ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Suchanová** Jméno: **Barbora** Osobní číslo: **420309**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Návrh uživatelského rozhraní pro filtrování a vizualizaci dat o dopravě extrahovaných z kamerových záznamů

Název diplomové práce anglicky:

Design of user interface for filtering and visualization of traffic data extracted from camera recordings

Pokyny pro vypracování:

Analyze the user interface of the GoodVision Video Insights service for visual analysis of traffic data extracted from camera recordings. Determine what data and tasks are used in the visual analysis process and identify the requirements and use cases of the potential target group using the application. Further, analyze existing techniques for visualization and visual analysis of traffic data. Based on your analysis, design and develop a high-fidelity prototype of a user interface for an intuitive formulation of queries over the traffic data extracted from camera recordings. The prototype will be using real data extracted from camera recordings. The user interface should be able to present the results of the queries in a visual form as well as in a form of reports. Verify your design of the user interface with usability study performed with 6 to 10 participants.

Seznam doporučené literatury:

Arnowitz J., M. Arent, and N. Berger, Effective Prototyping for Software Makers. Elsevier Science & Technology, 2007.
Munzner T., Visualization analysis and design. AK Peters/CRC Press, 2014.
Kuniavsky M., Observing the user experience: a practitioner's guide to user research. Elsevier, 2003.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Ladislav Čmolík, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **19.02.2021**

Ing. Ladislav Čmolík, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studentky

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Master Thesis

Design of User Interface for Filtering and Visualization of Traffic Data Extracted from Camera Recordings

Bc. Barbora Suchanová

Supervisor: Ing. Ladislav Čmolík, Ph.D.

Study Programme: Open Informatics, Master's degree

Field of Study: Software Engineering

May 22, 2019

Aknowledgements

I would like to sincerely thank my supervisor of this thesis, Ing. Ladislav Čmolík, Ph.D., for his patient guidance and continuous support during the last two semesters. Next, my thanks go to Daniel Štofan and Lukáš Hrubý, for giving me the opportunity to research this unique problem within the perspective project GoodVision Video Insights. Finally, I would like to greatly thank my boyfriend and my family for their constant support.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 22, 2019 .....

Abstract

With the increasing growth of population, big cities have to deal with the traffic optimisation every day. To achieve that, fast and accurate traffic data collection and analysis is fundamental. Nowadays, one of the progressive solutions is automatic mining of these data from camera recordings. The data gathering is just as important as the comprehensive filtering and visualization of the data that allows to understand the data for further analysis.

The goal of this project is to create a user interface (UI) for visualization and filtering of spatiotemporal traffic data extracted from camera recordings and incorporate it within the existing solution - Video Insights. To design the UI, I collected requirements from the target group interviews, from the research of alternative solutions and its visualization methods. As a result, I identified the main scenarios and defined the system specification for the future UI. Based on the analysis, I designed the low-fidelity and high-fidelity prototypes for non-trivial use cases. Finally, I implemented the user interface in the Video Insights solution and verified the design via the qualitative usability testing with 9 participants.

Abstrakt

Se zvyšujícím se počtem obyvatelstva jsou velká města nucena zabývat se čím dál více optimalizací dopravy. Základem pro to je rychlé a přesné získávání dat z dopravy. Jedno z aktuálních progresivních řešení je automatické dolování těchto dat z kamerových záznamů. Tak jako je důležité samotné získávání dopravních dat, je stejně důležité i filtrování a vizualizace těchto komplexních dat pro jejich pochopení a následnou analýzu.

Cílem této práce je vytvoření uživatelského rozhraní (UR) pro vizualizaci a filtrování prostorově-časových dopravních dat z kamerových záznamů a dále jeho integrace do existujícího řešení - Video Insights. Pro návrh jsem shromáždila požadavky od cílové skupiny, alternativních řešení a jejich vizualizačních metod. Výsledkem toho jsem identifikovala hlavní scénáře a specifikovala systém pro budoucí UR. Na základě analýzy jsem vytvořila několik low-fidelity a high-fidelity prototypů pro netriviální případy užití. Toto navržené uživatelské UR jsem následně zaintegrovala do Video Insights řešení a ověřila design během kvalitativního uživatelského testování s celkem 9 participanty.

Contents

1 Introduction **1**
1.1 Tasks Assignment . . . 4

2 Analysis **5**
2.1 Target Audience . . . 5
2.2 Traffic Engineering Terminology . . . 7
2.3 Collection of Requirements . . . 7
2.3.1 Current State . . . 8
2.3.2 In Person Interviews . . . 10
2.3.3 Alternative Solutions . . . 12
2.4 System Specification . . . 16
2.4.1 Scenarios . . . 16
2.4.2 Object Model . . . 17
2.4.3 Functional model . . . 18
2.5 Data Visualization . . . 23
2.5.1 Data Definition . . . 23
2.5.2 Task Mapping . . . 24
2.5.3 Task Visualization . . . 25
2.5.4 Visualization Challenges . . . 26

3 Architecture and Design **29**
3.1 Service Architecture . . . 29
3.1.1 Detector and Tracker Component . . . 29
3.1.2 Database . . . 31
3.1.3 Application Programming Interface . . . 31
3.1.4 Web Application . . . 32
3.2 User Interface . . . 33
3.2.1 Low-fidelity Prototypes . . . 33
3.2.2 Heuristic Evaluation . . . 35
3.2.3 High-fidelity Prototypes . . . 37
3.2.4 Redefining the Terminology . . . 43

4 Implementation **45**
4.1 Environment . . . 45
4.2 Third Party Libraries . . . 46
4.3 Client-Server Communication . . . 47

4.4 User Interface 47
4.4.1 Create a Movement 47
4.4.2 Create a Scenario 48
4.4.3 Create a Configurable Traffic Report 49
4.4.4 Visualization Challenges Solutions 51
4.5 Deployment 52

5 Usability Testing **53**
5.1 Preparation 53
5.1.1 Approach 53
5.1.2 Setup 54
5.2 Participants 55
5.3 Testing Scenarios 55
5.4 Testing Procedure 57
5.5 Findings 57
5.6 Conclusion 61

6 Conclusion **63**
6.1 Future Work 63

A Nomenclature **69**

Chapter 1

Introduction

In the last 50 years the world population doubled its size to nearly 8 billion. It is the first time in human history when we deal with such an exponential growth of population [13]. Even though the growth peak was reached in the mid 60s and from that time the percentage growth is slightly declining, we are still going to meet the magical boundary of 10 billion people in less than 30 years. Particularly, this growth will primarily apply to urban areas located in Asia and Africa [16].

Inevitably, the growth of urban areas relates to an expansion of its city centers and its suburbs. Due to astronomical prices of residences in the city centre, the majority of urban residents will live in the suburbs and they will be forced to commute to work located mostly in the city centre. Megalopolises like that are facing this urban-rural migration problem already. In less than 30 years, the number of agglomerations will grow and many new cities will be dealing with this issue on a daily basis too [3].

The objectives of this issue are clear - minimize the transport time from place to place while maximizing inhabitants comfort during daily commuting. Perfectly projected transportation infrastructure accessible to all, including people from the farthest locations might be one of the solutions. However, design of such an infrastructure is almost never planned from scratch or in advance. Current needs are the real game changers. These determine if the roads need to be expanded or if new underground stations should be developed. As a rule, civil engineers are then introduced to existing transportation infrastructure in some state and their task is to improve, modify or completely rebuild the infrastructure in order to relieve the current transportation problems.

To recognize the current needs in time or preferably in the near future, responsible people need to have access to relevant data. Information about the traffic situation of the specific place and its surroundings in a long term is one of them. Current approaches to gather these data usually incorporate installation of hardware sensors, such as pressure hoses, piezoelectric sensors, radars or custom hardware. Sadly, even today in 2019, human surveyors are still used for traffic surveys in specific countries, including the Czech Republic. In Figure 1.1, you can see screenshots from the mobile application used by some human surveyors since 2016. However, this digitization is still very rare among surveyors and more commonly they still use a paper notebook. Therefore, the traffic data collection can last several months to process the results. Moreover, human

surveyors do not have a reliable or stable accuracy. The unpredictability of the human factor is a common issue, surveyors get tired or they just simply fake the counts. I was present at one of the surveys in the field when it was raining and all surveyors were hiding under the roof and writing down random counts.

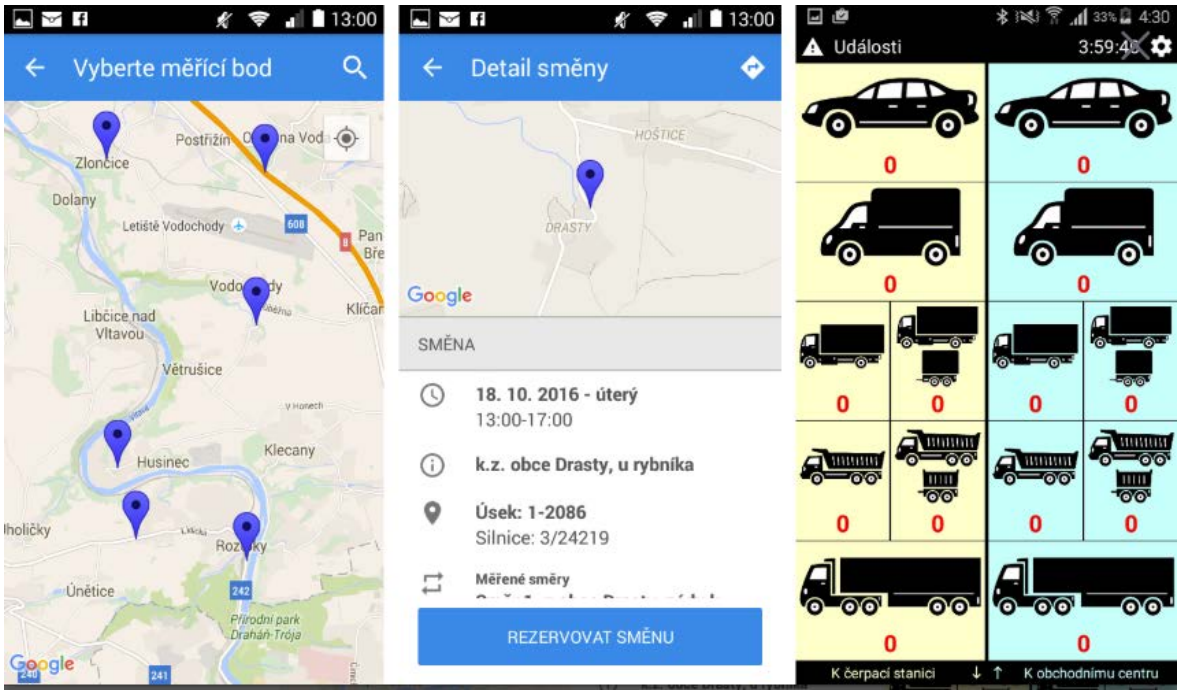

Figure 1.1: Mobile application by IPSOS used for human surveyors as part of the nationwide census for the Road and Motorway Directorate of the Czech Republic in 2016

The problem of manual counting inefficiency might be solved by one rising and tangible approach using current technologies and possibilities. It is to automatically collect traffic data from camera recordings. This AI-powered method is based on computer vision and deep learning algorithms trained to automatically detect, recognize (see Figure 1.2) and track objects (see Figure 1.3).

Generally, solutions using computer vision for extracting information from camera footage are obtaining more attention thanks to development of high performance GPUs as well to increasing global trend of traffic and surveillance cameras installation. Currently, there is more than 500 millions surveillance cameras installed globally. Moreover, it is predicted that in the next five years the world will be equipped with at least 52% more of these cameras than it is today [7]. More importantly, this method allows to represent each video and information about its detected objects in a structured way. For instance, every unique object can be described via positions on the scene in time forming a trajectory (Figure 1.3) as well as a specific object class (Figure 1.2). Therefore, if there is a tool that can visualize the structured data in a understandable way to a traffic analytic for instance, there is no need to store the raw video after its processing, which might be a significant advantage with regard to used storage and much more.

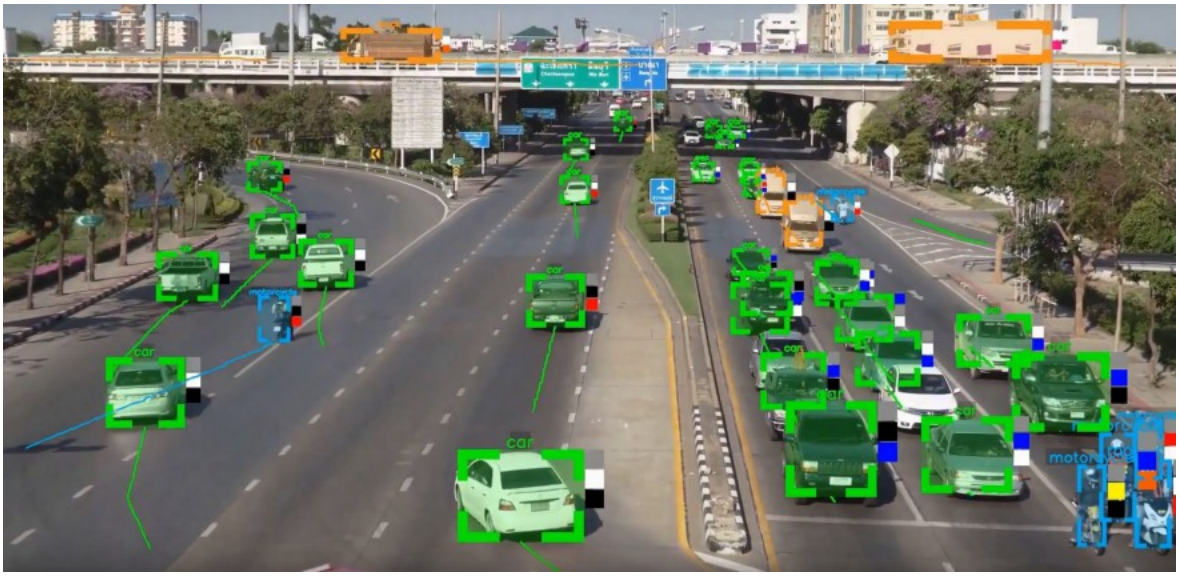

Figure 1.2: Side product after video processing with a detector displaying detected objects via bounding box. Various colours represent different object classes. Green represents cars whereas blue represents motorcycles.

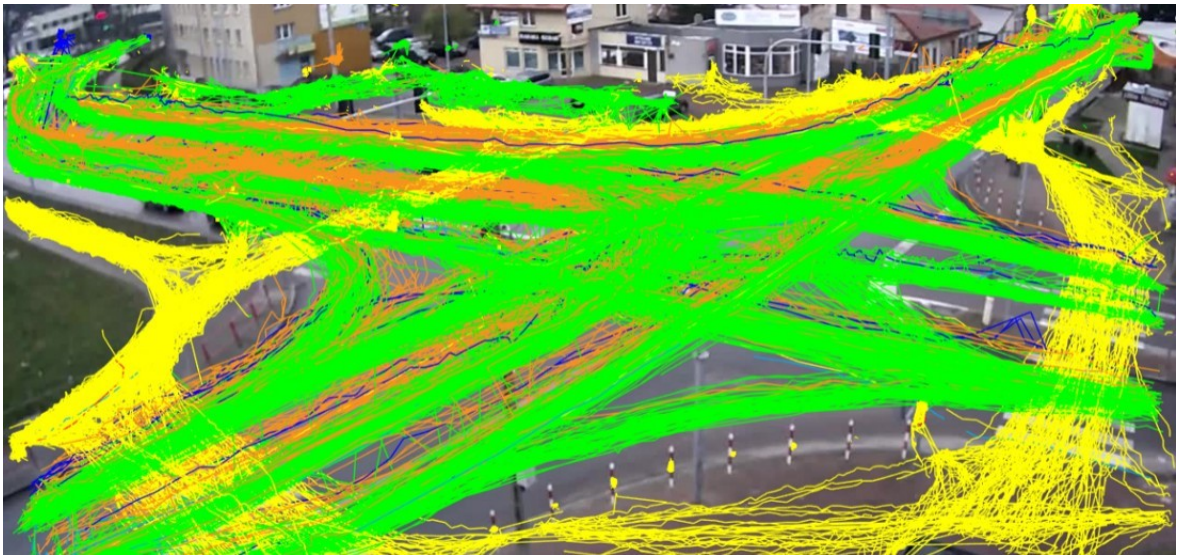

Figure 1.3: Visualization of object positions in time forming trajectories. Colors of trajectories represent different object classes as on the picture above.

In order to understand all these extracted data in a context and recognize their meaning and potential in the field of traffic analysis, it is required to visualize these data in a coherent manner readable for a human eye as just mentioned. Besides that, custom filtering of objects detected in a video might unveil traffic patterns that would otherwise stay hidden. There are many use cases and opportunities within this field and to cover them all via one visualization tool might seem a difficult task.

The tool GoodVision Video Insights[1] focused on automated traffic data collection and analysis offers an automated traffic data extraction as well as a basic visualization tool capable of displaying the mined traffic data. Therefore, it might be a good starting point for developing such a general tool that would cover the needs of the majority of traffic analysts and their use cases.

Finally, I believe this approach of mining data from camera recordings might introduce a completely new view on spatiotemporal data and traffic analytics in general. Therefore, I am highly engaged to analyze the existing solution Video Insights just mentioned and design and implement a new intuitive user interface offering the possibility to display, filter and report the automatically collected traffic data extracted from the camera recordings. As a result, decision-makers in the field of traffic analytics might maximize their efficiency when dealing with any traffic issues anywhere or it might even help them to recognize hidden information that could prevent traffic issues in advance.

1.1 Tasks Assignment

The goal of this thesis is to firstly collect the requirements of users working in the field of traffic analysis. To achieve that, I should analyze the existing user interface of the Video Insights, interview the target group in person and additionally research similar systems working with traffic data to see the functions available as well as other techniques in the traffic data visualization. Based on the collected requirements, I am supposed to identify the main scenarios as well as the system entities and functional model containing all use cases the future UI of Video Insights should support. Afterwards, I should describe available spatiotemporal data of detected objects and with regards to the defined scenarios, formulate tasks that can be mapped to a proper visualization methods. As a result, I should introduce suitable visualization methods for the classified tasks.

In the second part of the thesis, I am supposed to describe the Video Insights architecture and design the user interface based on the analysis. There should be low-fidelity prototypes displaying step by step system-user interaction for non-trivial use cases as well as their evaluation based on heuristics. The high-fidelity prototyping iteration should follow. Finally, I should incorporate the proposed user interface into the Video Insights solution and verify the design via usability testing performed with 6 to 10 participants.

[1] More about GoodVision Video Insights - <https://goodvisionlive.com/>

Chapter 2

Analysis

In this chapter, I will present the results of the analysis performed to describe a target audience of the system, such as Video Insights. Afterwards, I will summarize the collected requirements from in-person interviews and from the analysis of the alternative solutions. Consequently, I am going to define the main end-to-end user's scenarios and the object and functional model of the system. Activity diagrams for complex use cases will be appended below their definitions. In the second half of the chapter, I will focus on the data classification and tasks definition for a selected use cases to be able to determine suitable visualization techniques used in the UI. As a bonus, I will discuss the visualization challenges in the field of traffic data visualization.

2.1 Target Audience

As mentioned previously in the *Introduction*, traffic optimization is the key domain where automated traffic data collection and analytics might be useful. Due to complexity of this domain, this is not a business to end-customers (B2C) market but rather a business to business market (B2B). More specifically, the traffic optimization discipline includes fields, such as traffic management, traffic surveys & analyses, traffic simulations or even location analysis. Besides that, civil engineering is a discipline that is tightly linked to traffic optimization. For instance, road construction, urban planning and strategy demand a long term data about traffic situation, too.

Within these fields, I see the biggest potential in companies offering services performed now by human workers including primarily repetitive data collection, verification, and sorting. These processes can be easily replaced by robotic process automation (RPA) while bringing lower error rate and higher efficiency at the same time.

Therefore, all companies that execute at least some of their traffic data collection processes manually can be considered to be part of the target audience too. To list some of them as well as companies providing services for traffic optimization, see the list below.

