
        X_test = features_aligned.loc[situation].values
        y_test = grades_aligned.loc[situation]

        model.fit(X_train, y_train)
        w_est = model.named_steps['linear'].coef_

        # Estimate complexities on the validation hold-out set of situations
        scaler = StandardScaler().fit(X_test.reshape(1,-1))

        # GRADES ON THE SAME SCALE!
        complexity_est[k,i] = model.predict(X_test.reshape(1,-1))
# Maximum deviations from common mean accross 30 situations for each controller
temp = [(row[1],row[0]) for row in controllers_grade[['controller_id','diff']].abs().groupby
        ('controller_id').idxmax().reset_index().values]
controllers_diff_max = controllers_grade.reset_index().set_index(['situation','controller_id']).loc[temp]
controllers_diff_max[['diff']].sort_values(by='diff')
# Comparison of complexity grades given by different controllers to same traffic situations
# Our estimates are plotted as boxplots
# fig = plt.figure(figsize=(5,7))
fig = plt.figure(figsize=(5,25))

label_sorted='sorted by mean grade'
data=grade_duplicates_means

ax = plt.subplot2grid((1,1), (0,0))

ax.scatter(data['grade'],data['situation'],cmap=plt.cm.Set1,marker='o',label='individual grades')
for i,temp in data.iterrows():
    ax.annotate(temp['count'],(temp['grade'],temp['situation']),
                textcoords='offset points',
                xytext=(5,-3))
# for i,situation in enumerate(grade_means_sorted['situation'].values):
for i,situation in enumerate(grade_means_interpolation['situation'].values):
    ax.hlines(situation,
              xmin=np.percentile(complexity_est[i,:],5),
              xmax=np.percentile(complexity_est[i,:],95),
              color='gray')
```

```
ax.scatter(grade_means_sorted['grade_mean'],
           grade_means_sorted['situation'],
           color='red',marker='^',label='mean grade')
ax.scatter(grade_means_sorted['grade_mean_interpolation'],
           grade_means_sorted['situation'],
           color='green',marker='s',label='mean interpolated grade')
ax.set(title='Complexity grades given to traffic situations by\ncontrollers '+
             'and labeled with multiplicity\n('+label_sorted+') with 90% confidence intervals',
       xlabel='controllers complexity grades',
       ylabel='traffic situations')
ax.hlines(np.NaN,xmin=np.NaN,xmax=np.NaN,color='gray',label='90% confidence interval')
ax.xaxis.set_ticks(range(1,6))
ax.legend(loc='upper left')

plt.tight_layout();
model = Pipeline([('scaler', StandardScaler()),
                 ('linear', BayesianRidge())])

features = features6
test_size = 30 # same number of situations as each controller had
n_folds = 1000 # number of random subsets

features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

diff_estimates = np.zeros(n_folds,dtype=np.float64)
diff_max_estimates = np.zeros(n_folds,dtype=np.float64)
diff_extreme_estimates = np.zeros(n_folds,dtype=np.float64)

for i in range(n_folds):

    # We use a train_test_split function to select subsets of situations
    situations_train, situations_test = train_test_split(features.index,test_size=test_size)

    # We assume linear model instead of logistic!
    features_train_bool = features_aligned.index.isin(situations_train)
    X_train = features_aligned[features_train_bool].values
    y_train = grades_aligned[features_train_bool]

    # Test is equal for both logistic and linear model
    features_test_bool = features_aligned.index.isin(situations_test)
    X_test = features_aligned[features_test_bool].values
    y_test = grades_aligned[features_test_bool]

    model.fit(X_train, y_train)
    w_est = model.named_steps['linear'].coef_

    # Estimate complexities on the validation hold-out set of situations
    scaler = StandardScaler().fit(X_test)

    # GRADES ON THE SAME SCALE!
    complexity_est = model.predict(X_test)

    diff_estimates[i] = np.abs(complexity_est - y_test).mean()
    diff_max_estimates[i] = np.abs(complexity_est - y_test).max()

    # This will output the most extreme differences, whether positive or negative, with appropriate sign
    # These will later be separated in plotting to see whether there is any difference
    diff_extreme_estimates[i] = (complexity_est - y_test)[np.argmax(np.abs(complexity_est - y_test).values)]

fig = plt.figure(figsize=(5,7))

# Mean deviations
ax = plt.subplot2grid((4,1), (0,0))

ax.hist(diff_estimates,bins=np.linspace(0.25,0.70,40),label='model')
ax.set(title='Absolute AVERAGE differences accross controlers',
       xlabel='absolute AVERAGE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff['diff'].values,ymin=0,ymax=95,color='r')

fraction_higher = np.count_nonzero(diff_estimates > controllers_diff.max().values) / len(diff_estimates) * 100.0
fraction_lower = np.count_nonzero(diff_estimates < controllers_diff.min().values) / len(diff_estimates) * 100.0
ax.text(0.30,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(0.65,25,'{:.1f}'.format(fraction_higher)+'%')

# Max deviations
ax = plt.subplot2grid((4,1), (1,0))

ax.hist(diff_max_estimates,bins=np.linspace(0.70,2.3,30),label='model')
ax.set(title='Absolute MAX differences accross controlers',
       xlabel='absolute MAX difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max['diff'].abs().values,ymin=0,ymax=150,color='r')
```

```
fraction_higher = np.count_nonzero(diff_max_estimates > np.abs(controllers_diff_max['diff']).max()) /
len(diff_max_estimates) * 100.0
fraction_lower = np.count_nonzero(diff_max_estimates < np.abs(controllers_diff_max['diff']).min()) /
len(diff_max_estimates) * 100.0
ax.text(0.80,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.2,25,'{:.1f}'.format(fraction_higher)+'%')

# Max deviations - separate for positive and negative!
ax = plt.subplot2grid((4,1), (2,0))
ax.hist(np.abs(diff_extreme_estimates[diff_extreme_estimates<0.0]),bins=np.linspace(0.70,2.3,30),label='model')
ax.set(title='Extreme NEGATIVE differences accross controlers',
       xlabel='extreme NEGATIVE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max['diff'][controllers_diff_max['diff']<0.0].abs().values,ymin=0,ymax=150,color='r')

temp1 = np.abs(diff_extreme_estimates[diff_extreme_estimates<0.0])
temp2 = np.abs(controllers_diff_max['diff'][controllers_diff_max['diff'] < 0.0])
fraction_higher = np.count_nonzero(temp1 > temp2.max()) / len(temp1) * 100.0
fraction_lower = np.count_nonzero(temp1 < temp2.min()) / len(temp1) * 100.0
ax.text(0.80,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.2,25,'{:.1f}'.format(fraction_higher)+'%')

ax = plt.subplot2grid((4,1), (3,0))
ax.hist(diff_extreme_estimates[diff_extreme_estimates>=0.0],bins=np.linspace(0.70,2.3,30),label='model')
ax.set(title='Extreme POSITIVE differences accross controlers',
       xlabel='Extreme POSITIVE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max['diff'][controllers_diff_max['diff']>=0.0].abs().values,ymin=0,ymax=150,color='r')

temp1 = diff_extreme_estimates[diff_extreme_estimates>0.0]
temp2 = controllers_diff_max['diff'][controllers_diff_max['diff'] > 0.0]
fraction_higher = np.count_nonzero(temp1 > temp2.max()) / len(temp1) * 100.0
fraction_lower = np.count_nonzero(temp1 < temp2.min()) / len(temp1) * 100.0
ax.text(0.80,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.2,25,'{:.1f}'.format(fraction_higher)+'%')

plt.tight_layout();
# Groups of 30 situations
situations_group1 = ['A1','A10','A2','A3','A4','A5','A6','A7','A8','A9','B1','B10','B2','B3','B4','B5','B6','B7','B8',
                     'B9','C1','C10','C2','C3','C4','C5','C6','C7','C8','C9']
situations_group2 = ['A1','A11','A12','A13','A14','A15','A16','A2','A3','A4','B1','B11','B12','B13','B14','B15','B16',
                     'B2','B3','B4','C1','C11','C12','C13','C14','C15','C16','C2','C3','C4']
situations_group3 = ['A1','A17','A18','A19','A2','A20','A21','A22','A3','A4','B1','B17','B18','B19','B2','B20','B21',
                     'B22','B3','B4','C1','C17','C18','C19','C2','C20','C21','C22','C3','C4']
situations_group4 = ['A1','A2','A23','A24','A25','A26','A27','A28','A3','A4','B1','B2','B23','B24','B25','B26','B27',
                     'B28','B3','B4','C1','C2','C23','C24','C25','C26','C27','C28','C3','C4']
situations_group5 = ['A1','A2','A29','A3','A30','A31','A32','A33','A34','A4','B1','B2','B29','B3','B30','B31','B32',
                     'B33','B34','B4','C1','C2','C29','C3','C30','C31','C32','C33','C34','C4']
situations_group6 = ['A1','A2','A3','A35','A36','A37','A38','A39','A4','A40','B1','B2','B3','B35','B36','B37','B38',
                     'B39','B4','B40','C1','C2','C3','C35','C36','C37','C38','C39','C4','C40']
new_sector = pd.read_csv('data/new_sector_interpolated.csv')

temp = new_sector[['situation','new_sector_interpolated']].groupby(['situation'])\
                    .mean().rename(columns={'new_sector_interpolated':'mean'})\
                    .sort_values(by='mean')

temp = temp.join(features6,how='left')

features = temp[temp.columns.difference(['mean'])] # exclude target variable
target = temp['mean']
# Comparison of complexity grades given by different controllers to same traffic situations
# Our estimates are plotted as boxplots
# fig = plt.figure(figsize=(5,7))
fig = plt.figure(figsize=(6,5))

label_sorted = 'sorted by mean grade'