- Traffic surveys and analyses - IPSOS [1], Tracsis [2]
- Transport analysis - TRALYS [3]

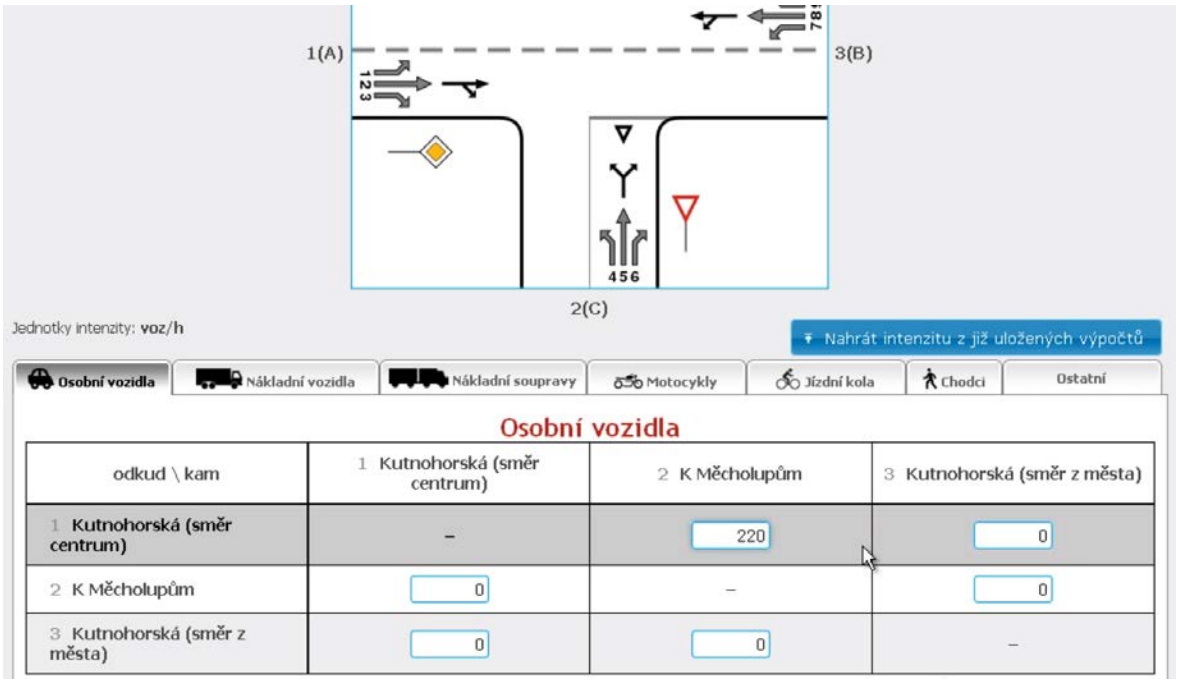

Figure 2.1: TRALYS software for generating traffic reports from manually entered data.

- Urban planning - Prague Institute of Planning and Development [4]
- Acoustic studies - Studio D - akustika s.r.o [5]
- Traffic simulations - PTV Vissim [6], TRANSYT [7]

[1] More about IPSOS - <https://bit.ly/2OUMBrY>
[2] More about Tracsis - <https://tracsistraffic.com/home/traffic-surveys/>
[3] More about TRALYS - <http://tralys.cz/>
[4] More about IPR <http://en.iprpraha.cz/clanek/1453/department-of-infrastructure>
[5] More about Studio D <https://www.akustikad.com/noise-measurement/>
[6] More about PTV Vissim <https://www.ptvgroup.com/en/solutions/products/ptv-vissim/>
[7] More about TRANSYT<https://trlsoftware.com/products/junction-signal-design/>

2.2 Traffic Engineering Terminology

Due to the specific field of traffic engineering the system deals with, I will use many terms in the thesis that might not be known to general public. For that reason, I list the basic traffic engineering terms [1] with their explanation in Table 2.2.

Traffic term	Definition
Origin-destination count	Number of vehicles* traveling between two locations
Origin-destination matrix	Origin-destination counts for all-pairs of locations
Traffic volume	Count of vehicles* on a road segment
Movement	The direction of the traffic flow
Through movement	Traffic flow going straight ahead within the intersection
Left-turn movement	Traffic flow turning to the left within the intersection
Right-turn movement	Traffic flow turning to the right within the intersection
Lane	One-directional lane on the road defined by painted lines
Entry zone	The place where the vehicles enter to an intersection or a roundabout
Exit zone	The place where the vehicles exit from an intersection or a roundabout
Downstream	In the direction of the movement
Upstream	In the opposite direction of the movement
Transit time	Time spent by vehicles traveling between two or more locations
Occupancy time	Amount of time that vehicles* occupy a specific location
Space length	Distance between two successive vehicles
Space time	Time distance between two successive vehicles
Near miss	A narrowly avoided collision between two vehicles*

Table 2.2: The basic set of terms of the traffic engineering terminology.

*valid also for bicycles and pedestrians

I will use most of the terms mentioned above in scenarios and use cases. Regarding the system, it should implement these terms appropriately to stay consistent with the existing traffic-focused software solutions.

2.3 Collection of Requirements

In every software development life cycle, the first step is always same, it starts with a collection of requirements. In this section, I will present the requirements that I collected directly from the target audience, from the current Video Insights solution and from alternative solutions. Based on that, I will introduce the main scenarios and the system specification in the following section.

2.3.1 Current State

As the first source for collecting requirements is naturally the current state of the application I am going to extend and redesign. Regarding the background, the development of the web application Video Insights (as well as the rest of the service) started approximately in autumn 2017. Since then, it has slowly developed into a minimal viable product (MVC) and continuously into a publicly available product. However, there are still only basic and limited options to display and filter the traffic data which correspond to needs of a small group of clients.

In Figure 2.2, screenshots depict the main dashboard for visual analytics and the immutable reporting page. On both pages, user is able to change the global filter and see almost immediately the results. This approach is sufficient for use cases where the users are 'exploring and browsing' a relatively short video (up to 24 hours) in order to understand the common situation on the roads. Nevertheless, it might not be suitable for the other use cases that I am going to reveal later on during the in-person interviews. The following research should help me to understand the requirements of the target group thoroughly and design the whole new concept of traffic data visualization, filtering and reporting being fully scalable. Moreover, the new design should be robust regarding the future extensions as well as compliant with the majority of traffic use cases and the requirements I will collect during my analysis.

To summarize, the **main disadvantages** of the current Video Insights system are

- Advanced filtering of the traffic data is not supported
- Immutable reports page displaying only predefined statistics that might not be relevant
- Users are not sure how to use the tool properly
- Users feel that they have to wait a long time when requesting visual data or statistics from a long video
- Users want to define the main traffic flows but the system does not support directional cross filters
- Traffic data visualization issues[8]

Therefore, the **requirements** based on the experience with the Video Insights are

- Support advanced filtering of the traffic data (including directional filters)
- Support configurable reporting page displaying only statistics user is really interested in
- Ensure users know how to use the tool properly
- Split the scene description phase and the data analysis phase within the system
- Make user focus on scene description once and then primarily on the data analysis within the system

[8] More discussed in the following section *Visualization challenges*

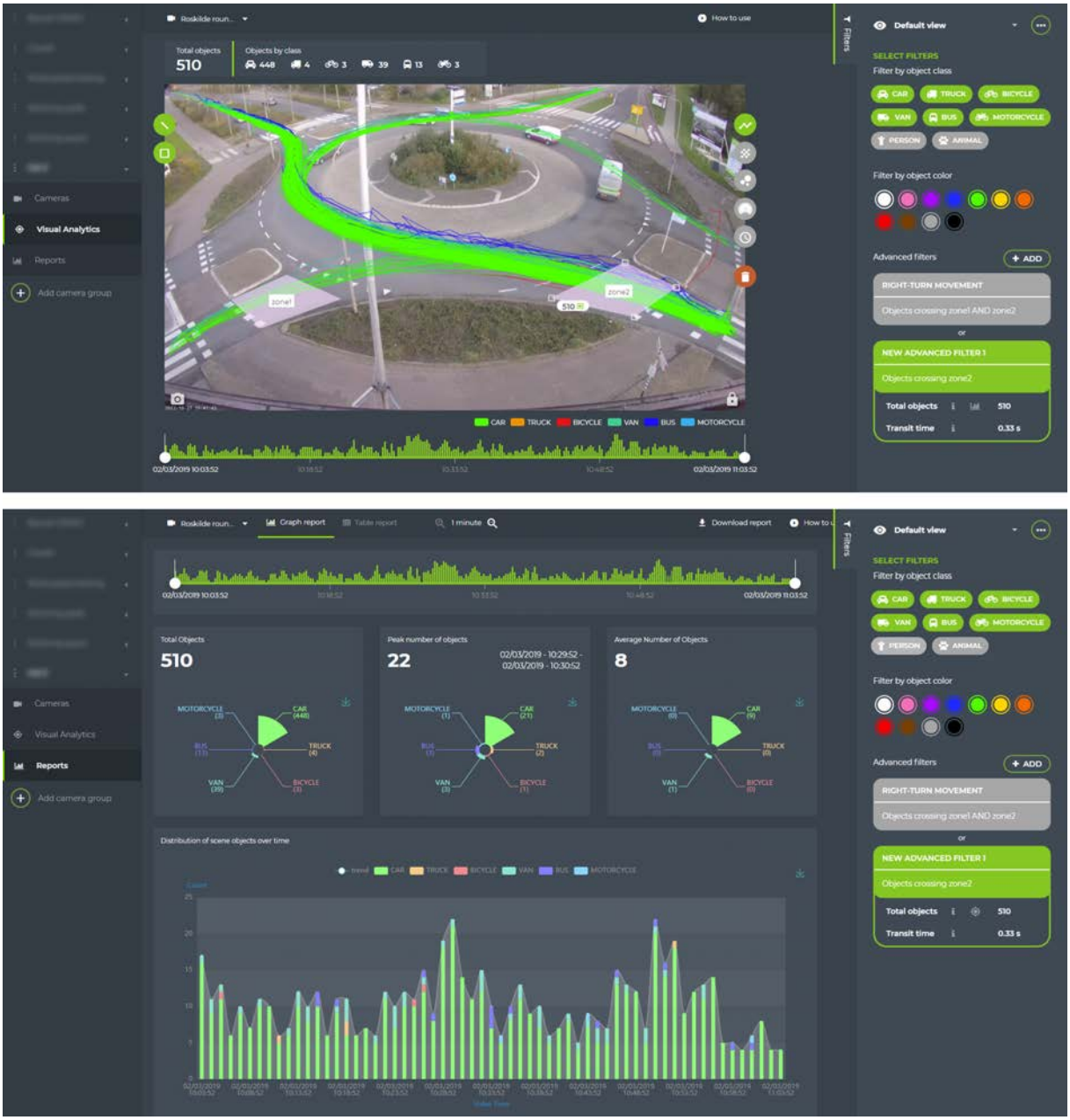

Figure 2.2: Current Video Insights dashboards displaying visualization of the traffic data with the basic filtering options (upper screenshot) together with the immutable reporting dashboard (lower screenshot).

2.3.2 In Person Interviews

As the second source for collecting requirements is to talk with the current and potential users. Therefore, I managed to contact and meet in person with representatives of two companies working on a daily basis with traffic-related data that they either process or edit and subsequently deliver to their customers (more details below). It is worth mentioning that both companies are already slightly familiar with the existing solution GoodVision Video Insights providing automated traffic data collection and analytics from camera recordings.

The first meeting was held in South Bohemia, the domicile of a company that creates on demand analysis or reports regarding sound acoustics indoor as well outdoor. This company sees potential of services like Video Insights mainly for the fast data retrieval from their camera recordings and possible data export to their proprietary traffic reports. If their customer's project is traffic related, they are obliged to attach a traffic report with their sound report. The traffic data they are interested in are related to the task, such as:

- They want to know the traffic volume in time on a specific road together with the object class distribution (car, truck, motorcycle, etc.)
- They want to know the approximate speed of the counted vehicles.
- They want to associate the traffic volume (including classes) to recorded sound in the same area.

I discovered that until today they have had to retrieve traffic data via industrial radars placed above the road. This approach offers accessing the data including vehicle types almost in real-time but on the other hand the data arrive in raw format and its acquisition is quite expensive.

The second company is focused primarily on transport analysis. Particularly, their portfolio consists of several applications that provide a GUI application where their customers manually enter the data about traffic volumes and transit times in specific locations. Each application can then export a table report for the specific location. In terms of tasks, they demand:

- They want to know the traffic volumes (downstream and upstream) in all defined movements together with the object class attribute (car, truck, motorcycle, etc.) in a specific time period.
- They want to know the transit times in defined movements.
- They want to see the traffic volumes in a context (map, scheme, etc.).

The biggest issue for their clients is the manual entry of the data via complex forms. The number of individual boxes in each form can reach up to one hundred, which is very time-consuming. Therefore, they find automated traffic data retrieval from camera recordings as a possible solution that can be incorporated in their solution.

Both interviews took approximately one hour and they helped me a lot to understand the real client's needs and requirements regarding the system and the UI.

Traffic engineering conference

The second round of in-person interviews took place in Birmingham where every two years one of the biggest traffic engineering events in the world is held - Traffex [9]. More than 350 companies as exhibitors and around 10 thousand professionals in the field of traffic engineering and architecture planning including highways engineers, traffic surveyors, researchers, consultants, contractors and many more attend this event every two years. This is a perfect place to gather the most up-to-date insights regarding the current and future traffic optimization trends.

Fortunately, I had the possibility to attend this event held between 2 and 4 April this year to collect even more requirements and insights for the system Video Insights from real prospective clients. Therefore, below I list a short overview of the facts and conclusions that I summarized based on the approximately 80 in-person interviews with people interested in the automated traffic data collection and traffic analytics that I met during the three days of the Traffex exhibition.

- To achieve a successful traffic optimization independently on the location (highway or intersection), there is a set of basic traffic analytics that is used repeatedly by majority of traffic engineers and planners, such as origin-destination counts, transit times for movements, etc.
- One of the most common traffic surveys for collecting data about traffic jams is done via recording only peak hours - very early morning and then afternoon due to people commuting to and from a work.
- Detailed one-time traffic survey for an x-million city might comprise up to 300 intersections (cameras) with approximately 15 hours each, resulting in almost 5000 hours of traffic data the system should be prepared to display and report in a coherent way.
- There is a higher need for a more detailed object classification, particularly to recognize a car model with high fidelity. This might be achieved via Automatic Number-Plate Recognition (ANPR) followed by a search in a proprietary car registry or via higher emphasis on user to upload only zoomed videos where the cars are distinguishable for current model classification algorithms.
- Near misses counts analytics is valuable information wanted primarily by traffic intersection engineers. Its main properties are a sudden deceleration or acceleration of the vehicles in the traffic flow or a movement and a small time distance between two objects in two different movements. Nevertheless, the exact definition vary across the traffic use cases and countries.
- There is still a high number of companies mediating basic traffic surveys via manual human surveyors. Particularly, one of the leading traffic surveys companies employs thousands of Indian workers to count traffic volume from about 600 million hours of video every year.
- Regarding manual traffic surveyors, they are still needed for very specific use cases that cannot be yet automatized by computer vision [10]. Specifically, occupancy of vehicles (number of people per vehicle) is a very rare information used in a highway design in the US primarily.

[9] More about - <https://www.traffex.com/>
[10] More about <https://tracsistraffic.com/survey_jobs/7871/>

- The accuracy of manual traffic surveyors is around 80 %. Therefore, the slight error rate made by a computer vision or by user in the system is acceptable.
- There is an increasing number of camera manufacturers offering cheaper and smaller cameras with embedded web servers that is able to upload the video stream in batches directly to existing endpoint with a minimal delay.
- There is a need to react to current traffic situation in time where in time does not mean in real-time but in a few hours delay ideally. Streaming of the video directly to the cloud service that subsequently processes the video in batches and serves it in the application is one of the possible solutions.

2.3.3 Alternative Solutions

Other solutions handling traffic data might be a rich source of information and inspiration in terms of functionality as well as user interface. Therefore, I introduce here four similar solutions to GoodVision Video Insights:

- **Miovision**[11], a company based in Canada and Germany, has been on the market for more than 15 years. This company provides traffic data from camera recordings as well as several traffic operation-related products (DataLink, Traffic Insights and others). Some of the products are focused on real-time video processing and traffic analytics for Automated Traffic Signal Performance Measures (ATSPMs). For instance, users can see overall traffic volumes, travel times and more on predefined intersections. Besides that, the DataLink system for visualization of the automatically collected traffic data shows the pre-defined traffic volumes of movements via colorful flows with the width corresponding to the traffic volume as it is depicted in Figure 2.3.

  In case a user wants to validate the accuracy of traffic volumes against the real footage in the given time range, a corresponding video is available there next to the flow diagram. It is an interesting fact that the client can watch the footage without the fear from sensitive data handling. This is possible due to the fact, that Miovision requires to purchase their proprietary hardware equipment including the cameras with embedded algorithms able of Automatic Number-Plate Recognition (ANPR) as well as face detection. That is why after the recording, Miovision claims, they automatically blur all the sensitive data (vehicle registration plates and faces)in the video.

  On the other hand, I do not find Miovision products (including UI) much flexible and robust due to many people involved in the process of traffic data collection as well as in the process of traffic insights generation. For instance, in order to obtain a specific origin-destination counts, user has to define it by email before the actual video recording including definition of all desired traffic movements and object classes user wants to count. In the next step, Miovision technicians manually configure the specified movements which might result for example in the intersection flow diagram as depicted in 2.3. To conclude, Miovision is a great source of traffic analytics as well as a useful inspiration in terms of visualization techniques when handling traffic data. However, from the UX point of view, most of the processes in the application do not follow the User Centered Design (UCD [6]) due to dependency on other technicians during the video processing.

[11] More about Miovision <https://miovision.com/>

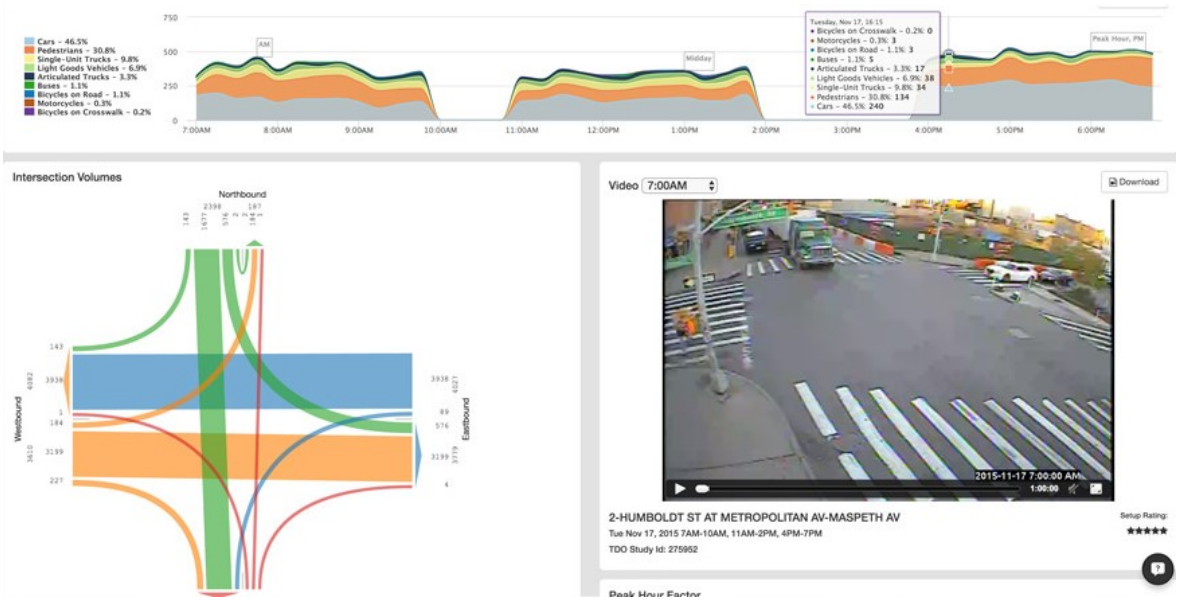

Figure 2.3: DataLink by Miovision showing intersection volumes on the left and camera preview on the right

- **DataFromSky**[12], a company providing the automated traffic data collection and analytics only from the video footage that is recorded by their own drones. Advantage is that the video footage is recorded from the bird's-eye view (See 2.4) and therefore the outcome is supposed to be better in readability for a human eye - it reminds a satellite map. Furthermore, the top view enables to measure reliably the speed of vehicles if the height from the ground to drone is known and the communication profile is flat.

  However, due to a limited battery in their drones, the length of the continuous footage is limited to 12 hours. Therefore, also their software is built to process and display the data from short camera recordings even though the traffic survey duration lasts typically a several days or even more. Besides that, their software for visualization of extracted traffic data does not follow the Nielsen heuristic [11] *Aesthetic and minimalist design* due to the main dashboard that is overwhelmed with too much information about each detected object, despite that not all information is equally relevant to the user (Figure 2.4). Thanks to a freely available demo, I can confirm that the UI is not intuitive (the user needs a manual or a workshop to understand it) and it is limited by functions as well. In conclusion, I find the way how they identify the main object flows as well as the individual vehicles inspiring.