# TODO: Align situations with distributions!
selected_situations = ['A1','A2','A3','A4','B1','B2','B3','B4','C1','C2','C3','C4']
selected_situations_bool = grade_duplicates_means['situation'].isin(selected_situations)
data = grade_duplicates_means[selected_situations_bool]
# data2 = grade_means_sorted[selected_situations_bool]
data2 = grade_means_sorted[selected_situations_bool] # & grade_means_sorted['situation'].isnull() ]

data3 = complexity_est[grade_means_interpolation['situation'].isin(selected_situations)]

ax = plt.subplot2grid((1,1), (0,0))

# Plot annotations
ax.scatter(data['grade'],data['situation'],cmap=plt.cm.Set1,marker='o',label='individual grades')
for i,temp in data.iterrows():
    ax.annotate(temp['count'],(temp['grade'],temp['situation']),
                textcoords='offset points',
                xytext=(5,-3))
```

```
# Plot confidence intervals for estimates
# TODO: Situations should be aligned with distributions, but somehow there is a disprepency from the plot above?!
for i,situation in enumerate(grade_means_interpolation['situation'][grade_means_interpolation
                                                                    ['situation'].isin(selected_situations)].values):
    ax.hlines(situation,
              xmin=np.percentile(data3[i,:],5),
              xmax=np.percentile(data3[i,:],95),
              color='gray')

# Plot means
ax.scatter(data2['grade_mean'],
           data2['situation'],
           color='red',marker='^',label='mean grade')
ax.scatter(data2['grade_mean_interpolation'],
           data2['situation'],
           color='green',marker='s',label='mean interpolated grade')
ax.set(title='Complexity grades given to traffic situations by\ncontrollers '+
             'and labeled with multiplicity\n('+label_sorted+') with 90% confidence intervals',
       xlabel='controllers complexity grades',
       ylabel='traffic situations')
ax.hlines(np.NaN,xmin=np.NaN,xmax=np.NaN,color='gray',label='90% confidence interval')
ax.xaxis.set_ticks(range(1,6))
ax.legend(loc='lower right')

plt.tight_layout();
model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

features = features6
# test_size = 30 # same number of situations as each controller had
# n_folds = 1000 # number of random subsets

features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

situations_group = [situations_group1,situations_group2,situations_group3,situations_group4,situations_group5,situations_group6]

diff_estimates = np.zeros(len(situations_group),dtype=np.float64)
diff_max_estimates = np.zeros(len(situations_group),dtype=np.float64)

# for i in range(n_folds):
for i,situations in enumerate(situations_group):

    # We use a train_test_split function to select subsets of situations
    # situations_train, situations_test = train_test_split(features.index,test_size=test_size)

    # We assume linear model instead of logistic!
    features_train_bool = ~features_aligned.index.isin(situations)
    X_train = features_aligned[features_train_bool].values
    y_train = grades_aligned[features_train_bool]

    # Test is equal for both logistic and linear model
    features_test_bool = features_aligned.index.isin(situations)
    X_test = features_aligned[features_test_bool].values
    y_test = grades_aligned[features_test_bool]

    model.fit(X_train, y_train)
    w_est = model.named_steps['linear'].coef_

    # Estimate complexities on the validation hold-out set of situations
    scaler = StandardScaler().fit(X_test)

    # GRADES ON THE SAME SCALE!
    complexity_est = model.predict(X_test)

    diff_estimates[i] = np.abs(complexity_est - y_test).mean()
    diff_max_estimates[i] = np.abs(complexity_est - y_test).max()

model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

n_folds = 100
test_size = 30

new_sector_est = np.zeros([len(target.index),n_folds],dtype=np.float64)

for k,situation in enumerate(target.index):

    X = features.drop(situation).values
    y = target.drop(situation).values

    for i in range(n_folds):

        X_train, _, y_train, _ = train_test_split(X, y, test_size=test_size)

        model.fit(X_train, y_train)

        new_sector_est[k,i] = model.predict(features.loc[situation].values.reshape(1, -1))

controllers_diff
```

```
print('Average difference')
for i in range(6):
    print(controllers_diff.loc[i*3+1:i*3+3])
    print('model = ' + '{:.6f}'.format(diff_estimates[i]))
from IPython.display import display, Markdown
formula = ''
for w,x in zip(np.round(w_est,4),features6.columns.values):
    formula += '+'+str(w)+'`'+x+'\'` ' if w>0 else '-' + str(np.abs(w))+'`'+x+'` '
display(Markdown('`complexity` = '+formula))
# Dataframe that connects complexity estimated with logistic regression vs grades given by controllers
complexity_dataframe = pd.DataFrame(data={'situation':features.index,'complexity':complexity_est})

# Use original grades
# complexity_dataframe = pd.merge(complexity_dataframe,grades[['situation','grade']],on='situation',how='inner')

# Use interpolated grades
complexity_dataframe = pd.merge(complexity_dataframe,grades_interpolation[['situation','grade']],on='situation',how='inner')

features2.hist(figsize=(30,25));
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler()
X = features2
scaler.fit(X)
X_scaled = scaler.transform(X)

pca = PCA() # use option n_components=2 to calculate just first two principal components
pca.fit(X_scaled) # fit PCA model
X_pca = pca.transform(X_scaled) # transform data onto the principal components

features_pca = pd.DataFrame(X_pca, columns=['PC'+str(i) for i in range(1,X_pca.shape[1]+1)])

plt.figure(figsize=(7,5))

explained_variance_ratio = pca.explained_variance_ratio_
explained_variance_ratio_cummulative = np.cumsum(pca.explained_variance_ratio_)

plt.plot(range(1,X_pca.shape[1]+1), explained_variance_ratio, '-o', label='individual components', c='b')
plt.plot(range(1,X_pca.shape[1]+1), explained_variance_ratio_cummulative, '-s', label='cummulative', c='r')

plt.ylabel('fraction of explained variance')
plt.xlabel('principal component')
# plt.xlim(0.75,X_pca.shape[1]+1.25)
plt.xlim(0.75,20.0 + 1.25)
plt.ylim(0,1.05)
# plt.xticks(range(1,X_pca.shape[1]+1))
plt.xticks(range(1,20+1))
plt.legend(loc='center right')
plt.show()
# Calculating grade averages and ranks - it's safe to aggregate complexity as well as it's identical for each situation
complexity_dataframe_ranks = complexity_dataframe.groupby('situation').mean().rename(columns={'grade':'grade_mean'})
complexity_dataframe_ranks['rank_complexity'] = complexity_dataframe_ranks['complexity'].rank()
complexity_dataframe_ranks['rank_grade'] = complexity_dataframe_ranks['grade_mean'].rank()
complexity_dataframe_ranks.reset_index(level=0, inplace=True)
from IPython.display import display, Markdown

mean = ['{:.2f}'.format(x) for x in model.named_steps['scaler'].mean_]
std = ['{:.2f}'.format(x) for x in model.named_steps['scaler'].scale_]

# From standardized data to original data
# formula = '; '.join([ '`'+x+'`\'='+s+'`'+x+'`'+'+'+m for m,s,x in zip(mean,std,features6.columns.values)])

# From original data to standardized data
formula = '; '.join([ '`'+x+'`\'=(`'+x+'`'+'-'+m+')/'+s for m,s,x in zip(mean,std,features6.columns.values)])

display(Markdown(formula))
from matplotlib import cm as cm
from mpl_toolkits.axes_grid1 import make_axes_locatable

X_corr = features2.corr()
cmap = cm.get_cmap('RdBu', 30)

fig, ax = plt.subplots(1, 1, figsize=(12, 12))
iax = ax.imshow(X_corr, interpolation="nearest", cmap=cmap)
ax.grid(False)
ax.set(title='Feature correlation')
ax.set_xticks(range(X_corr.shape[1]))
ax.set_xticklabels(features2.columns.values, rotation=90)
ax.set_yticks(range(X_corr.shape[1]))
ax.set_yticklabels(features2.columns.values)
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(iax,cax)

plt.show()

# fig.savefig('figures/correlation_matrix.pdf',dpi=300,bbox_inches='tight');
```

```
corr_with_target = features2.merge(grade_means_sorted.set_index('situation'),on='situation')\
                            .corr(method='pearson')['grade_mean_interpolation'].sort_values(ascending=False)

fig = plt.figure(figsize=(15,3))

plt.scatter(corr_with_target.index,corr_with_target.values)
plt.xticks(corr_with_target.index, rotation=90)
plt.grid(axis='y')
plt.ylabel('Pearson correlation')
plt.title('Pearson correlation with the mean interpolated grade (target variable)')
plt.show();
# How many times was each traffic situation graded?
grade_count = grade_duplicates_means.groupby('situation')['count'].sum()
# Estimated complexity vs mean of controllers grades, in numerical values and ranks - Features v5
fig = plt.figure(figsize=(9, 5))

correlation_pearson = complexity_dataframe_ranks[['grade_mean','complexity']].corr(method='pearson').values[0,1]
correlation_spearman = complexity_dataframe_ranks[['grade_mean','complexity']].corr(method='spearman').values[0,1]

# Numerical values
ax1 = plt.subplot2grid((1,2), (0,0))
ax1.scatter(complexity_dataframe_ranks['grade_mean'], complexity_dataframe_ranks['complexity'], cmap=plt.cm.Set1)
ax1.text(1.6,5.0,'R (Pearsons) = '+'{:.3f}'.format(correlation_pearson))
ax1.set(title='Estimated complexity (v6) vs\nmean interpolated controllers grades',
       ylabel='complexity (linear regression)',
       xlabel='mean interpolated controllers grade')

# Ranks
ax2 = plt.subplot2grid((1,2), (0,1))
ax2.scatter(complexity_dataframe_ranks['rank_grade'], complexity_dataframe_ranks['rank_complexity'], cmap=plt.cm.Set1)
ax2.text(10,95,'R (Spearmans) = '+'{:.3f}'.format(correlation_spearman))
ax2.set(title='Ranks of estimated complexity (v6) vs\nranks of mean interpolated controllers grades',
       ylabel='rank of complexity (linear regression)',
       xlabel='rank of mean interpolated controllers grade')

plt.tight_layout();
fig = plt.figure(figsize=(6,6))

label_sorted='sorted by mean grade'
data=grade_duplicates_means_validation

ax = plt.subplot2grid((1,1), (0,0))

ax.scatter(target.values,new_sector_est.mean(axis=1),cmap=plt.cm.Set1,label='individual grades')
ax.plot([-3,2],[-3,2],color='black',lw=0.5,linestyle='dashed')

ax.set(title='Estimate of a new sector (interpolated)',
       xlabel='controllers estimates of new sector',
       ylabel='models estimates of new sector')

plt.tight_layout();
# Estimated complexity vs mean of controllers grades, in numerical values and ranks
# More or less the repetition of the previous scatterplots so we show it only for features v1

fig = plt.figure(figsize=(13, 5))

correlation_pearson = complexity_dataframe_ranks[['grade_mean','complexity']].corr(method='pearson').values[0,1]
correlation_spearman = complexity_dataframe_ranks[['grade_mean','complexity']].corr(method='spearman').values[0,1]

# cmap will generate a tuple of RGBA values for a given number in the range 0.0 to 1.0
# (also 0 to 255 - not used in this example).
# To map our z values cleanly to this range, we create a Normalize object.
cmap = matplotlib.cm.get_cmap('viridis')
normalize = matplotlib.colors.Normalize(vmin=min(grade_count.values), vmax=max(grade_count.values))
colors = [cmap(normalize(value)) for value in grade_count.values]

# Numerical values
ax1 = plt.subplot2grid((1,2), (0,0))
ax1.scatter(complexity_dataframe_ranks['grade_mean'],
            complexity_dataframe_ranks['complexity'],
            c=colors, cmap=cmap)
ax1.text(1.6,5.0,'R (Pearsons) = '+'{:.3f}'.format(correlation_pearson))
ax1.set(title='Estimated complexity (v6) vs\nmean interpolated controllers grades',
       ylabel='complexity (Bayesian ridge regression)',
       xlabel='mean interpolated controllers grade')
```

```python
cax, _ = matplotlib.colorbar.make_axes(ax1)
cbar = matplotlib.colorbar.ColorbarBase(cax, cmap=cmap, norm=normalize)
cbar.ax.set_title('Grade\ncount')

# Ranks
ax2 = plt.subplot2grid((1,2), (0,1))
ax2.scatter(complexity_dataframe_ranks['rank_grade'],
            complexity_dataframe_ranks['rank_complexity'],
            c=colors, cmap=cmap)
ax2.text(10,95,'R (Spearmans) = '+'{:.3f}'.format(correlation_spearman))
ax2.set(title='Ranks of estimated complexity (v6) vs\nranks of mean interpolated controllers grades',
       ylabel='rank of complexity (Bayesian ridge  regression)',
       xlabel='rank of mean interpolated controllers grade')

cax, _ = matplotlib.colorbar.make_axes(ax2)
cbar = matplotlib.colorbar.ColorbarBase(cax, cmap=cmap, norm=normalize)
cbar.ax.set_title('Grade\ncount')

# plt.tight_layout();
plt.show()
# Do controllers differ in the number of comparisons that they made?
temp = comparisons['controller'].value_counts()
pd.DataFrame({'controller':temp.index,'num_comparisons':temp.values}).sort_values(by='num_comparisons')
# Joining mean grades of each situation with the features describing the situation (tasks)

# Use original grades
# temp = grades[['situation','grade']].groupby(['situation']).mean()

# Use interpolated grades
temp = grades_interpolation[['situation','grade']].groupby(['situation']).mean()

temp = temp.join(features6,how='left')
grade_value = temp['grade'].values # include only target variable
model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

n_folds = 100
test_size = 30

new_sector_est = np.zeros([len(target.index),n_folds],dtype=np.float64)

for k,situation in enumerate(target.index):

    X = features.drop(situation).values
    y = target.drop(situation).values

    for i in range(n_folds):

        X_train, _, y_train, _ = train_test_split(X, y, test_size=test_size)

        model.fit(X_train, y_train)

        new_sector_est[k,i] = model.predict(features.loc[situation].values.reshape(1, -1))
new_sector = pd.read_csv('data/new_sector.csv')
# Training on 90 random train situations
model = Pipeline([('scaler', StandardScaler()),
                 ('linear', BayesianRidge())])

features = features11
features_validation = features11_validation # TODO: THIS OVERWRITES features_validation WHICH IS USED ELSEWHERE!
test_size = 30 # same number of situations as each controller had
n_folds = 1000 # number of random subsets

# Training data (training situations)
features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

# Testing data (validation situations)
features_targets_validation = features_validation.merge(grade_means_sorted_validation.set_index('situation'),on='situation')
features_aligned_validation = features_targets_validation[features_targets_validation.columns.difference
                                                           (['grade_mean','grade_mean_interpolation'])]
grades_aligned_validation = features_targets_validation['grade_mean_interpolation']

diff_estimates = np.zeros(n_folds,dtype=np.float64)
diff_max_estimates = np.zeros(n_folds,dtype=np.float64)
diff_extreme_estimates = np.zeros(n_folds,dtype=np.float64)

# Training is done on 90 random situations to introduce variation.
# Testing is done on all 30 validation situations.
for i in range(n_folds):
```

```
    # We use a train_test_split function to select subsets of situations
    situations_train, _ = train_test_split(features.index,test_size=test_size)

    # Assumtion is that linear (rather than logistic) model is used
    # Training is on 90 random situations
    features_train_bool = features_aligned.index.isin(situations_train)
    X_train = features_aligned[features_train_bool].values
    y_train = grades_aligned[features_train_bool]