[12]More about DataFromSky <http://datafromsky.com/>

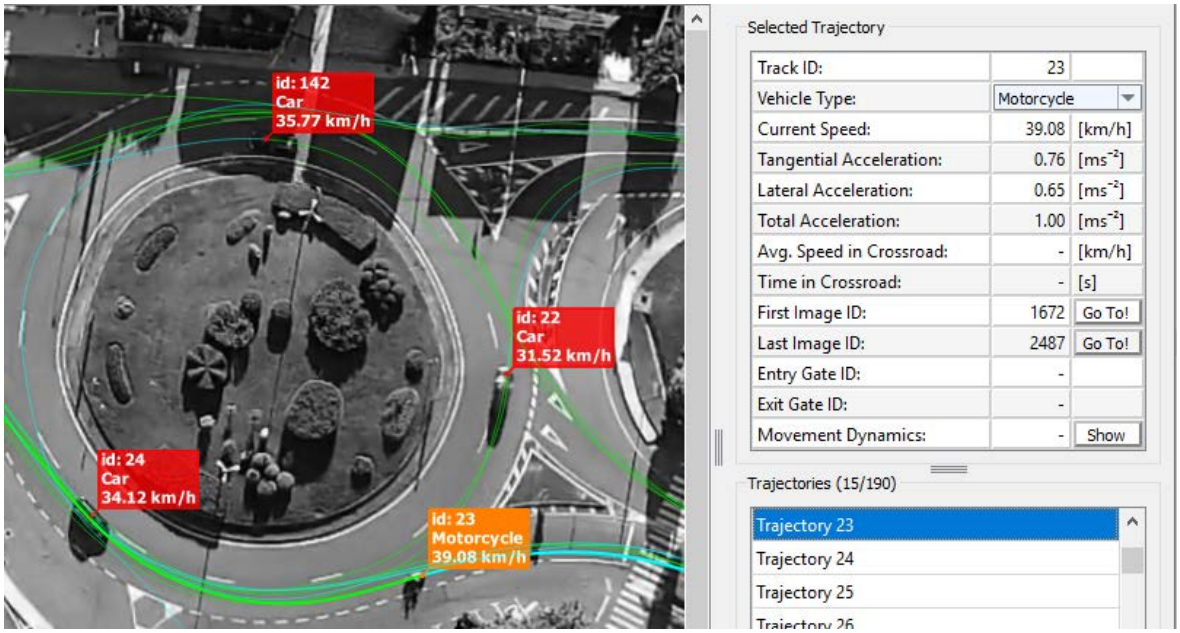

Figure 2.4: Application by DataFromSky showing detected objects within a video together with detailed information.

- **SenseTime**[13] appertains to one of the most funded AI companies in the world with its valuation over $4.5 billion. With such a financial support they have managed to develop highly optimized object detection algorithms used in one of their products called SenseVideo. SenseVideo is supposed to process up to 5 HD video streams by a single GPU in real time. It can recognize and classify pedestrians as well as motor and non-motor vehicles including their attributes, such as license plate and model, and in case of pedestrians also their gender, age and clothing.

  Unfortunately, the SenseVideo system is not available to open public, therefore I cannot obtain that many insights regarding the system features or UI for the inspiration except for a few images of the UI (Figure 2.5). These images support the theory that SenseVideo is purely focused on security use cases when focusing on pedestrian identification from real-time footage as well as re-identification of the pedestrian/vehicle among multiple cameras. Re-identification is a superior feature that would help to solve many problems in traffic analytics. It is, however, not possible to provide high-accuracy results for re-identification of objects even via a human eye. To conclude, although SenseVideo does not provide many insights in terms of features and UI, it shows the future possibilities of artificial intelligence (AI) that a scalable system should be prepared to have a space for.

[13]More about SenseTime <https://www.sensetime.com/intelligentVideo/86>

Figure 2.5: Application by SenseTime showing real-time object detection together with its detected attributes.

- **BriefCam**[14] - Briefcam is a purely software solution for automated traffic data collection as well as providing basic traffic analytics. Similarly to SenseVideo, it allows to review detected objects individually and even play them during the time they appeared on the scene. The user is able to filter the detected objects via object's attributes, such as class or color as well as via space criteria. For instance, the user can filter the objects in a given time range that turned to a given location (Figure 2.6) and summarize the results in basic video insights. Although the object filtering seems to be intuitive and very easy to understand, I know a person that used BriefCam in a real traffic survey and claimed accuracy was more than bad. This fact reminds an important design rule that the system should be first of all functional and only as second usable according to UX principles.

  Despite the rich system functionality, at the beginning the UI displays only an overview and basic options whereas the advanced filtering settings are pleasantly hidden. Therefore, with less options offered the user makes faster decisions as Hick's Law suggests [14]. However, the hidden features make the novice-to-expert transition harder. Discoverability of features is important. Finally, I might take some of the UI principles BriefCam uses into account when designing the UI for Video Insights. However, it is important to still bear in mind the priorities in UI design - functionality first, usability second [15].

In order to design a universal tool that would satisfy the majority of traffic use cases mentioned above, I am going to determine the main scenarios covering the majority of user requirements collected from all three resources and formulate them in the form of system-user interactions. Although not all the use cases will be covered by the scenarios first, it is crucial to have the rest of the use cases in mind for future scalability of the system.

[14]More about Briefcam <https://www.briefcam.com/solutions/review-search/>

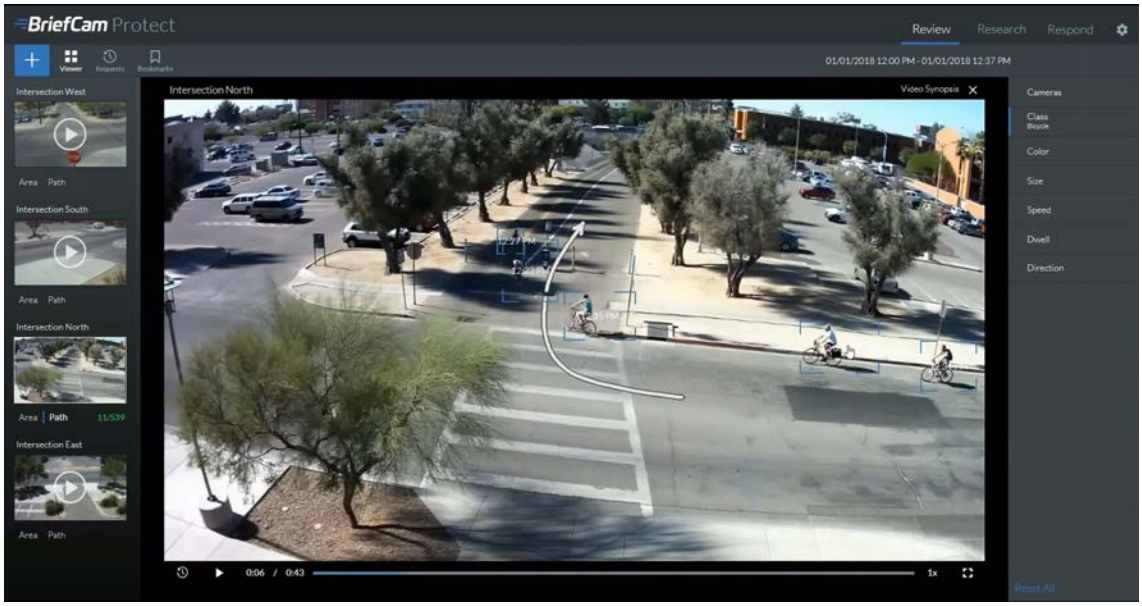

Figure 2.6: Briefcam dashboard showing an aggregated video of objects fulfilling the criteria - bikes that appeared along the drawn path.

2.4 System Specification

In this section, I am going to introduce the main user's scenarios, the object model, and the functional model of the system based on the collected requirements. Besides the textual description of the object and functional model, I enclose the domain model diagram as a visualization of the object model and for non-trivial use cases there are activity diagrams depicting the system-user interaction flow.

2.4.1 Scenarios

This section is dedicated to the formulation of the main end-to-end user scenarios for the system providing spatiotemporal traffic data from camera recording. The scenarios are based on collected requirements from all interviews and from the analysis of alternative solutions. I have determined two main scenarios where the first one is related to the users who are interested in one-time traffic analysis from a video and primarily are interested in an identification of the traffic pattern. Therefore, they do not know in advance what statistics they are looking for. In the second scenario, the users represent subscribed users who repeatedly upload their videos in order to gather specific statistics. These users are interested in the same, i.e. pre-defined traffic patterns and their counts and properties they generate in the long term. Therefore, these two scenarios might differ.

One-time traffic analysis

1. The user uploads a video and waits for the data extraction.
2. The system processes the video and lets the user know after the processing is finished.
3. The user defines the scene (depicts the main traffic flows via zones and lines)
4. The user browses and explores the data by applying filters.
5. The system displays overall information about objects fulfilling user's filters.
6. The user downloads a traffic report with statistics underlying the discovered traffic pattern.

Long-term traffic analysis

1. The user uploads new videos into an already defined camera and waits for the data extraction.
2. The system processes the video and lets the user know when the processing is finished.
3. The user request the same statistics as previously defined, only for the new date time range covering the newly processed videos.
4. The system displays the requested statistics.
5. The user downloads a traffic report.

2.4.2 Object Model

In this section, I present the system entities that I have distinguished during the collection of requirements. Entity relationships are illustrated via the domain model diagram in Figure 2.7. These entities take part in a following functional (use cases) and dynamic model (activity diagram) section.

A **User** entity represents a profile for the actual user. Each user should be able to manage its videos within separate global workspace reflecting the real-life structure. Therefore, **Camera Group** entity represents a folder for virtual **Camera** models representing a folder for individual **Video** files where the user might want to mine the data. During the upload and processing of the video, the video entity might occur in a different video **State**. After automated video processing, the video is represented via a list of **Object** models in time. Moreover, each object is also specified by a class (for instance, car or pedestrian) and a color.

To filter the video data, the user should first draw **Graphical Objects**, such as **Line** or **Zone**, into a camera preview image to define the crossings and zones user is interested in. With the borders specified, user should be able to define various **Filters**, such as **Cross Filter** or **Stay Filter**, and possibly combine them into **Composite filters**. Finally, to let users understand the filtered data in context, the system should provide statistics within a **Widget**. For each widget, there might be a different **Visualization** entity to present the data in the most coherent way. For example, a bar or line graph are the best practice to visualize the number of objects in time. However, representation of the main traffic flows in the scene requires a custom visualization approach on which I will focus more in the next section.

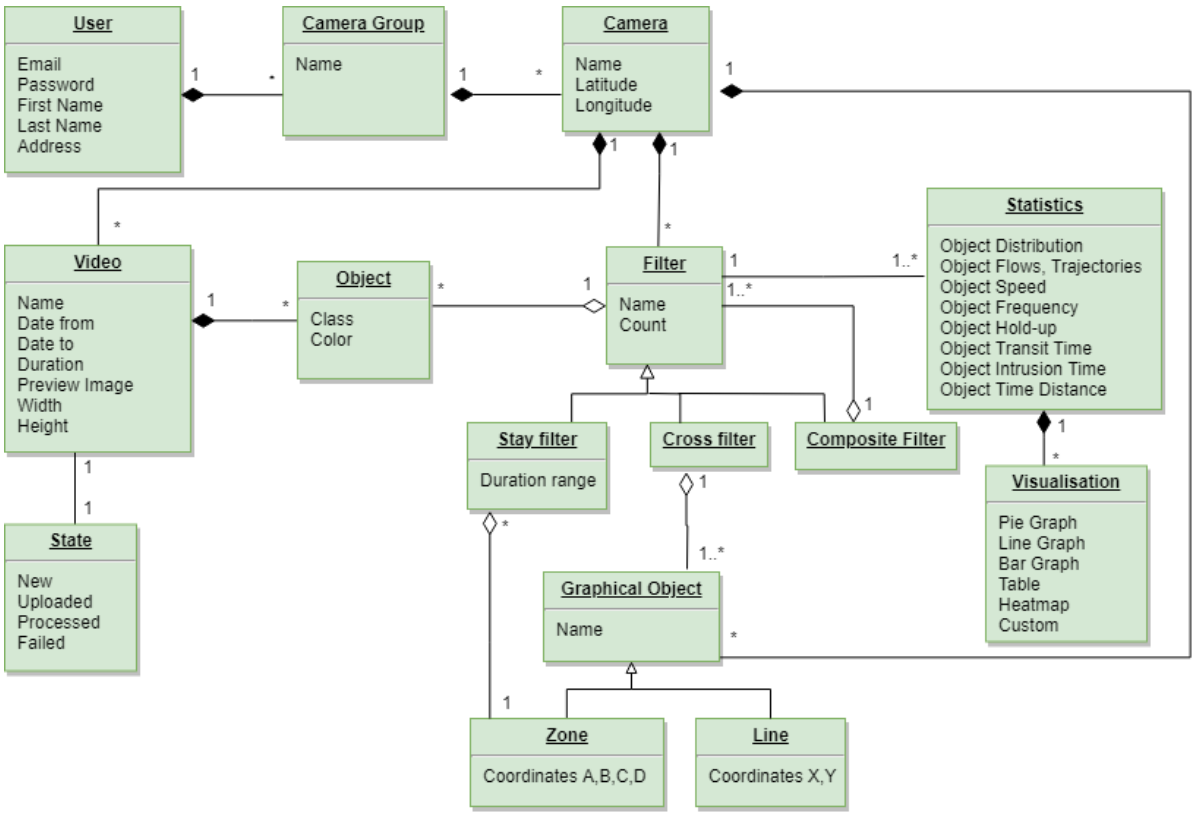

Figure 2.7: Domain model diagram of the redesigned automated traffic data collection and analytics system based on the collected requirements

2.4.3 Functional model

In the paragraphs below I name and describe functional requirements in a form of use cases. All use cases are based on users' requirements and similar solutions as researched above. Each use case consists of a name and a short description including its actors, preconditions and post conditions if necessary.

Video Management

- **Create a camera group** - The user can create a new camera group by entering a name. If confirmation is given, new camera group should appear in the list of camera groups.
- **Edit a camera group** - The user can edit a camera group name. If confirmation is given, an updated camera group should appear in the list of camera groups.
- **Delete a camera group** - The user can delete a camera group if it does not contain any videos and cameras. If confirmation is given, the deleted camera group should disappear from the list of camera groups.
- **Create a camera** - The user can create a camera by entering a name and location. If confirmation is given, new camera should appear in the list of cameras. It is an optional prerequisite that cameras should sit in a similar or a close geographic area.

- **Edit a camera** - The user can edit the camera name or location. If confirmation is given, updated camera should appear in the list of cameras.
- **Delete a camera** - The user can delete the camera if it does not contain any video. If confirmation is given, the deleted camera should disappear from the list of cameras.
- **Sort cameras** - The user can sort cameras by name or by date of last usage in descending or ascending order.
- **Display list of cameras** - The user can display the cameras in two modes. The first one displays camera's preview image. The second one displays cameras in a condensed list with a camera preview image on hover.
- **Display cameras in a geographical context** - The user can display the cameras of a camera group on a map illustrating the cameras in a geographic context.
- **Upload a video** - The user can upload a video into a selected camera by entering the exact date and time the video was taken. Precondition for a successful video upload is that the video file is valid and that it does not overlap the date time range where the parent camera already contains video data. If confirmation for upload is given and the precondition is fulfilled, the new video should appear in the list of videos of the parent camera and its state should change to uploading.
- **Delete a video** - The user can delete the video from the parent camera.
- **Upload a folder of videos** - The user can upload several videos at once into a selected camera. The same preconditions and post conditions are applied to this use case as it is described in the "Upload a video" use case above. Besides that, the videos must produce a continuous footage when arranged one after another according to their date of creation.
- **Display list of videos** - The user can display a list of videos with their current state in the parent camera in both modes of display.

Filtering Management

- **Filter objects by date time range** - The user can define a global filter for camera filters to filter only such objects that appeared in the selected date time range.
- **Filter objects by color** - The user can define a global filter for camera filters to filter only objects comprising the colors the user selects from the given group of colors.
- **Filter objects by class** - The user can define a global filter for camera filters to filter only objects of the class the user selects. The user should be able to select multiple classes, such as car, truck, van, tram, train, motorcycle, bicycle, pedestrian or animal.
- **Draw a line** - The user can create a line on the camera preview to define crossings the user is interested in.
- **Draw a zone** - The user can create a zone on the camera preview to define the zones the user is interested in.
- **Delete a graphical object** - The user can delete any graphical object (line or zone) if it is not included in any filter.

- **Create a cross filter** - The user can create a cross filter to filter only objects that crossed a specific graphical objects in a given order. If confirmation is given, the new cross filter should appear in the list of filters as well its graphical representation on the camera preview.

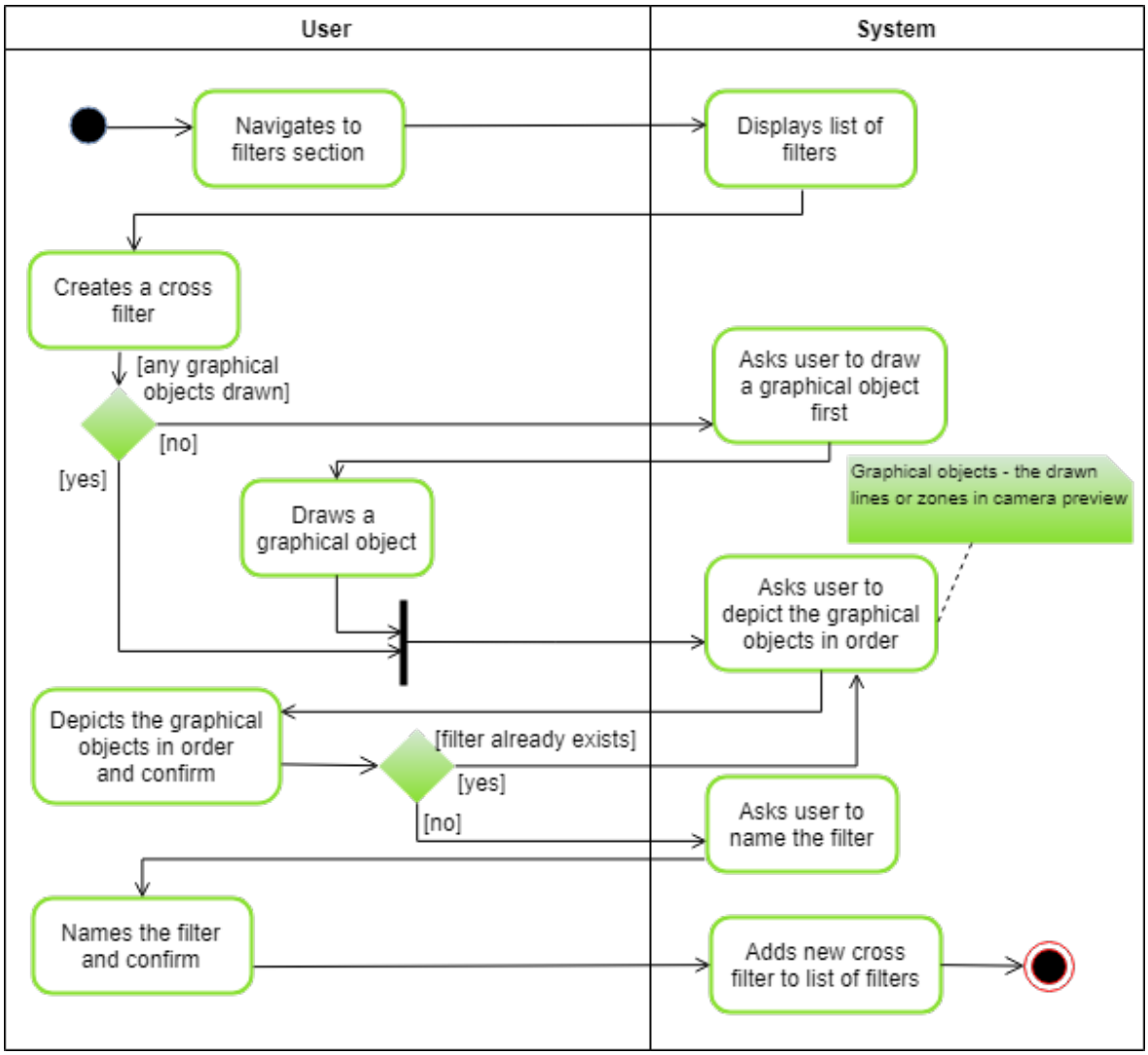

Figure 2.8: Activity diagram of the use case *Create a cross filter*

- **Create a stay filter** - The user can create a stay filter to filter only objects that stayed in one selected zone specific amount of time. If confirmation is given, the new stay filter should appear in the list of filters as well its graphical representation on the camera preview.

- **Create a composite filter** - The user can create a composite filter to filter only objects that fulfilled a sequential combination of cross or stay filters in a selected order. If confirmation is given, the new composite filter should appear in the list of composite filters as well its graphical representation on the camera preview.