    # Testing is done on 6 random validation situations
    _, situations_test = train_test_split(features_validation.index,test_size=6)
    features_test_bool = features_aligned_validation.index.isin(situations_test)
    X_test = features_aligned_validation[features_test_bool].values
    y_test = grades_aligned_validation[features_test_bool]

    model.fit(X_train, y_train)
    w_est = model.named_steps['linear'].coef_

    # Estimate complexities on the validation hold-out set of situations
    scaler = StandardScaler().fit(X_test)

    # GRADES ON THE SAME SCALE!
    complexity_est = model.predict(X_test)

    diff_estimates[i] = np.abs(complexity_est - y_test).mean()
    diff_max_estimates[i] = np.abs(complexity_est - y_test).max()

    # This will output the most extreme differences, whether positive or negative, with appropriate sign
    # These will later be separated in plotting to see whether there is any difference
    diff_extreme_estimates[i] = (complexity_est - y_test)[np.argmax(np.abs(complexity_est - y_test).values)]
# Load features of situations
features = pd.read_csv('data/features_count_v2.csv')\
              .pivot('situation','code','count')\
              .fillna(0.0).astype(np.float64)

# TODO: Complexities of aircrafts are calculated on a feature set which does not contain aircraft counts!
# However, this could be easily added, as aircraft count is one less than in original situation (the only
# issue whether the aircraft in question is inside or outside of the airspace).
data = new_sector.groupby(['situation','new_sector']).count()\
                 .rename(columns={'controller_id':'count'}).reset_index()

# Sorting situations by mean, used to order categorical axis in plots
sorted_situations = new_sector[['situation','new_sector']].groupby(['situation'])\
                    .mean().rename(columns={'new_sector':'mean'}).reset_index()\
                    .sort_values(by='mean')['situation'].values

fig = plt.figure(figsize=(5,18))
ax = plt.subplot2grid((1,1), (0,0))

# Ordering of categorical label is done in first invocation of barh
# So here we plot categories in sorted order as empty placeholders
ax.barh(sorted_situations,\
        width = 0,\
        height = 0, color="white",\
        align = "center", linewidth = 0)

ax.barh(data[data['new_sector']==1]['situation'].values,\
        width = data[data['new_sector']==1]['count'].values,\
        height = 0.8, color="red",\
        align = "center", linewidth = 0,
        label = 'open a new sector')

ax.barh(data[data['new_sector']==0]['situation'].values,\
        width = -data[data['new_sector']==0]['count'].values,\
        height = 0.8, color="steelblue",\
        align = "center", linewidth = 0,
        label = 'do not open a new sector')

ax.set(title='Controllers estimates for opening a new sector',
       xlabel='count',
       ylabel='situations')

ax.legend(loc='upper left')

plt.tight_layout();
# Load features of aircrafts
features_aircrafts = pd.read_csv('data/features_aircrafts.csv')
features_aircrafts = pd.pivot_table(features_aircrafts, index=['situation','aircraft'],columns='variable',
                                    values='value',fill_value=0.0)\
                       .astype(np.float64)
# features_aircrafts.head(12)
```

```python
fig = plt.figure(figsize=(6,6))

label_sorted='sorted by mean grade'
data=grade_duplicates_means_validation

ax = plt.subplot2grid((1,1), (0,0))

ax.scatter(target.values,new_sector_est.mean(axis=1),cmap=plt.cm.Set1,label='individual grades')
ax.plot([0,1],[0,1],color='black',lw=0.5,linestyle='dashed')

ax.set(title='Estimate of a new sector (discrete)',
       xlabel='controllers estimates of new sector (discrete)',
       ylabel='models estimates of new sector')

plt.tight_layout();
# Load features of aircrafts
features_aircrafts_CP = pd.read_csv('data/features_aircrafts_CP.csv')
features_aircrafts_CP = pd.pivot_table(features_aircrafts_CP, index=['situation','aircraft'],columns='variable',
                                       values='value',fill_value=0.0)\
                          .astype(np.int32)
# features_aircrafts_CP.head(12)
# Train on random 90 train situations
fig = plt.figure(figsize=(5,7))

# Mean deviations
ax = plt.subplot2grid((4,1), (0,0))

ax.hist(diff_estimates,bins=np.linspace(0.00,0.90,30),label='model')
ax.set(title='Absolute AVERAGE differences accross controlers',
       xlabel='absolute AVERAGE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_validation['diff'].values,ymin=0,ymax=150,color='r')

fraction_higher = np.count_nonzero(diff_estimates > controllers_diff_validation.max().values) / len(diff_estimates) * 100.0
fraction_lower = np.count_nonzero(diff_estimates < controllers_diff_validation.min().values) / len(diff_estimates) * 100.0
ax.text(-0.025,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(0.85,25,'{:.1f}'.format(fraction_higher)+'%')

# Max deviations
ax = plt.subplot2grid((4,1), (1,0))

ax.hist(diff_max_estimates,bins=np.linspace(0.00,2.2,30),label='model')
ax.set(title='Absolute MAX differences accross controlers',
       xlabel='absolute MAX difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max_validation['diff'].abs().values,ymin=0,ymax=150,color='r')

fraction_higher = np.count_nonzero(diff_max_estimates > np.abs(controllers_diff_max_validation['diff']).max()) /
len(diff_max_estimates) * 100.0
fraction_lower = np.count_nonzero(diff_max_estimates < np.abs(controllers_diff_max_validation['diff']).min()) /
len(diff_max_estimates) * 100.0
ax.text(-0.05,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.1,25,'{:.1f}'.format(fraction_higher)+'%')

# Max deviations - separate for positive and negative!
ax = plt.subplot2grid((4,1), (2,0))
ax.hist(np.abs(diff_extreme_estimates[diff_extreme_estimates<0.0]),bins=np.linspace(-0.10,2.2,30),label='model')
ax.set(title='Extreme NEGATIVE differences accross controlers',
       xlabel='extreme NEGATIVE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff']<
                                                  0.0].abs().values,ymin=0,ymax=130,color='r')

temp1 = np.abs(diff_extreme_estimates[diff_extreme_estimates<0.0])
temp2 = np.abs(controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff'] < 0.0])
fraction_higher = np.count_nonzero(temp1 > temp2.max()) / len(temp1) * 100.0
fraction_lower = np.count_nonzero(temp1 < temp2.min()) / len(temp1) * 100.0
ax.text(-0.1,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.05,25,'{:.1f}'.format(fraction_higher)+'%')

ax = plt.subplot2grid((4,1), (3,0))
ax.hist(diff_extreme_estimates[diff_extreme_estimates>=0.0],bins=np.linspace(-0.10,2.2,30),label='model')
ax.set(title='Extreme POSITIVE differences accross controlers',
       xlabel='Extreme POSITIVE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff']>=
                                                  0.0].abs().values,ymin=0,ymax=110,color='r')

temp1 = diff_extreme_estimates[diff_extreme_estimates>0.0]
temp2 = controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff'] > 0.0]
fraction_higher = np.count_nonzero(temp1 > temp2.max()) / len(temp1) * 100.0
fraction_lower = np.count_nonzero(temp1 < temp2.min()) / len(temp1) * 100.0
ax.text(0.25,50,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.05,25,'{:.1f}'.format(fraction_higher)+'%')

plt.tight_layout();
```

```python
grades = pd.read_csv('data/grades.csv')
grades_interpolation = pd.read_csv('data/grades_interpolation.csv')

grade_duplicates = grades[['situation','grade']].groupby(['situation','grade'],as_index=False).size().reset_index()
grade_duplicates = grade_duplicates.sort_values(by='situation')
grade_duplicates = grade_duplicates.rename(columns={0:'count'})

grade_means = grades[['situation','grade']].groupby(['situation'],as_index=False).mean()
grade_means = grade_means.rename(columns={'grade':'grade_mean'})

grade_means_interpolation = grades_interpolation[['situation','grade']].groupby(['situation'],as_index=False).mean()
grade_means_interpolation = grade_means_interpolation.rename(columns={'grade':'grade_mean_interpolation'})

grade_duplicates_means = pd.merge(grade_duplicates,grade_means,on='situation',how='inner')
grade_duplicates_means = pd.merge(grade_duplicates_means,grade_means_interpolation,on='situation',how='inner')\
                           .sort_values(by='grade_mean_interpolation')

grade_means_sorted = grade_duplicates_means[['situation','grade_mean','grade_mean_interpolation']].drop_duplicates(keep='first')
# grade_means_sorted.head(12)
# Estimate complexities of aircrafts using LOO crossvalidation

model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

situations = features_aligned.index

complexities_aircrafts = []

# We perform LOO crossvalidation after first excluding target situation from training set
for situation in situations:

    # Holds all crossvalidation scores for all aircrafts in a current situation
    complexity_est = np.zeros([len(situations)-1,len(features_aircrafts.loc[situation])],dtype=np.float64)

    for i,situation2 in enumerate(situations.drop(situation)):

        # Training data in a particular LOO loop
        X_train = features_aligned.drop(labels=[situation,situation2]).values
        y_train = grades_aligned.drop(labels=[situation,situation2]).values

        # Calculate complexities of situations where each of the aircrafts is missing
        X_test = features_aircrafts.loc[situation].values

        model.fit(X_train, y_train)

        # Produces grades on the same scale as the original data (statistically, not deterministically!)
        complexity_est[i,:] = model.predict(X_test)

    # print('\r'+situation,end='')

    # We hope that situations and aircrafts are in original order!
    complexities_aircrafts = np.concatenate((complexities_aircrafts,complexity_est.mean(axis=0)),axis=None)
# Join complexities_aircrafts with original index
complexities_aircrafts_cv = pd.DataFrame(complexities_aircrafts,index=features_aircrafts.index,columns=['complexity'])
# complexities_aircrafts_cv.head(12)
# Estimate complexities with the model trained on all of data

model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

# All situations are used for training the model!
X_train = features_aligned.values
y_train = grades_aligned.values

# Calculate complexities of situations where each of the aircrafts is missing
X_test = features_aircrafts.values

model.fit(X_train, y_train)
w_est = model.named_steps['linear'].coef_

# GRADES ON THE SAME SCALE!
complexity_est = model.predict(X_test)
# Complexities of situations based on full dataset - train data is used to estimate complexities
complexities_situations = pd.DataFrame([],index=features.index)
complexities_situations['complexity'] = model.predict(features)
# complexities_situations.head()
```

```python
complexities_aircrafts_cv['diff'] = np.nan # initialize difference column with NaN's

# How much does the removal of each individual aircraft decrease the complexity of situation
for situation, row in complexities_aircrafts_cv.iterrows():
    complexities_aircrafts_cv.loc[situation]['diff'] = row['complexity'] -
    complexities_situations.loc[situation[0]]['complexity']

# Sort by difference in complexity but within each situation
complexities_aircrafts_cv.sort_values(['situation','diff'],inplace=True)

# Join differences in complexity with the number of C and P conflicts for each aircraft
complexities_aircrafts_cv = complexities_aircrafts_cv.join(features_aircrafts_CP)

complexities_aircrafts_cv.xs('A1')

complexities_aircrafts_cv.xs('C39')
# Linear regression to infer weights

features = features6

X = features.values
y = grade_value # target

model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

n_folds = 600
test_size = 0.2
coeff_cross = np.zeros([n_folds,len(features.columns.values)],dtype=np.float64)
acc_cross = np.zeros(n_folds,dtype=np.float64)
acc_cross_balance = np.zeros(n_folds,dtype=np.float64)

for i in range(n_folds):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)

    model.fit(X_train, y_train)
    w_est = model.named_steps['linear'].coef_
    coeff_cross[i,:] = w_est

    acc_cross[i] = np.count_nonzero(~np.logical_xor(model.predict(X_test), y_test))/float(len(y_test))
    acc_cross_balance[i] = np.count_nonzero(y_test)/float(len(y_test))

# Training the model on full data
model.fit(X,y)
w_est = model.named_steps['linear'].coef_

# Produces grades on the same scale as the original data (statistically, not deterministically!)
complexity_est = model.predict(X)

# Complexity estimate for validation situations
complexity_est_validation = model.predict(features_validation.values)
# Choose here which starting feature set you want to use

# features1 - all conflict types and other numerical features
# features6 - only conflict types with aircraft counts

# features2 - all features with airspace counts

features_targets = features2.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]

# You can use grade_mean instead of grade_mean_interpolation as target here
grades_aligned = features_targets['grade_mean_interpolation']

k = 22 # number of best features we select
n_folds = 100 # number of random folds on which we perform crossvalidation
test_size = 0.2 # test size for crossvalidation

model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

# List with all the features and results array where we store best features and their cv scores
remaining_features = list(features_aligned.columns)
selected_features_scores = []

# We choose best k features one by one in greedy forward selection procedure
for i in range(k):
```

```python
    # Initialization
    best_score = 0.0
    best_feature = remaining_features[0]

    # Test all the remaining features one by one and then select the best one
    for current_feature in remaining_features:

        # Best currently selected features (by greedy forward selection)
        selected_features = [x[0] for x in selected_features_scores]

        # We add the current feature to the list of the best ones
        features_aligned_best = features_aligned[selected_features+[current_feature]]

        # Where we store cv scores in order to average them at the end
        corr_cross = np.zeros(n_folds,dtype=np.float64)

        # Crossvalidate on the currently selected features and store the score
        for fold in range(n_folds):

            # We use a train_test_split function to select subsets of situations
            situations_train, situations_test = train_test_split(features_aligned_best.index,test_size=test_size)

            # Train situations
            features_train_bool = features_aligned_best.index.isin(situations_train)
            X_train = features_aligned_best[features_train_bool].values
            y_train = grades_aligned[features_train_bool]

            # Test situations
            features_test_bool = features_aligned_best.index.isin(situations_test)
            X_test = features_aligned_best[features_test_bool].values
            y_test = grades_aligned[features_test_bool]

            model.fit(X_train, y_train)
            w_est = model.named_steps['linear'].coef_

            # Estimate complexities on the validation hold-out set of situations
            scaler = StandardScaler().fit(X_test)
            complexity_est = np.dot( scaler.transform(X_test), w_est )
            corr_cross[fold] = np.corrcoef(complexity_est,y_test)[0,1]

        # Current score is the mean of scores on k crossvalidation folds
        current_score = np.mean(corr_cross)

        # Keep track of the currently best feature in this iteration and its score
        if current_score >= best_score:
            best_feature = current_feature
            best_score = current_score

    # print('\rFeature '+str(i)+'/'+str(k),end='')

    selected_features_scores.append([best_feature,best_score])
    remaining_features.remove(best_feature)

# Features v2 - all features with aircraft counts
plt.figure(figsize=(7,5))

plt.plot(range(1, len(selected_features_scores) + 1), [x[1] for x in selected_features_scores], '-o', color='gray')
plt.xlabel('Features added one by one by greedy forward selection')
plt.ylabel('Pearson correlation with the true grades from test set')
plt.title('Recursive forward feature selection with crossvalidation')
plt.xticks(range(1,len(selected_features_scores)+1), [x[0] for x in selected_features_scores], rotation=90)
plt.show()
# Training on 90 random train situations
model = Pipeline([('scaler', StandardScaler()),
                  ('linear', BayesianRidge())])

features = features6 # Most similar to what we did with validation situations
test_size = 30 # same number of situations as each controller had
n_folds = 1000 # number of random subsets

# Training data (training situations)
features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

# Testing data (validation situations)
features_targets_validation = features_validation.merge(grade_means_sorted_validation.set_index('situation'),on='situation')
features_aligned_validation = features_targets_validation[features_targets_validation.columns.difference
                                                         (['grade_mean','grade_mean_interpolation'])]
grades_aligned_validation = features_targets_validation['grade_mean_interpolation']

diff_estimates = np.zeros(n_folds,dtype=np.float64)
diff_max_estimates = np.zeros(n_folds,dtype=np.float64)
diff_extreme_estimates = np.zeros(n_folds,dtype=np.float64)

# Training is done on 90 random situations to introduce variation.
# Testing is done on all 30 validation situations.
for i in range(n_folds):
```

```python
    # We use a train_test_split function to select subsets of situations
    situations_train, _ = train_test_split(features.index,test_size=test_size)

    # Assumtion is that linear (rather than logistic) model is used
    # Training is on 90 random situations
    features_train_bool = features_aligned.index.isin(situations_train)
    X_train = features_aligned[features_train_bool].values
    y_train = grades_aligned[features_train_bool]

    # Testing is done on 6 random validation situations
    _, situations_test = train_test_split(features_validation.index,test_size=6)
    features_test_bool = features_aligned_validation.index.isin(situations_test)
    X_test = features_aligned_validation[features_test_bool].values
    y_test = grades_aligned_validation[features_test_bool]

    model.fit(X_train, y_train)
    w_est = model.named_steps['linear'].coef_

    # Estimate complexities on the validation hold-out set of situations
    scaler = StandardScaler().fit(X_test)

    # GRADES ON THE SAME SCALE!
    complexity_est = model.predict(X_test)

    diff_estimates[i] = np.abs(complexity_est - y_test).mean()
    diff_max_estimates[i] = np.abs(complexity_est - y_test).max()