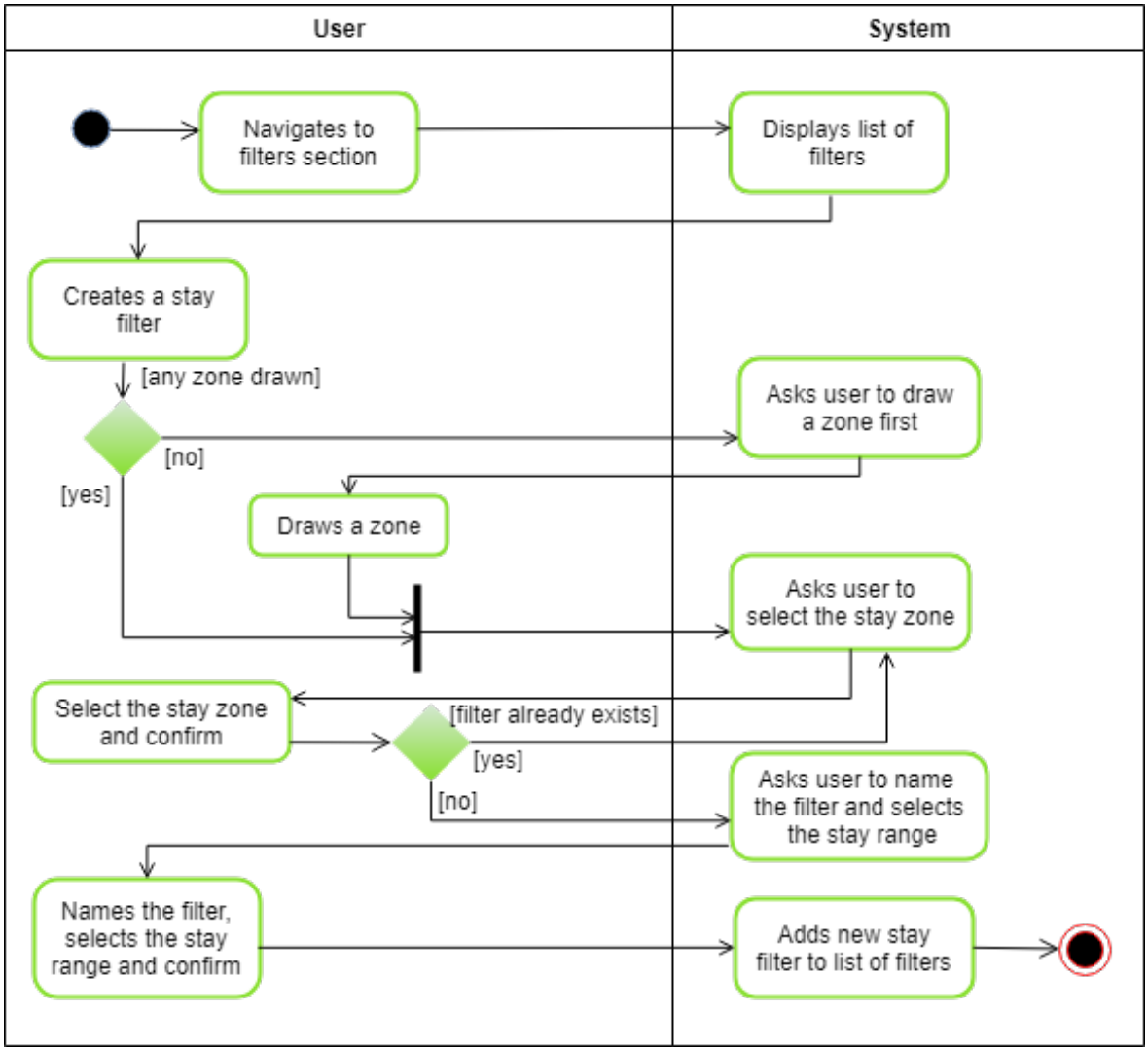

Figure 2.9: Activity diagram of the use case *Create a stay filter*

- **Select a filter** - The user can always select one created filter (cross, stay or composite) to filter objects.
- **Edit a filter** - The user can name a cross, stay or composite filter. The name should be visible to the user.
- **Delete a filter** - The user can delete a composite filter. Cross or stay filters may be deleted if they are not included in any composite filter.

Data Visualization Management

- **Display major traffic flows** - The user can display major traffic flows in the camera preview via appropriate visualization.
- **Display individual object trajectories** - The user can display trajectories of all objects fulfilling one of the filters.

- **Display a frequency of objects** - The user can display the frequency of objects to distinguish the most and the least frequent places in the camera preview.
- **Display a speed of objects** - The user can display the speed of objects to distinguish the places where the objects were moving relatively slowly or fast in the camera preview.
- **Display a hold-up of objects** - The user can display the hold-up of objects to see an average objects hold-up on defined places in the camera preview.
- **Create a configurable traffic report** - The user can configure a traffic report by adding custom-defined statistics the user is interested in.

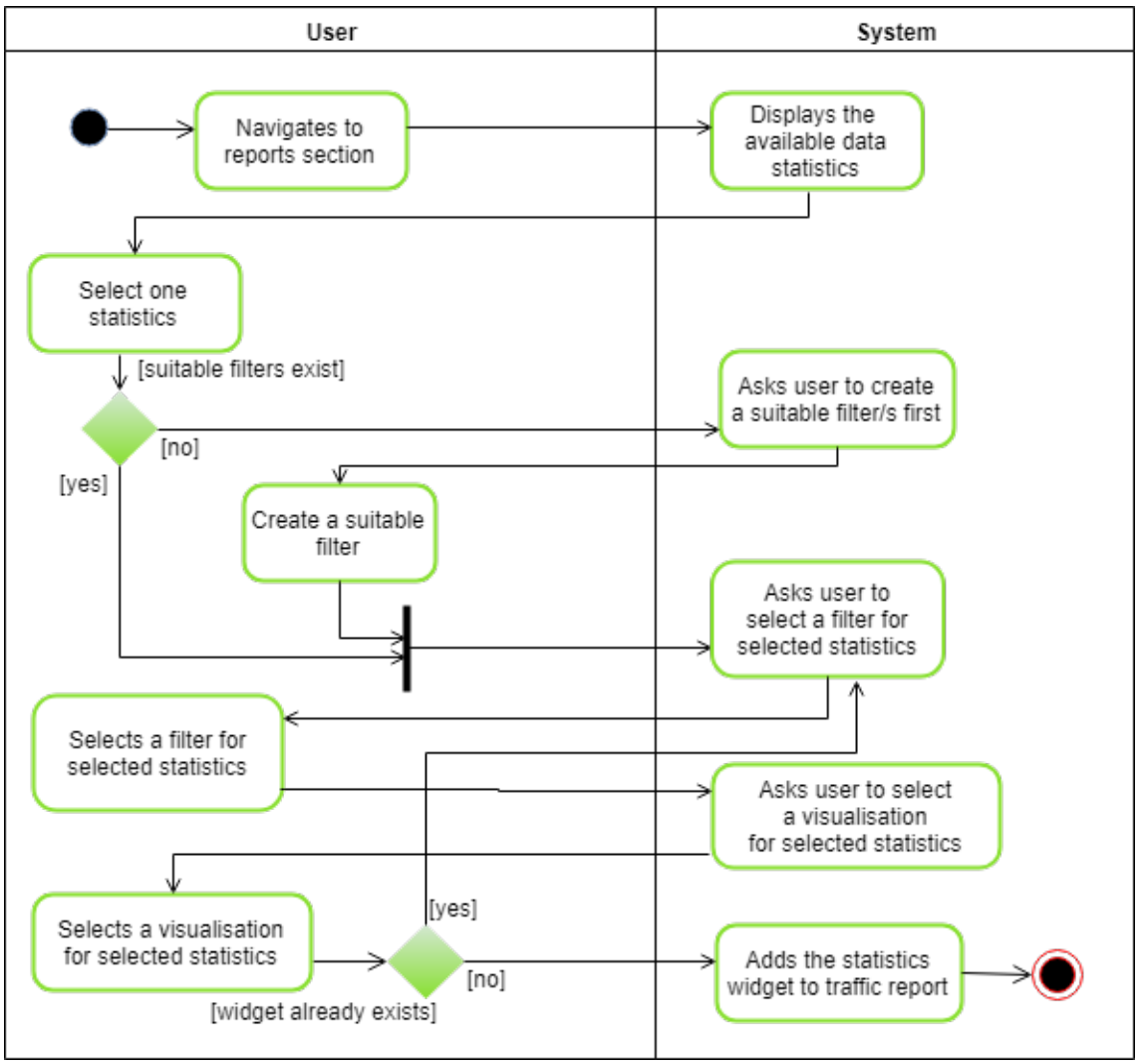

Figure 2.10: Activity diagram of the use case *Create a configurable traffic report*

- **Set data granularity** - The user can set the time granularity globally for all created statistics.
- **Set the report date time range** - The user can set the date time range globally for all created statistics.

- **Display the total number of objects** - The user can add a visualization of the total number of objects fulfilling the selected filter in time to the traffic report.
- **Display peak number of objects** - The user can add information about the peak number of objects and the time range when it occurred to the traffic report.
- **Display the average number of objects** - The user can add a visualization of an average number of objects fulfilling the selected filter in time to the traffic report.
- **Display the object distribution in time** - The user can add to the traffic report a visualization of distribution of objects fulfilling the selected filter in time.
- **Display the average transit time of objects** - User can add to the traffic report a visualization of the average time objects, fulfilling the selected cross filter in time, spend within the cross boundaries.
- **Display the average intrusion number of objects** - The user can add a visualization of the average number of objects intruding a selected graphical object of the selected cross filter in a given time interval to the traffic report.
- **Display the average time distance of objects** - The user can add a visualization of the average time distance of objects fulfilling the selected cross filter in time to the traffic report.
- **Delete statistics** - The user can delete any added statistics from the traffic report.
- **Download a configurable traffic report** - The user can download a condensed version of the configured traffic report as an Excel file.

2.5 Data Visualization

This section is dedicated to the description of the data the system Video Insights is able to extract from the camera recordings. Moreover, I will be working with these data in the upcoming implementation section. Afterwards, I will map the defined use cases connected with visualization to individual tasks [10] where specific tasks can be displayed via well-proven visualization methods. Finally, I will select the most suitable visualization methods to the use cases specified.

2.5.1 Data Definition

It is worth saying that the data of objects in the scene are automatically detected by trained computer vision algorithms as well as trained neural networks covering the field of tracking the objects. These algorithms are trained on manually annotated datasets in a traffic related environment. Therefore, the gathered information about the detected objects is limited by abilities of the algorithms. It can also happen that the algorithm detects false positives (detects an object even though there is none) or false negatives (not detects an object even though there is one). Another issue affecting validity of the data is a possible video distortion due to the perspective of the camera that is installed above the road. This all should be taken into account when working with the data.

Finally, the model Object as described in 2.7 has two categorical attributes **Color** and **Class** and three ordered attributes - ordinal **Speed** and sequential **Trajectory** and **Id**.

- Id (identifier of the object)
- Trajectory (aggregation of coordinates over time)
- Speed (relative unit due to unknown distances at the scene)
- Color (particularly, white, pink, blue, green, yellow, orange, red, gray and black)
- Class (particularly, car, truck, van, bus, motorcycle, bicycle, person or animal)

I included the speed attribute event that the service does not provide directly because, based on the position of the object in time, the relative speed of the object can be derived as well. In the end, the trajectory is composed of multiple segments. Each segment is composed of two points containing coordinates from the scene (the first one being the start, the second one the finish) and the time the object appeared in the specific place in the video. As the segments in a trajectory are connected, the trajectory represents continuous spatio-temportal data.

2.5.2 Task Mapping

Task mapping helps to find the suitable visualization methods for the use cases mentioned above in the *Functional model* [10]. Thanks to the data definition above, it is possible now to formulate some visualization related use cases as tasks identifying **Action** and **Target**. Action types are based on "Why" data abstraction (See Figure 2.11).

- Display individual object trajectories - **Action**: Identify, **Target**: Data items
- Display the frequency of objects - **Action**: Locate, explore, **Target**: Distribution, correlation
- Display the speed of objects - **Action**: Locate, explore, **Target**: Distribution
- Display the hold-up of objects - **Action**: Locate, explore, **Target**: Distribution
- Display the total number of objects - **Action**: Summarize, **Target**: Distribution, correlation with time
- Display the peak number of objects - **Action**: Summarize, identify, **Target**: Correlation with time
- Display the average number of objects - **Action**: Summarize, **Target**: Correlation with time
- Display the object distribution in time - **Action**: Summarize, **Target**: Distribution, correlation with time
- Display the average transit time of objects - **Action**: Discover, **Target**: Distribution, correlation with time
- Display the average intrusion number of objects - **Action**: Discover, **Target**: Distribution, correlation with time
- Display the average time distance of objects - **Action**: Discover, **Target**: Distribution, correlation with time

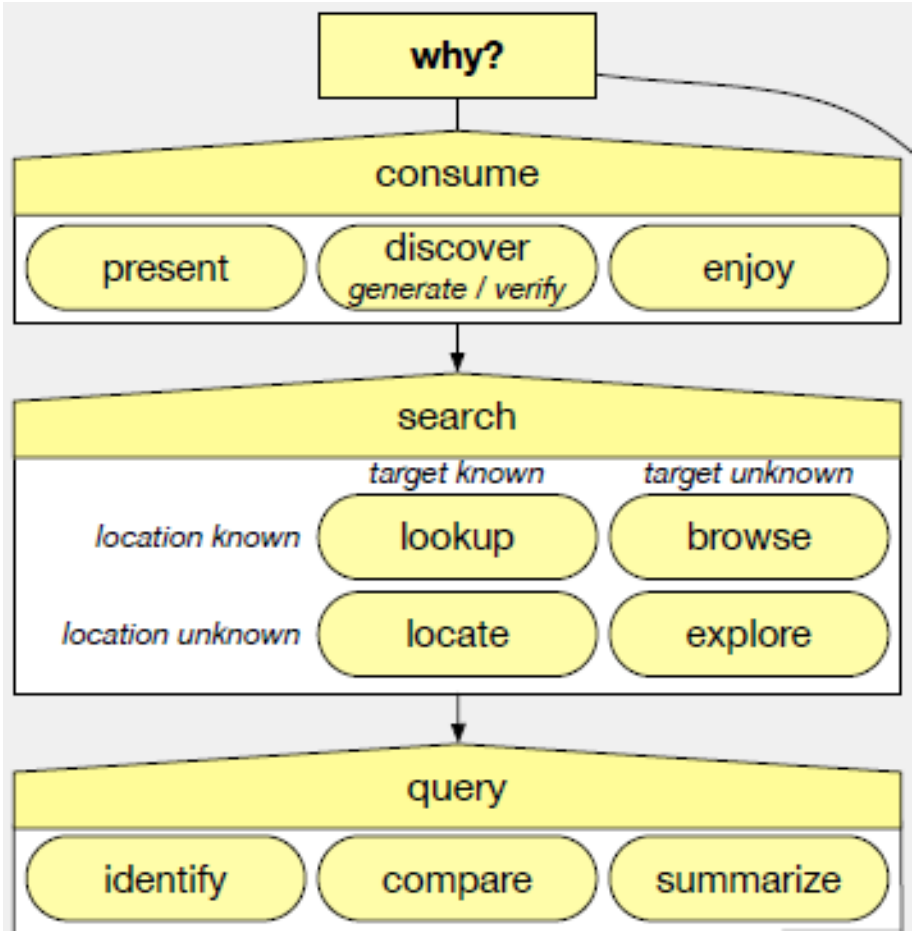

Figure 2.11: The part "Why" of why-what-how analysis framework [10]

2.5.3 Task Visualization

The tasks have been mapped and now it is apparent that the majority of tasks are distribution focused. Even though, distribution does not offer only one possible visualization method. For instance, the distribution of object frequencies or object hold-ups must be seen in context - in a camera preview. On the other hand, some more abstract statistics, such as average intrusion number of objects or average time distance of objects, might be too difficult to understand in the context of camera preview. Therefore, I enclose a list of common distribution visualization methods for the tasks defined above that I can incorporate into the final design in the next chapter.

- Heatmap - linear color range or one color with various levels of saturation
- Graph - line, bar, pie
- Table

2.5.4 Visualization Challenges

This part describes some visualization tasks the Video Insights system has currently problems with, such as the problem of displaying the object trajectories for long time ranges where the total amount of data might cross some threshold over which the displayed trajectories do not have any added value, because the user cannot recognize the main traffic flows anymore, as displayed in Figure 2.12.

One of the possible solutions is to hide or disable the option to show trajectories if the video is longer than a given threshold. However, this option is against the 'User control and freedom' Nielsen heuristic. The application should minimize disabling functions for the user. The users pay for certain functionality and this is why all functions should remain available to them. Another solution is to dynamically cluster the trajectories with the same coordinates and dynamically set high opacity to trajectories with a high recurrence and a low opacity to trajectories with a few occurrences.

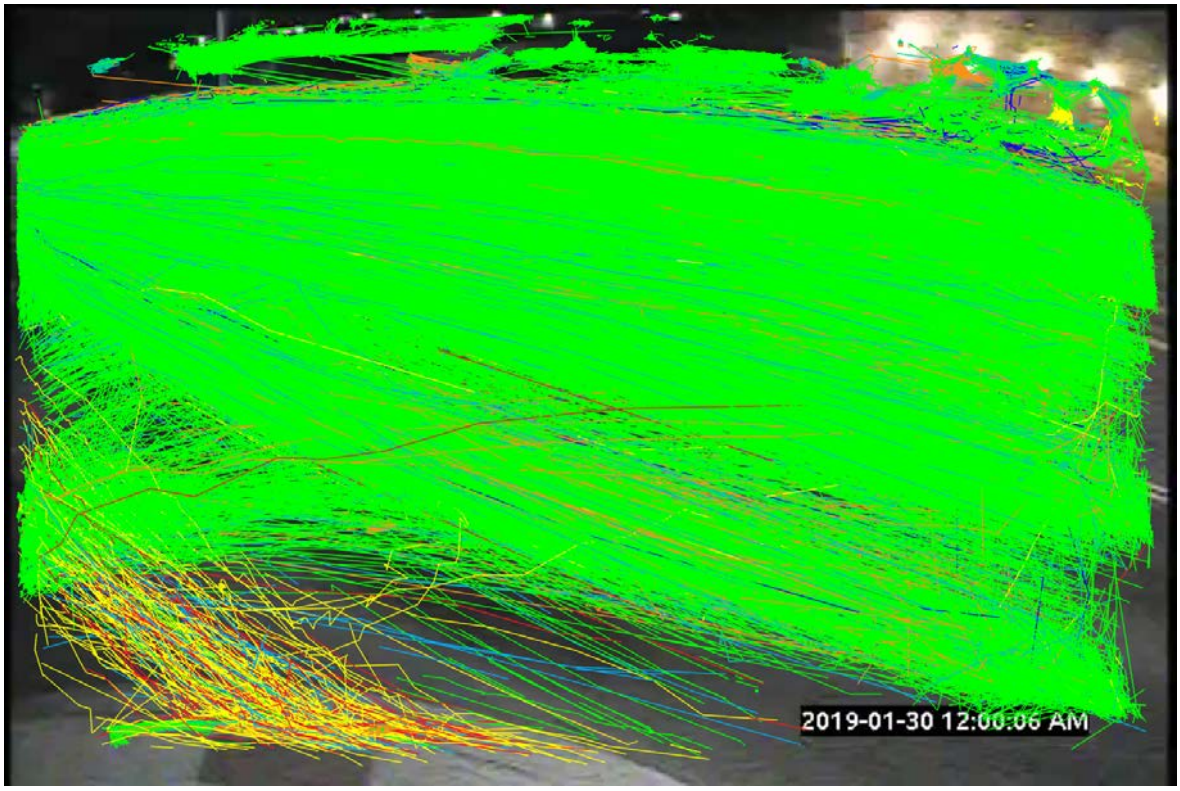

Figure 2.12: A screenshot from the Video Insights displaying trajectories for a 48-hours-long footage from a camera installed in a low height.

Similarly to that, we might imagine a situation where a new user might not understand the principle of the service - to upload a static video from the surveillance cameras - and the user uploads a video from his car camera. The user might argue that the camera is statically installed, however if the surrounding is moving, all the detected objects compose trajectories that do not have a sense. Figure 2.13 depicts the possible result if the user is not well educated before the video upload.