    # This will output the most extreme differences, whether positive or negative, with appropriate sign
    # These will later be separated in plotting to see whether there is any difference
    diff_extreme_estimates[i] = (complexity_est - y_test)[np.argmax(np.abs(complexity_est - y_test).values)]
# Train on random 90 train situaions
fig = plt.figure(figsize=(5,7))

# Mean deviations
ax = plt.subplot2grid((4,1), (0,0))

ax.hist(diff_estimates,bins=np.linspace(0.00,0.90,30),label='model')
ax.set(title='Absolute AVERAGE differences accross controlers',
       xlabel='absolute AVERAGE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_validation['diff'].values,ymin=0,ymax=150,color='r')

fraction_higher = np.count_nonzero(diff_estimates > controllers_diff_validation.max().values) / len(diff_estimates) * 100.0
fraction_lower = np.count_nonzero(diff_estimates < controllers_diff_validation.min().values) / len(diff_estimates) * 100.0
ax.text(-0.025,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(0.85,25,'{:.1f}'.format(fraction_higher)+'%')

# Max deviations
ax = plt.subplot2grid((4,1), (1,0))

ax.hist(diff_max_estimates,bins=np.linspace(0.00,2.2,30),label='model')
ax.set(title='Absolute MAX differences accross controlers',
       xlabel='absolute MAX difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max_validation['diff'].abs().values,ymin=0,ymax=120,color='r')

fraction_higher = np.count_nonzero(diff_max_estimates > np.abs(controllers_diff_max_validation['diff']).max()) /
len(diff_max_estimates) * 100.0
fraction_lower = np.count_nonzero(diff_max_estimates < np.abs(controllers_diff_max_validation['diff']).min()) /
len(diff_max_estimates) * 100.0
ax.text(-0.05,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.1,25,'{:.1f}'.format(fraction_higher)+'%')

# Max deviations - separate for positive and negative!
ax = plt.subplot2grid((4,1), (2,0))
ax.hist(np.abs(diff_extreme_estimates[diff_extreme_estimates<0.0]),bins=np.linspace(-0.10,2.2,30),label='model')
ax.set(title='Extreme NEGATIVE differences accross controlers',
       xlabel='extreme NEGATIVE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff']<0.0].abs().values,ymin=0,
          ymax=110,color='r')

temp1 = np.abs(diff_extreme_estimates[diff_extreme_estimates<0.0])
temp2 = np.abs(controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff'] < 0.0])
fraction_higher = np.count_nonzero(temp1 > temp2.max()) / len(temp1) * 100.0
fraction_lower = np.count_nonzero(temp1 < temp2.min()) / len(temp1) * 100.0
ax.text(-0.1,25,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.05,25,'{:.1f}'.format(fraction_higher)+'%')
```

```
ax = plt.subplot2grid((4,1), (3,0))
ax.hist(diff_extreme_estimates[diff_extreme_estimates>=0.0],bins=np.linspace(-0.10,2.2,30),label='model')
ax.set(title='Extreme POSITIVE differences accross controlers',
       xlabel='Extreme POSITIVE difference from mean')
ax.legend(loc='upper left')
ax.vlines(controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff']>=0.0].abs().values,ymin=0,
          ymax=110,color='r')

temp1 = diff_extreme_estimates[diff_extreme_estimates>0.0]
temp2 = controllers_diff_max_validation['diff'][controllers_diff_max_validation['diff'] > 0.0]
fraction_higher = np.count_nonzero(temp1 > temp2.max()) / len(temp1) * 100.0
fraction_lower = np.count_nonzero(temp1 < temp2.min()) / len(temp1) * 100.0
ax.text(0.25,50,'{:.1f}'.format(fraction_lower)+'%')
ax.text(2.05,25,'{:.1f}'.format(fraction_higher)+'%')

plt.tight_layout();
grade_duplicates_validation = grades_validation[['situation','grade']].groupby(['situation','grade'],
                                                                               as_index=False).size().reset_index()
grade_duplicates_validation = grade_duplicates_validation.sort_values(by='situation')
grade_duplicates_validation = grade_duplicates_validation.rename(columns={0:'count'})

grade_means_validation = grades_validation[['situation','grade']].groupby(['situation'],as_index=False).mean()
grade_means_validation = grade_means_validation.rename(columns={'grade':'grade_mean'})
grade_means_validation = grade_means_validation.sort_values(by='grade_mean')

grade_means_interpolation_validation = grades_interpolation_validation[['situation','grade']].groupby(['situation'],
                                                                                                      as_index=False).mean()
grade_means_interpolation_validation = grade_means_interpolation_validation.rename(columns={'grade':'grade_mean_interpolation'})
grade_means_interpolation_validation = grade_means_interpolation_validation.sort_values(by='grade_mean_interpolation')

# controllers_grade_validation = grades_interpolation_validation.set_index('situation').join
#(grade_means_interpolation_validation.set_index('situation'),how='left')
# Training on 90 random train situations
model = Pipeline([('scaler', StandardScaler()),
                 ('linear', BayesianRidge())])

features = features6 # Most similar to what we did with validation situations
test_size = 30 # same number of situations as each controller had
n_folds = 100 # number of random subsets

# Training data (training situations)
features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')
features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
grades_aligned = features_targets['grade_mean_interpolation']

# Testing data (validation situations)
features_targets_validation = features_validation.merge(grade_means_sorted_validation.set_index('situation'),on='situation')
features_aligned_validation = features_targets_validation[features_targets_validation.columns.difference
                                                          (['grade_mean','grade_mean_interpolation'])]
grades_aligned_validation = features_targets_validation['grade_mean_interpolation']

complexity_est = np.zeros((len(grade_means_interpolation_validation),n_folds),dtype=np.float64)

# situations in grade_means_interpolation_validation are sorted by mean grade!
for k,situation in enumerate(grade_means_interpolation_validation['situation'].values):

    # Training is done on 90 random situations to introduce variation.
    # Testing is done on all 30 validation situations.
    for i in range(n_folds):

        # We use a train_test_split function to select subsets of situations
        situations_train, _ = train_test_split(features.index,test_size=test_size)

        # Assumption is that linear (rather than logistic) model is used
        # Training is on 90 random situations
        features_train_bool = features_aligned.index.isin(situations_train)
        X_train = features_aligned[features_train_bool].values
        y_train = grades_aligned[features_train_bool]

        # Testing is done on a single particular situation
        X_test = features_aligned_validation.loc[situation].values
        y_test = grades_aligned_validation.loc[situation]

        model.fit(X_train, y_train)
        w_est = model.named_steps['linear'].coef_

        # Estimate complexities on the validation hold-out set of situations
        scaler = StandardScaler().fit(X_test.reshape(1,-1))

        # GRADES ON THE SAME SCALE!
        complexity_est[k,i] = model.predict(X_test.reshape(1,-1))
```

```
fig, axes = plt.subplots(7, 4, figsize=(12, 10))
ax = axes.ravel() # axes are 2-dimensional so we unfold them

# situations in grade_means_interpolation_validation are sorted by mean grade!
for k,situation in enumerate(grade_means_interpolation_validation['situation'].values):

    ax[k].hist(complexity_est[k,:],bins=np.linspace(0.00,5.00,50),label='model')
    ax[k].set(title=situation) #,
    #         xlabel='Estimated grades for situation '+situation)

    # temp = controllers_grade_validation.loc[situation]['grade']
    temp = grade_duplicates_validation[grade_duplicates_validation['situation']==situation]['grade'].values
    ax[k].vlines(temp,ymin=0,ymax=1,transform=ax[k].get_xaxis_transform(),color='r',linestyles='dashed',label='grades')

    # Apply a random jitter to distinguish between identical grades
    # ax.vlines((temp + np.random.normal(0,0.1,len(temp))).values,ymin=0,ymax=1,transform=ax.get_xaxis_transform(),color='r')