Another challenge following the first one is that some request responses might have the size exceeding tens of MB if the user requests trajectories for videos longer than 24 hours. In

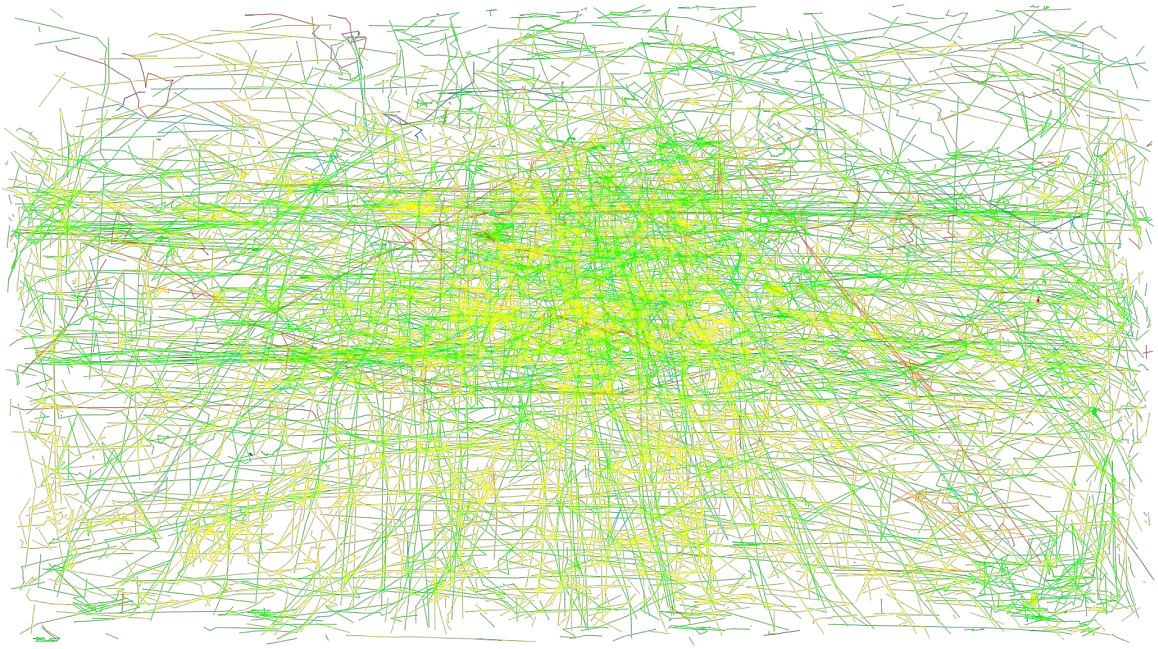

Figure 2.13: Exported trajetories from a scene with a not-statically installed camera.

comparison to human surveyors and their results, which we would normally obtain in months, the user in the responsive web era is used to getting results, statistics or images within a few seconds. Therefore, it is going to be a design challenge to display the 'loading' system status in as much pleasant and intelligent way to avoid annoying the user during the waiting.

It could also be a theoretical challenge if the user requires a selection of a single trajectory in the future. For instance, the user might want to investigate a suspicious object and revise it - a pedestrian on the highway and similarly. Today, it would be impossible to locate and select the specific trajectory unless it has a different object class.

Next issues come with heatmaps. Figure 2.14 illustrates an example of a non-linear heatmap that should highlight very frequented places in red, normally frequented places in green and less frequented places with a blue color. However, to distinguish the two extremes, i.e. the low and high frequency locations, might not be clear at all.

Analogous to speed heatmaps, the users want to recognize the places with fluent traffic and with slow or even jammed traffic. Due to the possibly distorted data of the speed (when not recording a video from the bird's-eye view), the task to visualize the speed correctly in the context of the camera scene becomes even more challenging, for example when vehicles on the horizon might appear as slow but they are just too far away and vice versa. To eliminate this issue, it is necessary to know many parameters of the scene to 'skew the scene to the top view' and incorporate information about the depth into the heatmap.

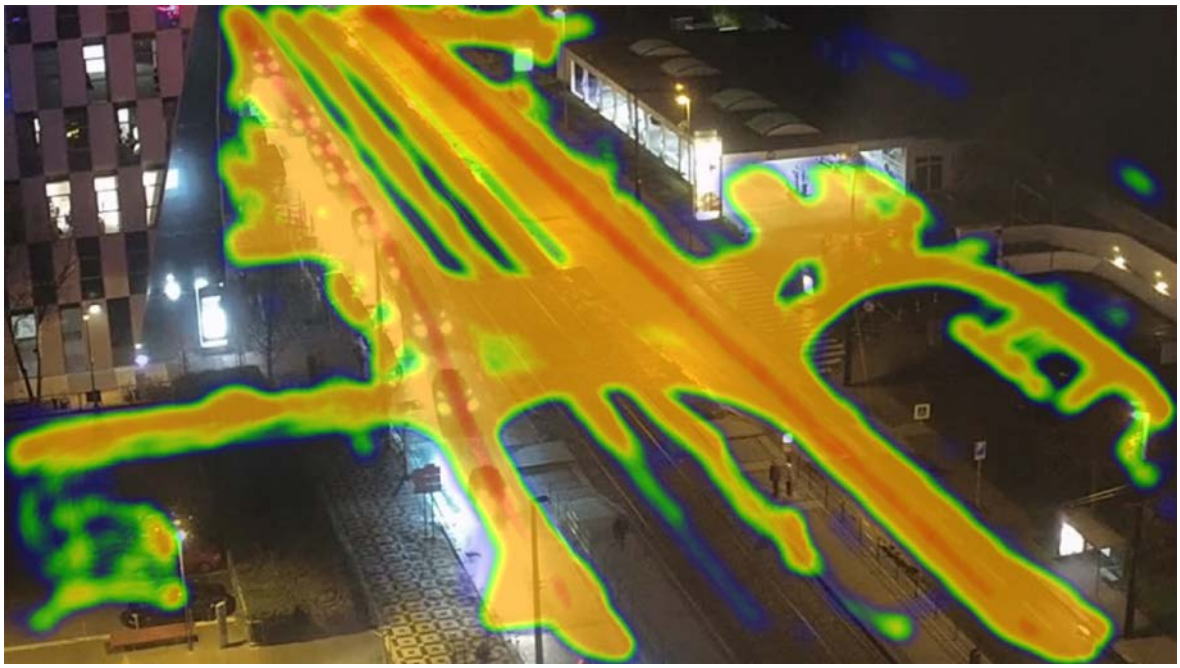

Figure 2.14: A screenshot from the Video Insights displaying a frequency heatmap with a hard-coded upper value and with a non-linear rainbow gradient.

Chapter 3

Architecture and Design

In the initial part of this chapter, I am going to describe the entire GoodVision Video Insights service infrastructure in order to understand how the camera footage is transformed into the traffic data I use later on within the user interface implementation. More specifically, I will explain the data flow among the major components of the service pipeline. Then, a brief specification of each major component will follow. Particularly, what each component accept as an input, how the component transforms the data, what is the component's output and what technologies the component use and why. Furthermore, I will specify the client-server communication between the API node and the web application - Video Insights.

In the second part of this chapter, I am going to present the user interface development stages (including evaluation) based on the requirements collected and formulated in the previous chapter. Firstly, I will describe the low-fidelity prototypes for the non-trivial use cases discussed in the analysis chapter. Then, I evaluate the prototypes via the heuristic evaluation to converge to the best possible UI in the high-fidelity prototyping stage. The high-fidelity prototypes will then follow and will be used for the final implementation of the user interface.

3.1 Service Architecture

A simplified deployment diagram of the GoodVision Video Insights architecture is depicted in Figure 3.1. As mentioned before, the whole solution is cloud based, specifically all major parts are deployed on Amazon Web Services (AWS). Major components are a detector and tracker component responsible for parallel extraction of all data from the video, the SQL database server storing all the data extracted from the video as well as data about the users. The next component is the AWS CloudFront mediating the communication between the user and application programming interface (API) server as well as serving the Video Insights web application I will create the frontend for.

The entire solution is scalable by design due to modularity and high emphasis on automation within the video processing pipeline as well as due to CloudFront component that scales the application according to the load (of requests).

3.1.1 Detector and Tracker Component

The detector component is responsible for the automated object detection, including the object classification, whereas the tracker component's goal is to track the detected objects

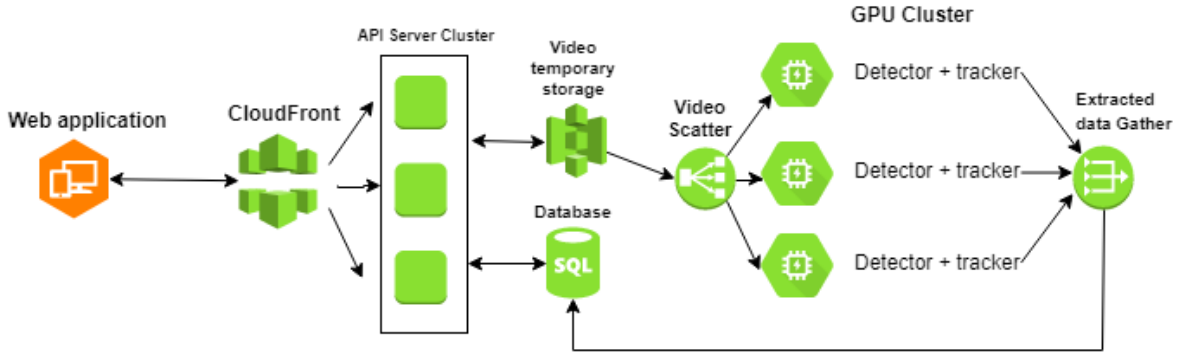

Figure 3.1: Video Insights simplified diagram of the cloud infrastructure displaying all the major components and their relationships. The orange component represents the web application I will create the frontend for.

between individual video frames. There are trained neural network models accelerated via CUDA framework on the given GPU behind both components.

To ensure as high accuracy as possible, these neural networks are continuously trained via annotated data from public datasets as well as from own annotated videos provided by GoodVision. In case of object detection, these annotated data have the form of a picture with rectangles depicting objects on the scene together with its class. In case of object tracking, the annotated data have the form of a video where the annotator has to key the object on the scene in time. A further description of the models used in these neural networks is proprietary know-how of GoodVision.

The typical flow then looks as follows. After a successful video upload by the user the video is stored in a temporary AWS S3 storage. The video is then cut into a number of chunks and based on that, the same number of GPU instances where the detector-tracker component is deployed on each is started. Then, each detector-tracker component accepts in parallel a raw video chunk as an input, processes it frame by frame and produces extracted data for the given video chunk in a form of .prototxt file as an output. The Protocol Buffers format is chosen due to its properties - as Google claims, Protocol Buffers produce a smaller, faster and simpler format in comparison to XML [9].

In the next step of the pipeline, the generated prototxt files from all instances are stitched together and saved into the SQL database appropriately. The advantage is that after the video processing, the service does not need to store the original video footage anymore. Therefore, in the next step of the pipeline, the video is deleted from the temporary S3 storage, which enormously saves the space, costs and naturally expenses.

To conclude, the parallel video processing accelerates the whole pipeline dramatically so the data extracted from videos can be delivered within 1 hour approximately, regardless of the video length. This fact is useful to have in mind when designing the UI for traffic data visualization in Video Insights. Also, it is valuable to know the limits of the AI for the future scalable UI design. For instance, if we gather enough training data for recognition of the car models or new object classes (middle truck, heavy truck, boat, adult or child) and if users upload the compatible videos to the system, the detector-tracker component classification might be one day easily extended with these features (as the SenseVideo, mentioned in the *Alternative Solutions* section) and the UI should be ready for that.

3.1.2 Database

As mentioned above, the storage for both traffic data as well as the user management is ensured by SQL database which is deployed on database server on AWS. The database (DB) type and used technology is proprietary know-how of GoodVision. However, for the future high availability demands, I imagine the database might be a weak point of the solution due to the high volume of the data saved in one place. Division of the database into two might be one of the optimization solutions, where the first one would store only the user data and the second one would be optimized for querying of the spatiotemporal data about the objects detected in the scene.

Besides that, to ensure high availability of the data, the DB is replicated on two more backup DB servers. This is why the consistency of these redundant resources is ensured by synchronization via a traditional master slave replication model.

The data from the database is accessed (read and write) exclusively via the application programming interface described further in the next section.

3.1.3 Application Programming Interface

The next essential component is the application programming interface (API) that is deployed on its own server and is the only medium when accessing the data from the DB. The API implements the principles of the Representational State Transfer (REST) architectural style [17] to provide a stateless data transfer. As an input the API receives the requests from the CloudFront[1] content delivery network (CDN) service which originally come from the web application.

This CDN allows load balancing of incoming requests from the web application so that one API instance cannot be overwhelmed by a high number of requests that it would otherwise not be able to serve. Besides that, it adds an extra layer of security against possible vulnerabilities as well as it guarantees to return the data from the API with constant high speed regardless of the request's origin or location thanks to the CloudFront distributed network.

Majority of the API endpoints is secured by design, which means the user has to authenticate first before requesting e.g. traffic data or any user-sensitive data. After the user authentication, each request sends a bearer token identifying the concrete user in its header.

After the request is read, the API performs one or several ACID transactions over the DB. Based on the request type (CRUD) it queries appropriate data, transforms it accordingly and sends the response with data in the JSON format within the response body if needed. It is important to note that some requests related to traffic data filtering (e.g. over the long period of time) might take a while ($>$ 30s) and therefore the UI should display a corresponding loading indicator or other means for indicating visibility of the system status.

As mentioned above in the in the Analysis chapter, this API might be open to public in the future. This step would preserve all the functionality GoodVision Video Insights provides except the frontend side. On the other hand the user with access to this API would have a possibility to develop its own frontend implementation if he/she needs a special functionality, proprietary look or only very limited functionality. For example, the user might need only the upload endpoint to upload a continuous stream of video (or batches of video) without the need to physically access the Video Insights frontend and click there on "upload a video" button.

[1] More about - <https://aws.amazon.com/cloudfront/>

As a result, the client might experience a pseudo-online video processing in the future (there would be approximately 1-hour delay if the video stream would be processed in 1-hour long batches). This feature would require to incorporate a one-time configuration page or a window into the Video Insights design to link the camera stream directly to the web application as well to design accompanying error prevention and handling.

3.1.4 Web Application

The last major component is the client web application, the only medium the end user is in touch with. In case the whole service pipeline works flawlessly but the web application does not function according to user expectations, the whole user experience from the application might be ruined and moreover, the chance of loosing the client increases.

Inversely, if some of the components from the pipeline do not work properly, the web application is able to detect it and respond accordingly and the user does not need to even notice the misbehavior of the component. For instance, if the API server is under maintenance, the web application can detect it (by receiving a 5xx HTTP status code) and display the competent page with explication.

Therefore, this component belongs to one of the most essential components in the entire service architecture when focusing on growing and maintaining of the product audience [8]. This is also the only component I will be in touch with during the implementation stage. Below, I will describe in short the web application background, architecture and how this component communicates with the rest of the system.

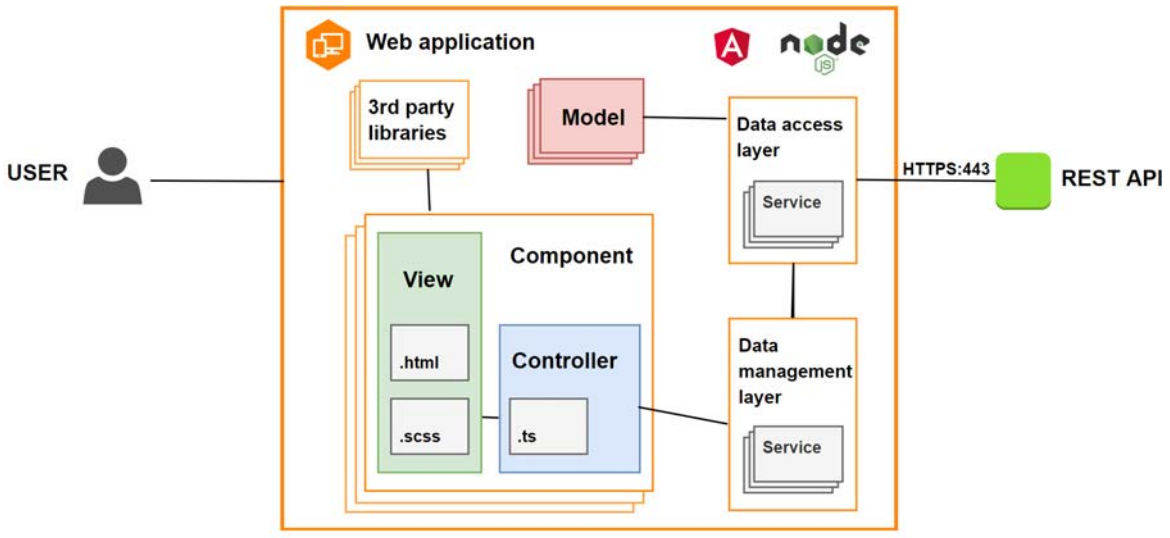

Figure 3.2: Component diagram of the Video Insights web application with highlighted Model (red), View (green) and Controller (blue) of the MVC architectural pattern.

Architecture

Similarly as the API, also the web application is deployed on the AWS while the CloudFront service assures the availability of the application according to the load. As an input the

application receives interactions from the user. These interactions require in most cases data of various kinds (user management, traffic data, etc.). Therefore, these data are accessed via HTTP requests to API as it was already described in the API component section. When the response from the API is accepted, the application parses the response body and uses or displays the data accordingly in the UI of the web application to the user.

Regarding the technologies, the web application is developed in Angular [2] Javascript framework. As the whole service architecture, the web application also follows the good design principles of the low coupling and high cohesion [4] as seen in Figure 3.2 depicting the component diagram. The component diagram describes the inner architecture of the web application and implementation of the MVC architectural pattern Angular has support for. Besides that, high emphasis is put on building a modular application during the whole design and development phase. For instance, each layout component (side navigation bar or toolbar) and each functionality component (user management, traffic data visualization) are separated into independent components. Each component comprises of own HTML file, SCSS file and Typescript file. The HTML file is responsible for visual description of the UI as well as for two-way binding and displaying of appropriate data. The SCSS format is an extension of the traditional Cascading Style Sheets (CSS) responsible for styling the HTML elements and building the layout. Further information about technologies used in the application is discussed in the next chapter *Implementation*. To conclude, the dynamic modular design and component reusage, supported by Angular, might accelerate the frontend development as well as readability of the code for future development.

3.2 User Interface

In the last section of this chapter I am going to present the low-fidelity prototypes for two non-trivial use cases - Create a cross filter and Create a configurable traffic report. I sketched two approaches for each use case where both will lead to the same result only via different components or interactions. Next, I will evaluate these via heurisitic evaluation, and based on the results, I will create high-fidelity prototypes for the same use cases and for more.

3.2.1 Low-fidelity Prototypes

Create a cross filter

Both prototypes of "Create a cross filter" fulfill the system-user flow as defined in the associated activity diagram. Prototype A (see Figure 3.3) illustrates step by step (in orange circles) creation of the new cross filter that forms a direction via drawing a line with the mouse across objects in the desired direction. This method is very authentic in cases when the users use a tablet so that they can depict the movement naturally via a touch gesture.

On the other hand, prototype B illustrates how the direction can be defined via clicking on zones in the sequence as the vehicles cross them. The time needed to accomplish the task might vary as Fitt's Law [14] suggests. Therefore, prototype B might not be suitable for the elderly and people who have a problem with mouse manipulation. To alleviate this downside, I did not register any possible elderly users during the analysis of the target audience, therefore this fact is not fully relevant.

[2] More about Angular - <https://angular.io/>

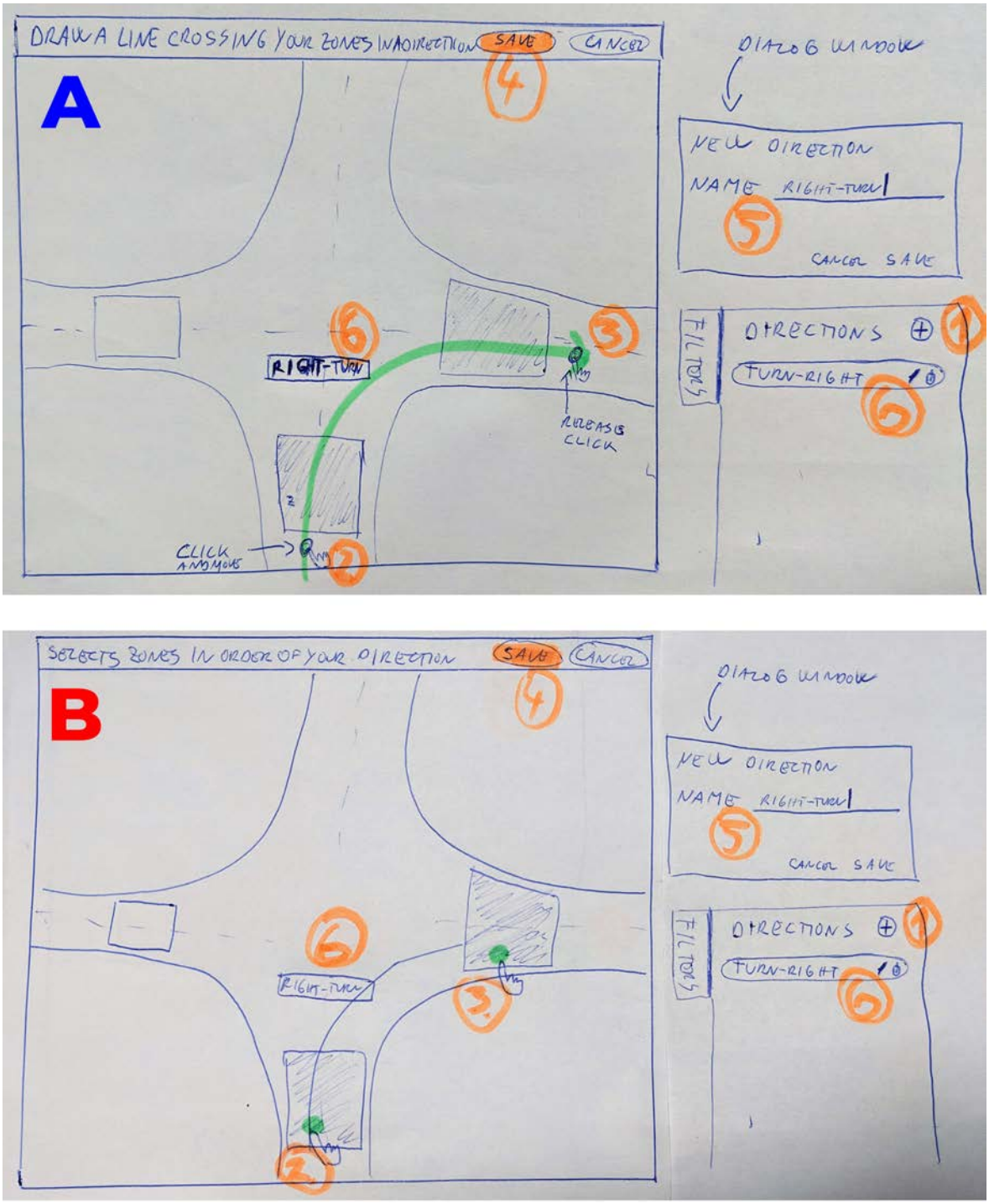

Figure 3.3: Create a cross filter via drawing a direction is visible in prototype A whereas selection of the zones in a sequence via clicking on zones in the sequence is depicted in prototype B.