    # grade_mean_interpolation is the same for all controllers within situation!
    # temp = controllers_grade_validation.loc[situation]['grade_mean_interpolation'].values[0]
    temp = grade_means_validation[grade_means_validation['situation']==situation]['grade_mean'].values
    ax[k].vlines(temp,ymin=0,ymax=1,transform=ax[k].get_xaxis_transform(),color='r',linestyles='solid',label='mean grade')


    temp = grade_means_interpolation_validation[grade_means_interpolation_validation['situation']==
                                                 situation]['grade_mean_interpolation'].values
    ax[k].vlines(temp,ymin=0,ymax=1,transform=ax[k].get_xaxis_transform(),color='g',linestyles='solid',label=
                 'mean interpolated grade')

    ax[k].set_yticks(()) # remove ticks on y-axis

    # ax[k].legend(loc='upper left') # does not fit on such small graphs! :-)

fig.tight_layout()
# Comparison of complexity grades given by different controllers to same traffic situations
# Our estimates are plotted as boxplots
fig = plt.figure(figsize=(5,7))

label_sorted='sorted by mean grade'
data=grade_duplicates_means_validation

ax = plt.subplot2grid((1,1), (0,0))

ax.scatter(data['grade'],data['situation'],cmap=plt.cm.Set1,marker='o',label='individual grades')
for i,temp in data.iterrows():
    ax.annotate(temp['count'],(temp['grade'],temp['situation']),
                textcoords='offset points',
                xytext=(5,-3))
for i,situation in enumerate(grade_means_sorted_validation['situation'].values):
    ax.hlines(situation,
              xmin=np.percentile(complexity_est[i,:],5),
              xmax=np.percentile(complexity_est[i,:],95),
              color='gray')
ax.scatter(grade_means_sorted_validation['grade_mean'],
           grade_means_sorted_validation['situation'],
           color='red',marker='^',label='mean grade')
ax.scatter(grade_means_sorted_validation['grade_mean_interpolation'],
           grade_means_sorted_validation['situation'],
           color='green',marker='s',label='mean interpolated grade')
ax.set(title='Complexity grades given to traffic situations by\ncontrollers '+
             'and labeled with multiplicity\n('+label_sorted+') with 90% confidence intervals',
       xlabel='controllers complexity grades',
       ylabel='traffic situations')
ax.hlines(np.NaN,xmin=np.NaN,xmax=np.NaN,color='gray',label='90% confidence interval')
ax.xaxis.set_ticks(range(1,6))
ax.legend(loc='upper left')

plt.tight_layout();
```

```python
# Joint evaluation of all models

model1 = Pipeline([('scaler', StandardScaler()),
                   ('logistic', LogisticRegression(solver='liblinear',fit_intercept=True))])
model2 = Pipeline([('scaler', StandardScaler()),
                   ('logistic', LogisticRegression(solver='liblinear',fit_intercept=True))])
model3 = Pipeline([('scaler', StandardScaler()),
                   ('logistic', LogisticRegression(solver='liblinear',fit_intercept=True))])
model4 = Pipeline([('scaler', StandardScaler()),
                   ('linear', BayesianRidge())])
model5 = Pipeline([('scaler', StandardScaler()),
                   ('linear', BayesianRidge())])
model6 = Pipeline([('scaler', StandardScaler()),
                   ('linear', BayesianRidge())])
model7 = Pipeline([('scaler', StandardScaler()),
                   ('linear', BayesianRidge())])
model8 = Pipeline([('scaler', StandardScaler()),
                   ('linear', BayesianRidge())])
model9 = Pipeline([('scaler', StandardScaler()),
                   ('linear', BayesianRidge())])
model10 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
model11 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
model12 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
 model13 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
 model14 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
 model15 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
 model16 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])
 model17 = Pipeline([('scaler', StandardScaler()),
                    ('linear', BayesianRidge())])

 n_folds = 300
 test_size = 0.2

 # Not using them for evaluation, only later when we estimate feature weights!
 # coeff_cross4 = np.zeros([n_folds,len(features4.columns.values)],dtype=np.float64)
 # coeff_cross5 = np.zeros([n_folds,len(features5.columns.values)],dtype=np.float64)

 corr_cross1 = np.zeros(n_folds,dtype=np.float64)
 corr_cross2 = np.zeros(n_folds,dtype=np.float64)
 corr_cross3 = np.zeros(n_folds,dtype=np.float64)
 corr_cross4 = np.zeros(n_folds,dtype=np.float64)
 corr_cross5 = np.zeros(n_folds,dtype=np.float64)
 corr_cross6 = np.zeros(n_folds,dtype=np.float64)
 corr_cross7 = np.zeros(n_folds,dtype=np.float64)
 corr_cross8 = np.zeros(n_folds,dtype=np.float64)
 corr_cross9 = np.zeros(n_folds,dtype=np.float64)
 corr_cross10 = np.zeros(n_folds,dtype=np.float64)
 corr_cross11 = np.zeros(n_folds,dtype=np.float64)
corr_cross12 = np.zeros(n_folds,dtype=np.float64)
corr_cross13 = np.zeros(n_folds,dtype=np.float64)
corr_cross14 = np.zeros(n_folds,dtype=np.float64)
corr_cross15 = np.zeros(n_folds,dtype=np.float64)
corr_cross16 = np.zeros(n_folds,dtype=np.float64)
corr_cross17 = np.zeros(n_folds,dtype=np.float64)

for features,model,model_type,corr_cross in zip([features1,features2,features3,features4,features5,features6,features7,
                                                  features8,features9,features10,features11,features12,features13,features15,
                                                  features15,features16,features17],
                                                 [model1,model2,model3,model4,model5,model6,model7,model8,model9,model10,model11,
                                                  model12,model13,model14,model15,model16,model17],
                                                 ['logistic','logistic','logistic','linear','linear','linear','linear',
                                                  'linear','linear','linear','linear','linear','linear','linear','linear',
                                                  'linear','linear'],
                                                 [corr_cross1,corr_cross2,corr_cross3,corr_cross4,corr_cross5,corr_cross6,
                                                  corr_cross7,corr_cross8,corr_cross9,corr_cross10,corr_cross11,corr_cross12,
                                                  corr_cross13,corr_cross14,corr_cross15,corr_cross16,corr_cross17]):

    # Merging features with grades (targets)

    # Version 1 - Grade means
    # features_targets = features.merge(grade_means.set_index('situation'),on='situation')

    # Version 2 - Grade means interpolation
    features_targets = features.merge(grade_means_sorted.set_index('situation'),on='situation')

    # Now we separate features from targets, knowing that values are properly aligned
```

```
    # Version 1 - Grade means
    # features_aligned = features_targets[features_targets.columns.difference(['grade_mean'])]
    # grades_aligned = features_targets['grade_mean']

    # Version 2 - Grade means interpolation
    features_aligned = features_targets[features_targets.columns.difference(['grade_mean','grade_mean_interpolation'])]
    grades_aligned = features_targets['grade_mean_interpolation']

    for i in range(n_folds):

        # We use a train_test_split function to select subsets of situations
        situations_train, situations_test = train_test_split(features.index,test_size=test_size)

        # Logistic models learn on comparison data
        if (model_type=='logistic'):

            # Logical vector which indexes which comparisons are in train set
            comparisons_train_bool = comparisons['situation1'].isin(situations_train) &\
                                     comparisons['situation2'].isin(situations_train)

            # Train comparisons have both situations from our training set
            comparisons_train = comparisons[comparisons_train_bool]

            # Test comparisons can have situations from both train and test set (or only test set)
            comparisons_test = comparisons[~comparisons_train_bool]

            X_train = features.loc[comparisons_train['situation2'].values].values-\
                      features.loc[comparisons_train['situation1'].values].values

            y_train = comparisons_train['comparison'].values

        # Linear regression models learn on original situation data
        if (model_type=='linear'):

            features_train_bool = features_aligned.index.isin(situations_train)

            X_train = features_aligned[features_train_bool].values

            y_train = grades_aligned[features_train_bool]

        # Test is equal for both logistic and linear model
        features_test_bool = features_aligned.index.isin(situations_test)
        X_test = features_aligned[features_test_bool].values
        y_test = grades_aligned[features_test_bool]

        model.fit(X_train, y_train)
        w_est = model.named_steps[model_type].coef_
        if (model_type=='logistic'):
            w_est = w_est[0]
        # coeff_cross[i,:] = w_est

        # Estimate complexities on the validation hold-out set of situations
        scaler = StandardScaler().fit(X_test)
        complexity_est = np.dot( scaler.transform(X_test), w_est )
        corr_cross[i] = np.corrcoef(complexity_est,y_test)[0,1]
# Crossvalidation score distributions for all models

# Crossvalidation scores for all models
corr_cross_array = [corr_cross1,corr_cross2,corr_cross3,corr_cross4,corr_cross5,corr_cross6,corr_cross7,corr_cross8,
                    corr_cross9,corr_cross10,corr_cross11,corr_cross12,corr_cross13,corr_cross14,corr_cross15,corr_cross16,
                    corr_cross17]

fig = plt.figure(figsize=(5,18))

for i,corr_cross in enumerate(corr_cross_array):

    # Marginal distribution of crossvalidation scores
    ax = plt.subplot2grid((len(corr_cross_array),1), (i,0))

    ax.hist(corr_cross,bins=np.linspace(0.45,0.97,30))
    ax.set(title='Model v' + str(i+1) + ', mean = '+'{:.3f}'.format(np.mean(corr_cross)),
           xlim=[0.45,0.97])

    ax.axvline(np.mean(corr_cross),color='r')

plt.tight_layout();
```

```
# How many times is one model better than the other in terms of their crossvalidation scores

# Choose the two models
corr_cross_v1 = corr_cross6
corr_cross_v2 = corr_cross11

first_model_name = 'features6'
second_model_name = 'features11'

fig = plt.figure(figsize=(12,6))
ax = plt.subplot2grid((2,2), (0,0), rowspan=2)

# Scatter plot of joint distribution of crossvalidation scores
ax.scatter(corr_cross_v1,corr_cross_v2,cmap=plt.cm.Set1)

success_ratio = np.count_nonzero((corr_cross_v1-corr_cross_v2)>0)/len(corr_cross_v1)

ax.set(title='600-fold crossvalidation Pearson correlation for two models',
       xlabel='Bayesian ridge regression model '+first_model_name+' (better in ' +
       '{:.1f}'.format(100.0*success_ratio) + '% cases)',
       ylabel='Bayesian ridge regression model '+second_model_name+' (better in ' +
       '{:.1f}'.format(100.0*(1.0-success_ratio)) + '% cases)',
       xlim=[0.55,0.97],
       ylim=[0.55,0.97])
ax.plot([0.55,0.97],[0.55,0.97],'r',linestyle='--')

# Marginal distribution of crossvalidation scores of first model
ax = plt.subplot2grid((2,2), (0,1))
ax.hist(corr_cross_v1,bins=np.linspace(0.55,0.97,30))
ax.set(title='Bayesian ridge regression model '+first_model_name+' (mean = '+'{:.3f}'.format(np.mean(corr_cross_v1))+')',
       xlim=[0.55,0.97])

ax.axvline(np.mean(corr_cross_v1),color='r')

# Marginal distribution of crossvalidation scores of second model
ax = plt.subplot2grid((2,2), (1,1))
ax.hist(corr_cross_v2,bins=np.linspace(0.55,0.97,30))
ax.set(title='Bayesian ridge regression model '+second_model_name+' (mean = '+'{:.3f}'.format(np.mean(corr_cross_v2))+')',
       xlim=[0.55,0.97])

ax.axvline(np.mean(corr_cross_v2),color='r')

plt.tight_layout();
# Dataframe that connects complexity estimated with logistic regression vs grades given by controllers
complexity_dataframe_validation = pd.DataFrame(data={'situation':features_validation.index,
                                                     'complexity':complexity_est_validation})

# Use interpolated grades
complexity_dataframe_validation = pd.merge(complexity_dataframe_validation,grades_interpolation_validation
                                           [['situation','grade']],on='situation',how='inner')

# Calculating grade averages and ranks - it's safe to aggregate complexity as well as it's identical for each situation
complexity_dataframe_ranks_validation = complexity_dataframe_validation.groupby('situation').mean().rename
(columns={'grade':'grade_mean'})
complexity_dataframe_ranks_validation['rank_complexity'] = complexity_dataframe_ranks_validation['complexity'].rank()
complexity_dataframe_ranks_validation['rank_grade'] = complexity_dataframe_ranks_validation['grade_mean'].rank()
complexity_dataframe_ranks_validation.reset_index(level=0, inplace=True)
# Estimated complexity vs mean of controllers grades, in numerical values and ranks - Features v6
fig = plt.figure(figsize=(10, 5))

correlation_pearson_validation = complexity_dataframe_ranks_validation
[['grade_mean','complexity']].corr(method='pearson').values[0,1]
correlation_spearman_validation = complexity_dataframe_ranks_validation
[['grade_mean','complexity']].corr(method='spearman').values[0,1]

# Numerical values
ax1 = plt.subplot2grid((1,2), (0,0))
ax1.scatter(complexity_dataframe_ranks_validation['grade_mean'], complexity_dataframe_ranks_validation
            ['complexity'], cmap=plt.cm.Set1)
ax1.text(1.1,4.3,'R (Pearsons) = '+'{:.3f}'.format(correlation_pearson_validation))
ax1.set(title='Estimated complexity (v6) vs mean interpolated\ncontrollers grades (validation situations)',
       ylabel='complexity (linear regression)',
       xlabel='mean interpolated controllers grade\n(validation situations)')

# Ranks
ax2 = plt.subplot2grid((1,2), (0,1))
ax2.scatter(complexity_dataframe_ranks_validation['rank_grade'], complexity_dataframe_ranks_validation['rank_complexity'],
            cmap=plt.cm.Set1)
ax2.text(2,27,'R (Spearmans) = '+'{:.3f}'.format(correlation_spearman_validation))
ax2.set(title='Ranks of estimated complexity (v6) vs ranks of mean\ninterpolated controllers grades (validation situations)',
       ylabel='rank of complexity (linear regression)',
       xlabel='rank of mean interpolated controllers grade\n(validation situations)')

plt.tight_layout();
# Complexity dataframe with only the situations that are shared among all controllers
complexity_dataframe_ranks_shared = complexity_dataframe_ranks[complexity_dataframe_ranks['situation'].isin(situations_shared)]
complexity_dataframe_ranks_shared['rank_shared'] = complexity_dataframe_ranks_shared['complexity'].rank()
# complexity_dataframe_ranks_shared
```

```
# Add rankings of these shared situations from our statistical model to the ones from controllers
# NOTE: It appears that our statistical model has identical rankings to controller 13!
temp = rankings_shared.sort_values(['controller_id','situation'])['rank_within'].values.reshape((18,12))
temp = np.vstack([temp,complexity_dataframe_ranks_shared['rank_shared'].values])
# temp
# Correlation matrix between all controllers for their rankings of 12 common situations
# NOTE: Controller 19 is our statistical model!
# corr_matrix = pd.DataFrame.from_dict(temp.T).corr(method='spearman').values

# plt.imshow(corr_matrix)
# plt.xticks(range(len(corr_matrix)),range(1,len(corr_matrix)+1)) # To show all x-tick labels
# plt.yticks(range(len(corr_matrix)),range(1,len(corr_matrix)+1)) # To show all y-tick labels
# plt.colorbar();
# TODO: The cells on the edge are only half-drawn, this is a problem in matplotlib v3.1.1. which
# is fixed by upgrading to v3.1.2.

# Permuting the rows and columns of correlation matrix so that the more similar controllers are at top
# NOTE: Controller 19 is our statistical model!
corr_matrix = pd.DataFrame.from_dict(temp.T).corr(method='spearman').values

p = corr_matrix.mean(axis=1).argsort()[::-1]
for i in range(19): corr_matrix[:,i] = corr_matrix[p,i]
for i in range(19): corr_matrix[i,:] = corr_matrix[i,p]

plt.imshow(corr_matrix)
plt.xticks(range(len(p)),p+1)
plt.yticks(range(len(p)),p+1)
plt.colorbar();
pd.set_option('display.float_format', lambda x: '%.2f' % x)
pd.DataFrame(corr_matrix,columns=p+1,index=p+1)
# Features v6, from Bayesian ridge regression
fig, axes = plt.subplots(15, 5, figsize=(15, 20)) # both task type and extra features (76 features)
ax = axes.ravel() # axes are 2-dimensional so we unfold them
for i in range(len(features.columns.values)):
    ax[i].hist(coeff_cross[:,i], bins=np.linspace(-0.13,0.13,50))
    ax[i].set(title=features.columns[i])
    ax[i].set_yticks(()) # remove ticks on y-axis
fig.tight_layout()
```

Author biography

Bruno Antulov-Fantulin, master of aeronautical engineering was born on the 10th of August in 1988 in Slavonski Brod, the Republic of Croatia. He graduated in July 2013 at the Faculty of Transport and Traffic Sciences, University of Zagreb where he was also employed in December 2014 as a teaching assistant at the Department of Aeronautics. In February 2015, he enrolled in the postgraduate doctoral study at the Faculty of Transport and Traffic Sciences, University of Zagreb which he finished with summa cum laude in October 2020. He published several scientific papers and is the author of the first patent pending application for the Faculty of Transport and Traffic Sciences, University of Zagreb.