Create a configurable traffic report

Both prototypes of "Create a configurable traffic report" fulfill the system-user flow as defined in the associated activity diagram. Prototype A (see Figure 3.4) illustrates step by step (in orange circles) addition of the new statistics widget into the traffic report via a dialog window where the user has to customize the statistics in detail, including the desired visualization.

While in prototype B, the user can add a new statistics for a selected direction with default settings, including the visualization type and time granularity. In case the users want to edit some of the properties of the statistics widget, they can do so by clicking on the vertical hamburger menu associated with every statistics widget in the upper right corner. There, the user can select an option to edit the widget. Then, the same dialog window presented in prototype A would open and the user would eventually change the settings of the statistics widget.

3.2.2 Heuristic Evaluation

Before designing the high-fidelity prototypes, I will first evaluate the low-fidelity prototypes of each use case via Nielsen's heuristics [11]. The 'create a cross filter' use case is denoted in Table 3.2.2 below as 'Use case 1' and the use case 'Create a configurable traffic report' is denoted as 'Use case 2'. After evaluation, I should recognize possible weaknesses of each design and therefore be able to choose the more appropriate one for each use case. Besides that, I will know which heuristics need to be improved to satisfy them all in the further high-fidelity prototyping phase.

Nielsen Heuristic	Use case 1		Use case 2	
	A	B	A	B
Visibility of system status	Yes	Yes	Yes	Yes
Match between system and the real world	Yes	Yes	No	No
User control and freedom	Yes	Yes	Yes	Yes
Consistency and standards	Yes	Yes	Yes	Yes
Error prevention	Yes	Yes	Yes	Yes
Recognition rather than recall	Yes	Yes	No	Yes
Flexibility and efficiency of use	Yes	Yes	Yes	Yes
Aesthetic and minimalist designs	No	Yes	No	Yes
Help users recognize, diagnose, and recover from errors	Yes	Yes	Yes	Yes
Help and documentation	Yes	Yes	Yes	Yes

Table 3.2.2: Results of the heuristic evaluation for the use case 1 - 'Create a cross filter' and the use case 2 - 'Create a configurable traffic report'.

Heuristic evaluation - Results

As the evaluation table above suggests, the more suitable UI for the use case 'Create a cross filter' is prototype B supporting the selection of graphical objects via mouse drawing. Similarly, the more suitable UI for the use case 'Create a configurable traffic report' is prototype B that enables creation of statistics widgets via the right pane instead of a dialog window. However, this prototype does not fulfill the heuristic 'Match between system and the real world' that should be enhanced in the high-fidelity prototype in the next section.

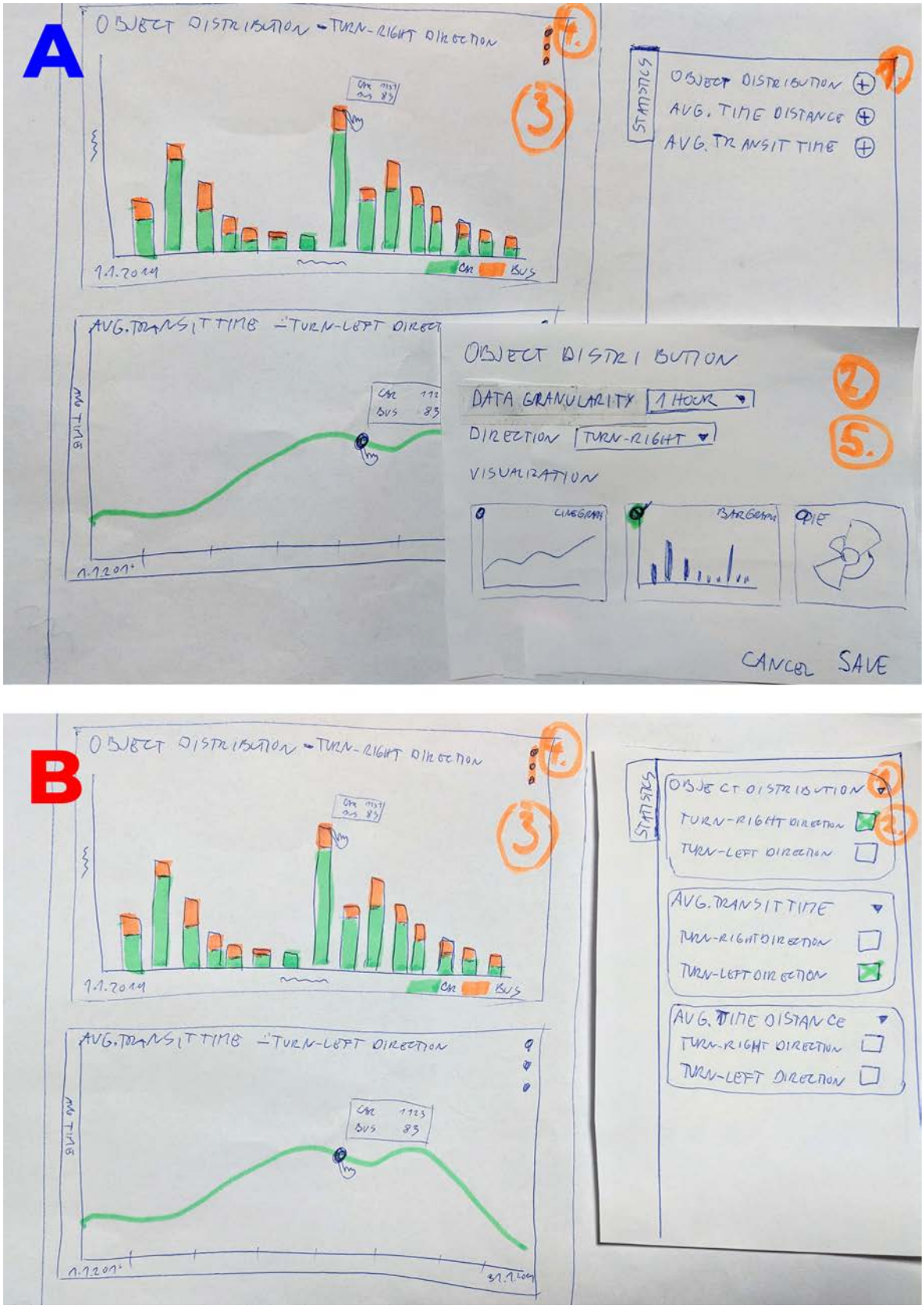

Figure 3.4: 'Create a traffic report' use case by adding individual custom-defined statistics via a dialog window is depicted in prototype A whereas creation of a statistics widget via the right pane is depicted in prototype B.

3.2.3 High-fidelity Prototypes

In the next UI development iteration according to the guidelines [2], I created high-fidelity prototypes that cover the two non-trivial use cases already researched during the low-fidelity prototyping stage. This time they were designed with a higher focus on details. Besides that, I include the high-fidelity prototypes of the use case 'Create a composite filter' that did not need to go though the low-fidelity prototyping stage and further evaluation due to a clear design. The tools used for the high-fidelity prototypes were Adobe XD and Photoshop from the Adobe Creative Cloud suit that I have a subscription for. Besides that, I followed the Angular Material design guidelines to create a minimalist but efficient UI that can be easily integrable into the existing frontend of the Video Insights solution.

Create a cross filter

After the heuristic evaluation, I designed the idea of creation of the cross filter by drawing, as seen in Figure 3.5. In the first step, the user has to click on the 'New' button to invoke the 'draw direction' state. Subsequently, the user should draw the direction crossing graphical objects (lines or zones) in a specific order by using a mouse. To achieve the best user experience, the direction composed from arrows should be moving in the same direction as it was drawn so that the flow is visible for the user via two channels - the arrow's angle and the animation of arrows in that angle. When the user finishes, the user should click on 'Save' and a dialog window with further details about the filter should pop up. I added an extra feature so that the user can reverse the direction by clicking on the button with swap icon. This is useful if the 'duplicate filter' feature is supported in the future. If the user draws a direction that crosses only one graphical object, it is highlighted to the user and all vehicles that ever crossed that graphical object in any direction will be filtered. The last step is to click on 'Save' and then the new filter is represented via a green rectangle in the right tab together with the text description of itself.

Figure 3.5: A cross filter high-fidelity prototype displaying a step by step (see numbers) events when creating a cross filter by drawing.

Create a configurable traffic report

The second non-trivial use case I transformed from the low-fidelity into high-fidelity prototype is the 'Create a configurable traffic report' use case. Figure 3.6 illustrates the page containing all statistics widgets for a given camera group. Every widget can have a various width (dependent on its type and selected visualization). Moreover, to distinguish the statistics of individual cameras, I added a colorful border on the left and the legend into the toolbar enabling to filter the statistics by camera, too. In the upper right corner of every widget, a hidden menu that consists of not that often used actions, such as edit, delete or export, should be available. The grid with widgets should support sorting via drag drop.

To create a widget for the specific filter, the user has to be present on the widgets page and click on the 'new widget' button. However, to fulfill the only heuristic the low-fidelity prototype did not satisfy, i.e. to 'match between system and the real world' due to an absent camera preview on the widgets page where the user cannot see the filter in a context, I added a shortcut to every filter in the form of a button (see Figure 3.7) that opens the 'new widget' pane (see Figure 3.8) on the same page as the camera scene is. It should pre-fill the new widget pane with the information about the filter and when the user selects a statistics and clicks on 'Save', it should redirect the user to the widgets page containing the newly created widget.

Figure 3.6: A create a configurable traffic report high-fidelity prototype displaying a page with statistics widgets of all kinds.

To describe Figure 3.8 more in detail, there are two steps when creating a new widget. Firstly, the user has to select a camera and then appropriate filters should be loaded. When the user selects a filter, the system should be redirected on the second step where the user chooses what information the user is interested in together with the date time range, time granularity, if needed, and the type of visualization of the statistics. Naming of the widget should be optional.

Figure 3.7: The red arrows depict two ways of creating the widget. The arrow on the left points to the tab that navigates to 'Analyze page' where the user can create a widget. The arrow on the right points to a button (attached to each filter) that invokes creation of the new widget with the given pre-selected filter.

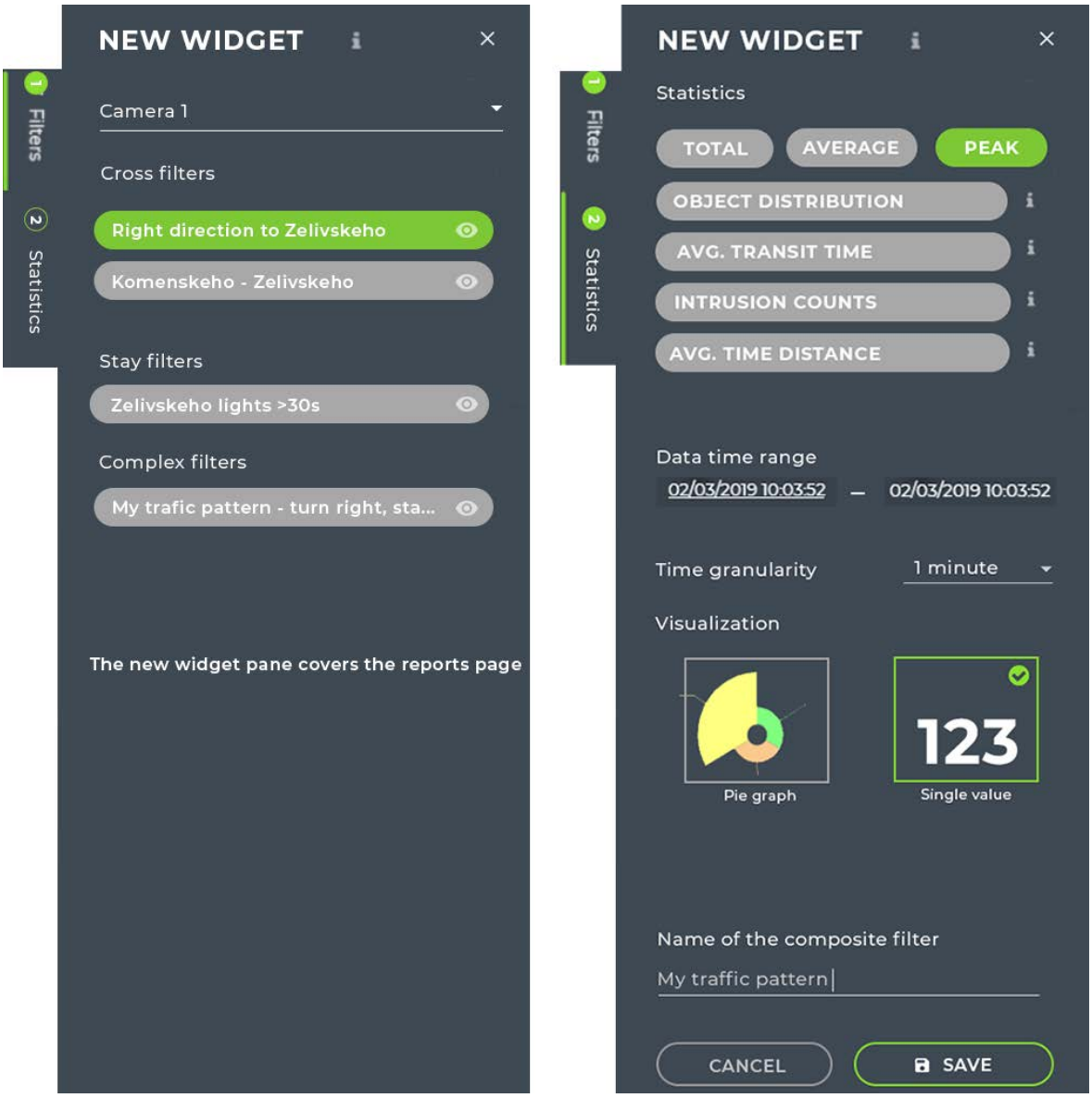

Figure 3.8: A high-fidelity prototype of the right pane where the user can create a new widget in two simple steps.

Create a composite filter

The last high-fidelity prototype is focused on the creation of the composite filters. This feature was described only briefly in the requirements specification and it should allow the user to combine the already created cross or stay filters (filtering vehicle trajectories) via logical operators. Based on the requirements, I included three binary logical operators - **AND, OR** and ,**THEN** that can be used when combining filters filtering the object trajectories in the design. However, to understand the whole 'grammar' when handling the filters, I introduce the definitions and syntax below.

G = {Z, L}	Zone (Z) or line(L) are defined as graphical objects G
I $\in N^+$	Interval (I) defines the duration of stay within Stay(Z,I)
Cross(G)	Cross filter filters trajectories that intersects G
Stay(Z,I)	Stay filter filters trajectories that intersects Z and stay in Z for a specific amount of time within the range of the interval I
O = {AND, OR, THEN}	Binary operator O evaluates two operands
x = {Cross(G), Stay(Z, I), x **O** x}	Operand x can have three different definitions
x **AND** x	Expression for filtering trajectories fulfilling both operands x
x **OR** x	Expression for filtering trajectories fulfilling at least one of the operands x
x **THEN** x	Expression for filtering trajectories fulfilling the operands sequentially (meant in time)

Regarding the design fulfilling the specified grammar, I found an inspiration in the block programming language Scratch [3]. In Scratch, the user can build unlimited expressions, nest them into each other while using the basic logical operators as it is depicted in Figure 3.9. Besides building blocks, there are also other forms of representing the binary tree in UI, e.g. via the tree view [4]. However, after the research I decided to choose the 'block building' approach due to its intuitive, readable and not space-demanding design. Moreover, I know from the collected requirements that clients are not interested in building too complicated expressions (meant in tree depth) yet. Therefore, Figure 3.10 depicts the step by step flow when creating a simple composite filter from the existing cross filters. The design includes the tools to change the operator as well as expand or delete the nested expression. It should be possible to expand the expression up to some reasonable depth.

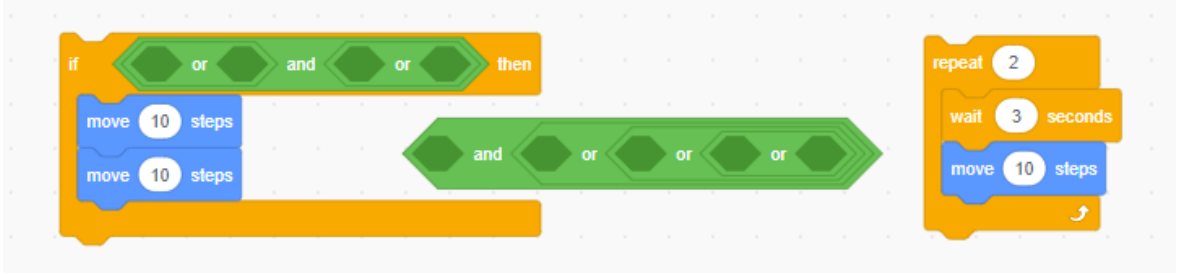

Figure 3.9: The inspiration in the form of a code snippet from the block programming language Scratch primarily targeted at children.

[3] More about Scratch - <https://scratch.mit.edu/>
[4] More about the TreeView - <https://material.angular.io/components/tree/overview>

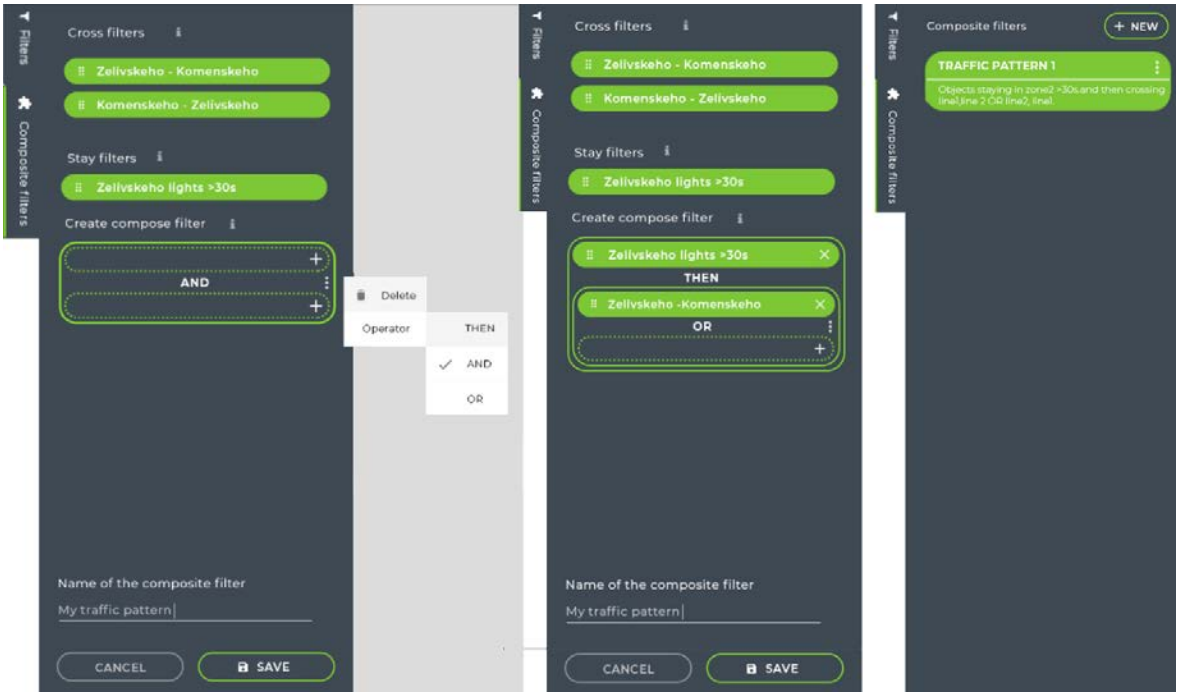

Figure 3.10: A high-fidelity prototype of the right pane where the user can create a new composite filter from existing filters.

During the design phase, I was dealing with the issue whether each tuple has to have a border. In case that the whole expression is using the equal operator, it does not matter and the tuple border can be eliminated. However, in case the user creates an expression with various operators, as it is depicted in Figure 3.11, it is not clear what term is evaluated as the first one, and therefore it might evoke multiple meanings. For this reason, I include the border within the each expression tuple.

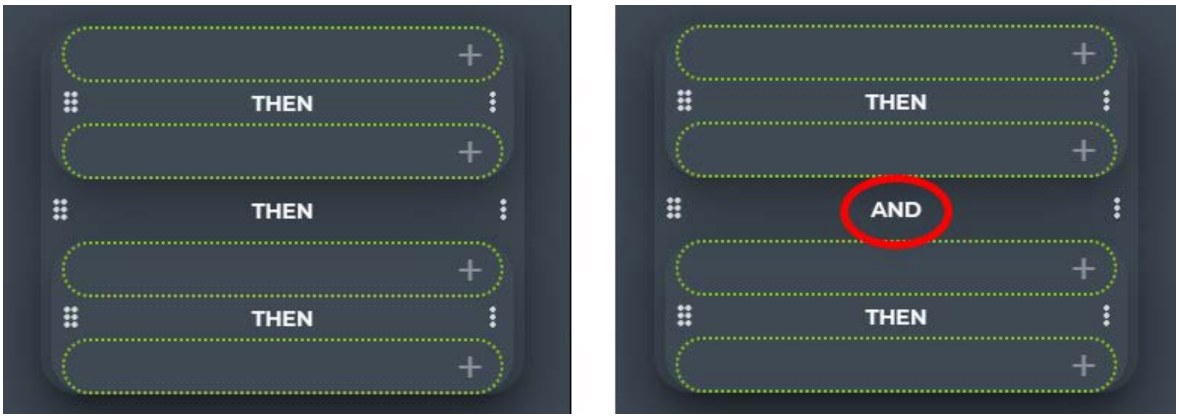

Figure 3.11: Figure on the left shows the expression where the order of terms does not matter, whereas it is crucial in the expression on the right to depict the expression's depth so that the user can build and self-evaluate the expression correctly.

Finally, I had to make sure that users can easily recognize the expression depth and therefore the order in which the whole expression is evaluated. Figure 3.12 depicts the evolution of the design that converged into the best design possible, combining three elements that emphasizes the depth - the black shadow, size of the rectangle and the contrast of the green rectangle.

Figure 3.12: Evolution of the high-fidelity prototype for the binary tree structure of the composite filter expression where number 1 represents the initial state that is not well readable, and the best design - number 5 implementing three factors for a better depth recognition.

3.2.4 Redefining the Terminology

After further research into the traffic terminology, I finally decided to enhance the UI prototypes with more user-friendly terms as Nielsen suggests [11] in his second heuristic 'Match between system and the real world', which recommends to 'speak the language of the users' with words, phrases rather than system-oriented terms. Therefore, below I list a mapping of the past terms to the new ones. I will use these new terms in the UI instead of the deprecated terms within the final implementation step. I validated the terminology after a discussion with traffic engineer specialists and a few clients.

- **Cross filter** = Movement
- **Stay filter** = Stay event
- **Composite filter** = Traffic scenario

Chapter 4

Implementation

This chapter discusses the implementation of the proposed solution to visualize, filter and report traffic data in the Video Insights web application. Specifically, I will describe the environment and third-party libraries used during the development. I will explain how I gathered the real-world traffic data extracted from camera recordings via API and finally, how I implemented the user interface based on the high-fidelity prototypes.

4.1 Environment

As mentioned earlier in the *Architecture and Design* chapter, Video Insights is an Angular web application. Therefore, it is necessary to have the node.js[1] installed for the local development and run of an Angular application. The node.js is installed together with the node package manager (npm) [2], which enables installation of the Angular packages. For the development, I used the latest version of the Angular framework - Angular 7. This version enables me (in comparison to the older versions and besides other advantages, such as fast and more efficient compilation) to use the officially implemented drag drop feature that I have incorporated in my design many times. Therefore, I do not need to use another third-party library to implement it.

Regarding the development environment, I selected the cross-platform IDE Webstorm by JetBrains [3]. It provides me with all the necessary functionality for Angular development, including an embedded terminal and a static analysis for Typescript code (tslint). Regarding programming languages, as mentioned earlier in Figure 3.2, Angular applications have logic written in Typescript[4], which is a superset of JavaScript that compiles to plain JavaScript later on. Another language I used during the development was naturally HTML, which is not a programming language but primarily a markup language for the UI definition. Besides that, I used for UI styling the Syntactically awesome style sheets (Sass) framework [5] extending the CSS limited functions to support e.g. global variables, nesting the styles or a definition of mixins. Lastly, to run the Angular application locally, I used Google Chrome browser mainly due to advanced inspect options that helped me to debug the application if needed.

[1] More about node.js <https://nodejs.org/en/>
[2] More about npm <https://www.npmjs.com/>
[3] More about Webstorm - <https://www.jetbrains.com/webstorm/>
[4] More about Typescript - <https://www.typescriptlang.org/>
[5] More about Sass - <https://sass-lang.com/>

Below, I attach Table 4.1 showing the used versions and the licenses of the technologies mentioned above.

Title	Version	License
Angular	7.2.9	MIT License
Google Chrome	74.0.3729	Proprietary freeware
Git	2.15.1.2	GNU GPL
node.js	8.9.1	MIT License
npm	5.8.0	Artistic License 2.0
Typescript	3.4.5	Apache License 2.0
Webstorm	2019.1.2	Apache License 2.0

Table 4.1: The technologies used for the development, including their versions and licenses.

4.2 Third Party Libraries

During the development, I used several Angular packages for the basic functions and UI components as well as additional libraries for the advanced visualization and special UI controls. These libraries are distributed into the system via the node package manager with the versions and licenses listed in Table 4.2 below.

Title	Version	License
Angular/*	7.2.9	MIT License
Angular Material	7.3.5	MIT License
Bootstrap	4.1.1	MIT License
Echarts	3.8.5	Apache License 2.0
Fabric	2.7.0	MIT License
Heatmap.js	2.0.5	MIT License
Moment	2.18.1	MIT License
Ng-pick-datetime	7.0.0	MIT License
Nouislider	10.1.0	MIT License
RxJS	6.3.3	Apache License 2.0
Simplify.js	2.18.1	BSD

Table 4.2: Used third party libraries, including their versions and licenses.

The majority of UI controls used in the system are from the **Angular Material** library that offers a rich selection of components with a modern uniform design. Next, it is the **Bootstrap** framework that I use mainly via Sass styles when styling the system to be responsive and usable on tablets or in a very limited way also on mobile phones. **Echarts** is used to visualize the data in widgets via pie, line, bar graphs and others. **Heatmap.js, Fabric** and **Simplify.js** libraries are used when working with canvas elements, such as the creation of heatmaps, zones, lines, directions and others. Similarly, **Ng-pick-datetime, Moment** and **Nouislider** libraries are used when handling a date time range to a new widget or when displaying an overall traffic volume on the timeline below the scene on the 'Describe page'. Finally, **RxJS** is used to handle data in a reactive way (to filter data as streams and similarly).

4.3 Client-Server Communication

As mentioned in the *Architecture and Design* chapter, the web application implements the client-server architecture style for communication with the API server. Therefore, I was using HTTP requests to create, read, update and delete (CRUD) the main entities, such as camera, video, filter, graphical object (virtual zones and lines) and widget. Thanks to the API development in parallel with the new frontend implementation, I did not need to mock the data and HTTP requests because I used the API deployed in its own development environment. It is worth mentioning that I was also working with real traffic data extracted from the processed videos and not any mocked or artificially generated ones. In my data access layer services, I created request calls via the default HttpClient from @angular/common/http namespace. The JSON responses are then bound automatically to appropriate class entities. When the data obtained had to be manipulated, I created appropriate data manager services, making the data globally available to all components that subscribe to that service. The subscription also enables to react to any data change. For instance, if I created a widget displaying a total count for a specific movement and I delete that movement, the page responsible for displaying widgets deletes all the widgets that have the source of data from the particular deleted movement.

4.4 User Interface

After the low-fidelity prototyping stage, heuristic evaluation, successive high-fidelity prototyping stage and integration description within the existing architecture, I am going to describe how I implemented the major use cases, such as 'Create a movement', 'Create a configurable traffic report' and 'Create a scenario'. Furthermore, I will mention how I resolved the visualization challenges mentioned in the *Analysis* chapter. It is also worth mentioning that besides the major use cases development, I implemented many minor tasks accompanying these use cases, such as error handling and user education (tutorials). These tasks are essential when developing a new feature into a production system used by real users. However, to mention all these implemented tasks is out of the range of this thesis and therefore, skipped.

4.4.1 Create a Movement

By slightly modifying the prototype, I created a 'right-pane' component (see Figure 4.2), which comprises of multiple panes, such as 'Movements', 'Events' and 'Scenarios' (due to unresolved terminology). They are built as independent components, and the movement component contains purely cross filters that are accessed via the filter data manager. To create a movement, the user is at first exposed to a tutorial showing a gif how to create - draw a direction of the movement. The animated direction was implemented via the fabric.js library. Besides that, to ensure the smooth animation of the direction and minimize problems with performance, the simplify.js library reduces the number of points in a polyline while retaining the shape of the whole direction [6].

In Figure 4.1 below I depict the evolution of the movement arrow where in picture one, the arrows are too small and can easily disappear next to the trajectories of cars (green trajectories). In the second picture, there are only bigger arrows, and in picture three I changed the color to white and added an extra black shadow to be visible on any background.

[6]More about simplify.js <http://mourner.github.io/simplify-js/>

Figure 4.1: Evolution of direction arrows. Thanks to its size, color and black shadow, the third option is visible on the majority of backgrounds.

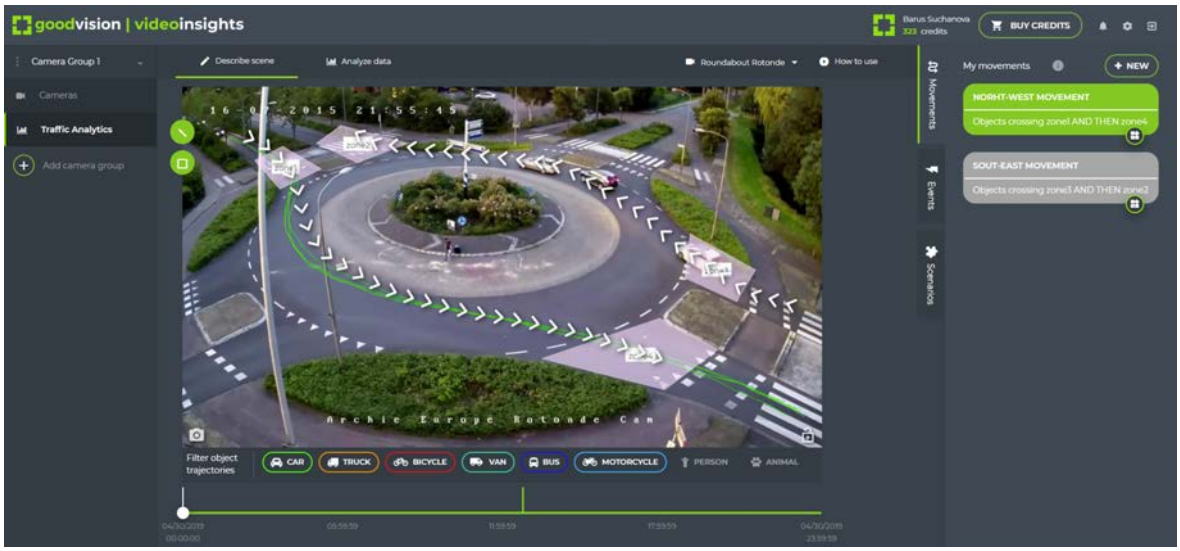

Figure 4.2: Screenshot from Video Insights displaying the implemented 'Describe page' with the camera scene showing two movements on a roundabout.

4.4.2 Create a Scenario

The most challenging feature regarding the implementation was to 'Create a scenario'. To make this feature, I built a 'Scenario' tab next to the 'Events' and 'Movements' tab, where I created a space for the scenario composition as seen in Figure4.3 below. As mentioned earlier, this scenario represents a binary tree where the children might be another binary expressions or the filter entities (movements or events) itself. This data structure is also followed by the API server that accepts the scenario in a JSON formulated as a binary tree. Therefore, I created my own binary tree data structure and the independent binary expression component (representing one tuple, as seen in Figure4.3 on the left). Then, when a user wants to create a new scenario, a root binary expression component appears and binds one tree data structure entity. Subsequently, if a user wants to expand and nest a new expression, the initial tree data entity is expanded, and the reference to that specific child is sent to the new DOM binary expression component that is recursively created inside the parent component. Similarly, if the user wants to delete an expression, it will destroy the DOM expression component (including successors). The major advantage is that without the need of subscribing to any tree entity change, the root component always contains the whole scenario tree. For future deployment, some limits of the tree depth, which is not limited by now on the backend side, should be set.

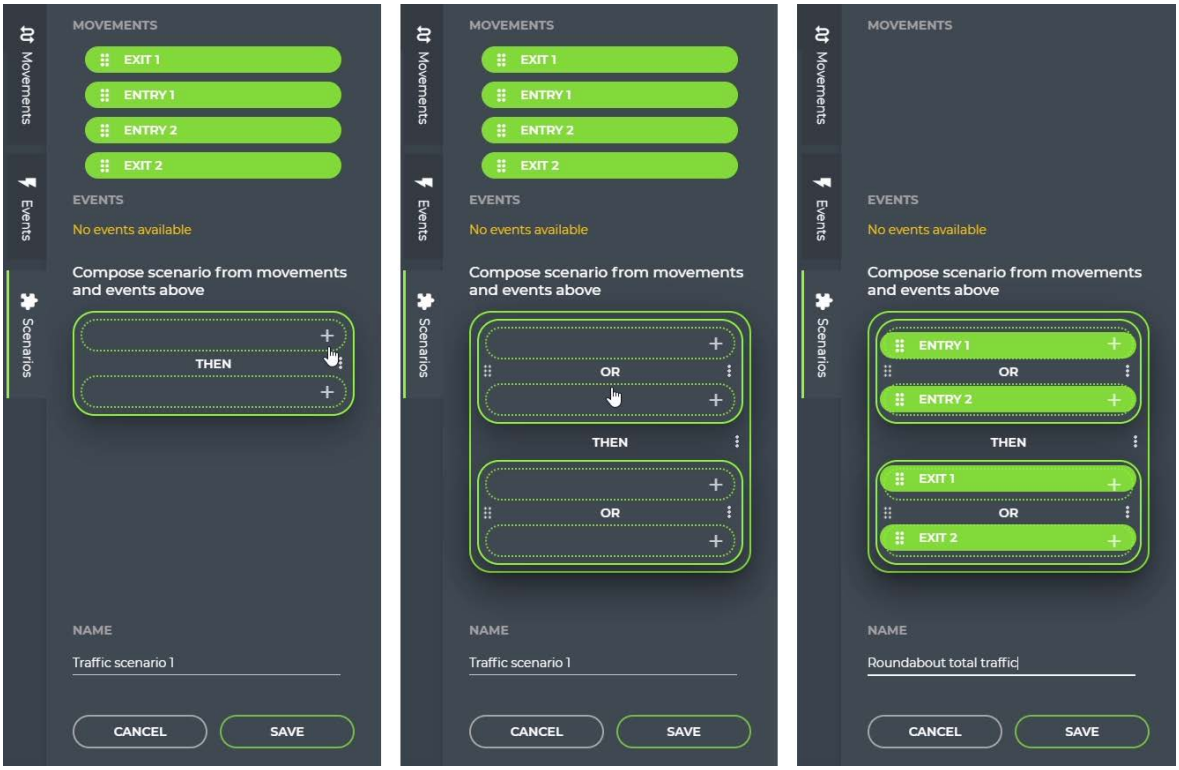

Figure 4.3: Screenshot from the Video Insights application displaying the implemented step by step creation of the traffic scenario, covering the entire traffic volume on a roundabout.

4.4.3 Create a Configurable Traffic Report

As the high-fidelity was designed, I developed a widget grid that loads all available widgets and their data from the widget data manager. Every widget dynamically sets its width in a twelve column grid with a constant row height according to the set visualization. To dynamically assign the left border color to widgets of a specific camera, I created a set of colors that are well distinguishable and color them via the pipeline during the initialization. The widget grid also supports the drag drop ordering of widgets thanks to modified Material Angular drag drop support. To see the configurable traffic report as a whole, see Figure 4.4.

In comparison to the high-fidelity prototype, the right pane (used for the creation of a new widget) was slightly modified during the implementation to provide the best usability. Now the three steps to create a widget (see Figure 4.5) are implemented via the intuitive Material Angular stepper control used often within e-commerce checkout processes. See Figure 4.5 to see the three steps next to each other.

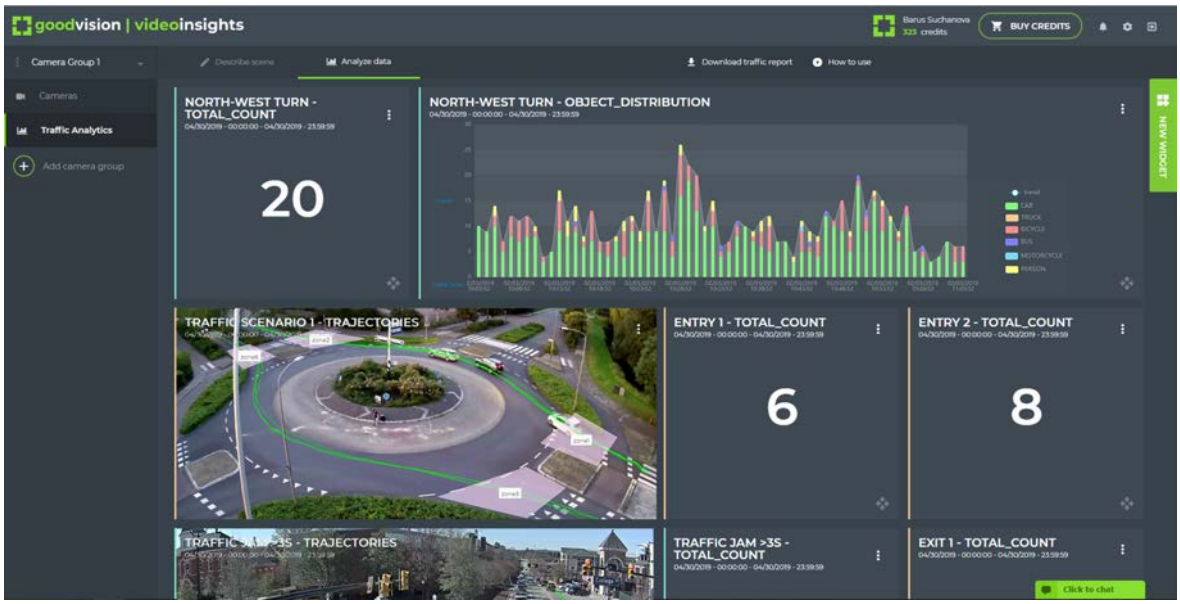

Figure 4.4: Screenshot from Video Insights displaying the implemented 'Analyze page' with widgets that show statistics and scene pictures for two cameras.

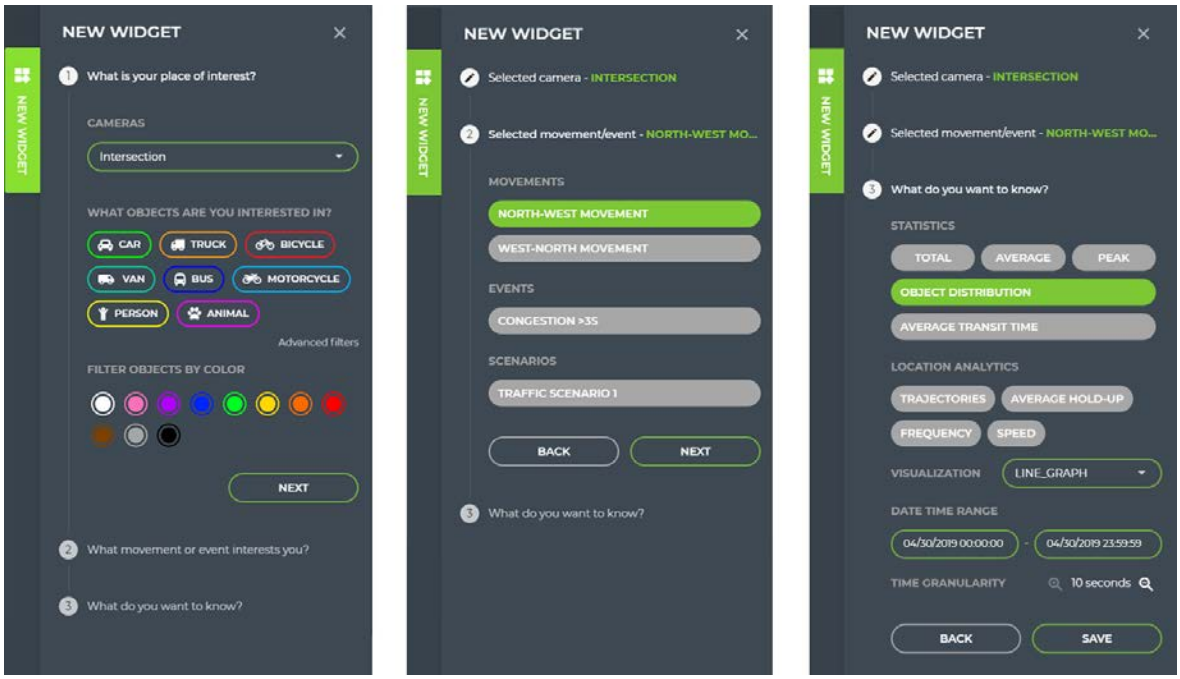

Figure 4.5: Screenshot from Video Insights displaying the step by step right tab when creating or editing a new widget.

4.4.4 Visualization Challenges Solutions

As discussed in the *Analysis* chapter, the traffic data visualization field contains many challenges. During the implementation, the web application could resolve some of them while some challenges were resolved at the backend side as well. Regarding the extremely high number of trajectories to display that caused a readability and performance problem on the frontend side, it has been solved via an asynchronous server-side rendering of the trajectories. In other words, the web application obtains trajectories only in the form of a transparent png picture that it subsequently displayed on the scene or in the widget. Also, if the user wants to know the location of the main traffic flows (for instance, when putting zones to the scene), only a sample of trajectories is sent to the frontend for this use case.

Concerning the detected object trajectories, I improved the filtering section that substitutes the object class legend as well (see Figure 4.6). Besides that, I implemented a 700ms timeout to prevent sending PUT requests to the camera every time the user changes the camera object classes. There was a case that the user wanted to select/deselect multiple object classes and it was just slowing down the system when sending individual requests instead of one request when the user stopped selecting/deselecting the object classes.

Figure 4.6: Redesign of the trajectories legend with embedded filtering. This pane is put below the camera scene in the 'Describe page'.

The last achievement regarding the visualization challenges is connected with the location analytics, particularly with the frequency and speed heatmap visualization. After the research in the analysis chapter, I replaced the non-linear rainbow gradient with the divergent blue-white-red gradient and modified the gradient steps. I used the library heatmap.js for the heatmap implementation. As you can see in Figure 4.7, in the past, it was not possible to distinguish well the very frequented places. It is because the average (represented by white color) from all matrix values is very low due to a high number of zeros and low values. For instance, in a six-hour-long video with the regular traffic covering two roads, the camera resolution similar to HD has a matrix with 810 thousand values where more than 642 thousand values are zeros and together with values lower than three it sum up to 714 thousand cells. Therefore, I tested many configurations where I skipped matrix cells with values from 1 to 10. Finally, I obtained the best results when tested on multiple scenes to skip all cells with values lower than five. Moreover, I dynamically set the gradient points for the heatmap with regards to the rest of the matrix data. The result frequency and speed heatmap in comparison to the past visualization is depicted below in Figure 4.7.

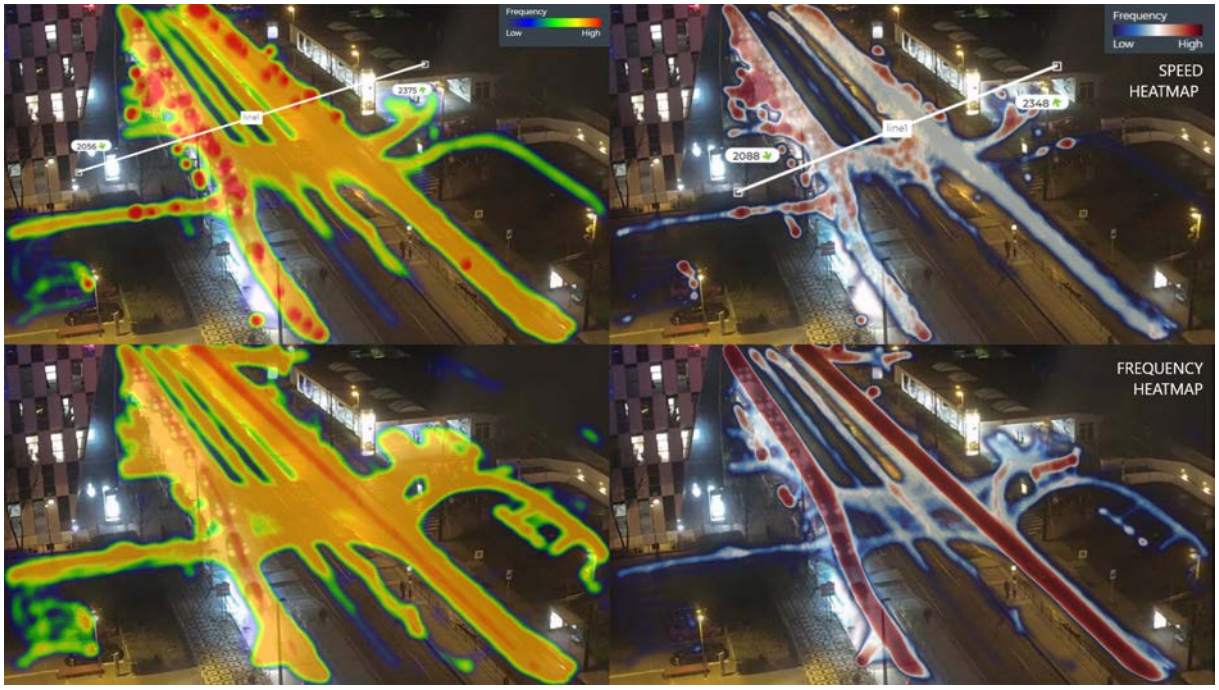

Figure 4.7: Four heatmaps displaying the relative speed of vehicles in two upper pictures and the frequency of the vehicles in a given area in the two lower pictures. The heatmaps on the left display the past visualization with the rainbow color gradient. The heatmaps on the right display the improved dynamically set blue-white-red gradient.

4.5 Deployment

Finally, thanks to the integrated support of the continuous integration (CI) tool within the Git version control system, the deployment of the new Video Insights frontend into production is a relatively easy task. Furthermore, the Angular framework provides a production build feature that compresses the whole application into several files while optimizing the code via dead code elimination, concatenating of library files and Ahead-of-Time (AOT) compilation that pre-compiles the Angular component templates [7].

However, before every deployment, all components of the Video Insights pipeline have to be tested via integration tests if they work flawlessly together. Moreover, I have to test the new frontend features via usability testing that will verify the usability of the web application as a whole. I will focus on this topic of usability testing in the next chapter.

[7] More about the Angular production build <https://angular.io/guide/deployment>

Chapter 5

Usability Testing

After the successful implementation, the next step of any software development life cycle is to perform usability testing. Usability testing verifies that the system fulfils the fundamental customer needs in an intuitive and user-friendly way. The goal of every usability testing is to recognize all minor as well as major issues regarding the UX and system control that were not discovered via the heuristic evaluation in the analysis phase. In comparison to the heuristic evaluation, usability testing is executed directly with users in 1:1 sessions[1] where the users, test participants, have to follow the test scenarios the other person, a test moderator, prepares for them.

In this chapter, I will give you a detailed overview of the test preparation, participants, the test procedure itself and findings from the usability testing I executed with the new design of Video Insights I have implemented.

5.1 Preparation

5.1.1 Approach

There are two approaches to usability testing - quantitative or qualitative testing. The quantitative testing includes a high number of participants (30 and more) that pass a quick test that might be conducted even remotely and the data from the participants (e.g. speed, error rate) are processed with statistical methods. The second, a qualitative approach, includes a lower number of participants (around 10) and it is typically conducted via a personal meeting with the test moderator. The aim of this testing is to understand the mental model that the participant is creating about the tested system as well as identify major problems in the system. As a rule, this approach requires at least 30 minutes to complete all test scenarios, including the pre-screening and post-test interview (see the next paragraph for more details). The qualitative testing is more suitable for not-common systems where it is important to lead or help the participant on the fly. Due to the uniqueness of the Video Insights system [2], I chose the qualitative approach, which is more suitable according to the definition.

[1] Other setups exist as Dumas[5] suggests.

[2] As common systems, I consider e.g. reservation systems, e-shops and others. Therefore, I do not find the system for filtering and analyzing the traffic data much known among users.

5.1.2 Setup

In this testing, I wanted the participant to test solely the part of the Video Insights system I re-designed, which is the traffic data visualization, scene description and the data filtering and analysis. Therefore, before the test conduction, I created a test workspace in the system to skip the steps not relevant for the testing, such as create an account, login, upload a video, wait for the video processing. This workspace contained two uploaded and processed videos that were ready to be described and analyzed. During the test, the Video Insights application was running locally on my computer with the development instance of the API. Furthermore, I set up the screen recording as well as an audio recording to log all valuable channels of information I can return to later. This is useful mainly when summarizing the findings as well as when searching for exact participant's citations.[3]

The next thing I needed to prepare for the testing was the environment. As Dumas[5] recommends, I chose an empty room where I prepared the setup as seen in Figure 5.1. It is important to have the closed door during the test so that the other participants cannot hear anything and have advantage during the test.

Figure 5.1: The room for the usability testing with Video Insights. A paper with test scenarios and tasks is available to the participant on the left side of the table while the notebook with the running web application is available in front of the participant. The moderator sits on the right.

Apart from that, I created a short screening form that I introduced to each participant before the actual testing session so that I could have some overall information about the participant's background. This screener asked for

- Gender, age, education and experience with Video Insights

[3]To prevent any future problems when recording the sensitive data, I created an approval form that each participant was asked to sign up to agree with the processing of the sensitive data for academic purposes.

Next, I created and printed a document with test scenarios and individual tasks to assure that each participant performs the tasks that are formulated in the same way and given in the same order (see the next section for more about the test scenarios). Lastly, I prepared a post-test interview that encouraged participants to discuss their first impressions over the just conducted test. This post-test interview contained questions, such as

A. **What was the biggest problem for you during the test?**

B. **What did you like the most about the system?**

C. **Can you imagine yourself using this system for traffic data collection and analysis if needed?**

5.2 Participants

As mentioned above, I selected the qualitative approach for my usability testing. Therefore, I selected **9 participants** in total. To obtain a relevant feedback about the system, I chose only the people that have the same attributes (age and education) as the target group I identified during the in-person interviews when collecting the user requirements. Particularly, there is a very low probability that people with none or only elementary education will be using Video Insights as well as people younger than 20 years or older than 55 years. Besides that, I selected a high variety of participants that ensures the generation of non-biased findings. Below, I introduce Table 5.2 with the participant's background based on the screening interview before the testing.

Participant ID	Age	Gender	Education	Experience with Video Insights
1	28	Male	Graduate	Yes
2	31	Male	Graduate	Yes
3	37	Male	Graduate	Yes
4	22	Female	Undergraduate	No
5	28	Male	Graduate	No
6	29	Male	Undergraduate	No
7	28	Male	Undergraduate	No
8	45	Male	Graduate	No
9	38	Male	Undergraduate	No

Table 5.2: List of participants with their background.

5.3 Testing Scenarios

It is a fact that some participants were never in touch with Video Insights. Therefore, to understand the meaning of the prepared scenarios, I encouraged every participant to embody into a typical user of the system that has a specific background. This background description (available below) was part of the printed document with test scenarios all participants had at disposal during the testing.

Background

„*You are a traffic engineer. Your job is to do regular traffic counting on several locations to collect data about the traffic volumes, including the vehicle classification in the inspected area. With such data, you are able to identify peak hours, locate the most frequent places and much more. Because you have access to cameras in the inspected areas, you want to use the tool Video Insights for fast traffic data extraction and analysis. You have already uploaded a footage of the 'Toomer's corner intersection' in Alabama, U.S., and a footage of the 'Rotonde roundabout' in the Netherlands into the system. Video Insights has already extracted the traffic data for you, and now you just need to analyze the data in the application to obtain the counting summaries you are asked for.*“

The test scenarios follow. I designed the first scenario to be very descriptive, almost like a tutorial that appears to new users when they open the system for the first time. I, as a moderator, open the system and navigate to the 'Describe page' each participant should start from. Below, I list all tasks formulated in the same way as it was for the participants and that are part of the first scenario - **Execute a traffic counting on left-turn and right-turn movement at the 'Toomer's corner intersection'**.

1. Create zones on your entry and exit zones in the scene as in the picture below (illustrative picture attached). Use the drawing tools below (illustrative picture attached).
2. Filter the trajectories on the scene to not show pedestrians and animals.
3. Create movements for traffic flows where you want to perform the traffic counting. These are the traffic movements - **from west to north** and **from north to west**.
4. Besides the traffic counting, you want to know if the vehicles stayed in the middle of the intersection for more than 3 seconds (as a traffic engineer you know it means a light congestion in the given area). Define a new stay event with the appropriate zone (draw it if needed).
5. Next, you heard about the new feature of traffic scenarios Video Insights provides, and you want to try it. Create a traffic scenario that will filter vehicles going from west to north or from north to west and stayed in the congestion in the middle of the intersection.
6. The visual data gives you a nice overview of the traffic situation. However, you need to know the exact numbers about the counts of both movements. Create two widgets that will display the total count of vehicles (classified as a car, bus or truck) that followed the west to north movement and north to west movement.
7. Download the detailed traffic report as a whole.
8. Delete both zones and widgets you created (this step returns the camera into the initial setup for the next test participants).

The second scenario is similar to the first one - **Count the total, average and peak traffic volume at the 'Rotonde roundabout'**. However, this time, I encouraged the participants to act more independently in comparison to the first scenario.

1. Proceed in the same way with the second camera. This time you want to count the traffic volume as a whole - there are multiple correct approaches to achieve that.

2. Create a scenario that will filter all vehicles that enter via one of the two roundabout entries and then leave via one of the two roundabout exits.

3. Create widgets for the scenario that will give you a total count, average and peak number of vehicles fulfilling the scenario.

4. Delete both zones and widgets you created (this step returns the camera into the initial setup for the next test participants).

5.4 Testing Procedure

The usability testing was conducted on 9 May 2019, and I created a schedule to have approximately half an hour for each participant. During each test run, I asked each participant first to sign the approval about recording the audio and screen (all participants agreed to be recorded before the actual usability testing). Next, I introduced briefly how this test will be conducted that there is the same list of tasks for the participant he/she has to complete. I emphasized to each participant that I will not help them with finishing the tasks unless they have a serious problem. Additionally, I highlighted that every step the participant makes is correct because every action helps me to understand the behaviour of users when interacting with the system. Therefore, I repeated many times to the participants during the test that even it is called a test, I am not trying to test them but only observe them how they deal with the tasks.

During the testing, the moderator can sometimes obtain valuable information only from the participant's voice and intonation. Therefore, I also encouraged all participants to read aloud the documents with the tasks as well as 'think aloud' as Nielsen [12] recommends. The 'think aloud' method can reveal hidden problems that otherwise users struggle with in silence. Particularly, during the implementation and testing with myself I noticed that some functionality (f.e. flow of edit of the movement or event) might not be ideal but because I was not convinced about that I let the participants interact with it during the test (without telling them that I think this functionality is not ideal). As a result, via the 'think aloud' method, I received useful feedback, so I am now able to decide whether my suspicion about the functionality was valid. However, I will discuss the findings more in the next section. Finally, after the test, I thanked the participants for their time and asked them to answer my post-test questions.

5.5 Findings

In this section, I am going to summarize the findings and often usability problems the participants struggled with. In the first test scenario focused on the intersection, I noticed an issue regarding defining the scene. Figure 5.2 depicts that some participants drew the zones (despite the tutorial depicting zones on individual lanes) across the whole communication. Later on, when defining movements and generating total counts, it would include all vehicles, including those going in the reverse direction, which does not need to be incorrect, but it is in most cases.

Regarding the second test scenario focused on the roundabout, the goal was to count the total traffic volume or, in other words, count all vehicles that entered via one of the two entries and then left the roundabout via one of the two exits. In Figure 5.3 below, some participants composed the traffic scenario as expected (and then created a widget for the total count) in the

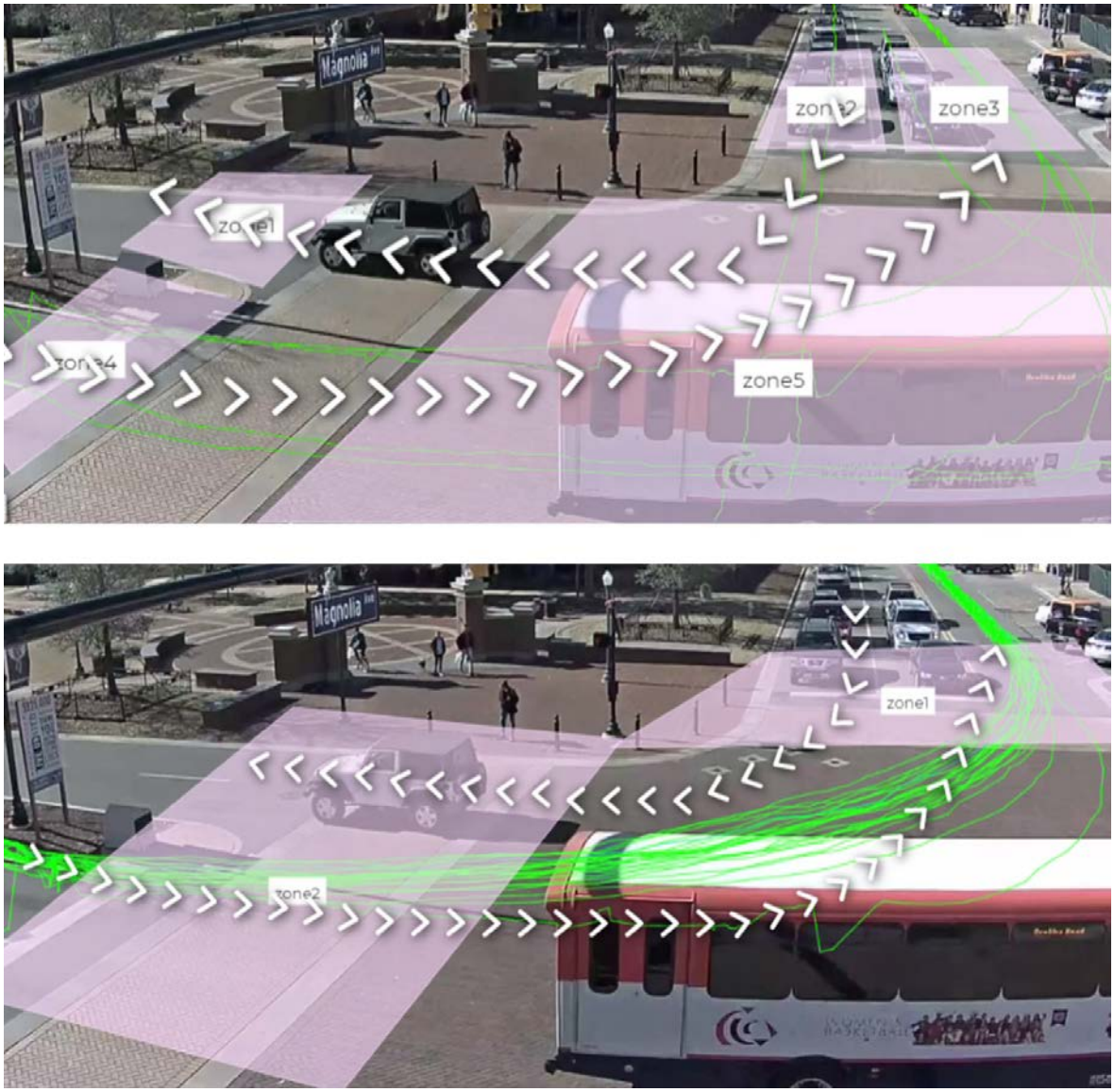

Figure 5.2: A camera preview of the intersection from the first test scenario. In the upper picture, a participant drew zones for independent lanes, whereas in the bottom picture, another participant drew zones crossing the whole communication in the given area.

first picture. Nevertheless, one participant accomplished the goal - to count the total traffic volume - via creating one big zone covering all entry and exit zones and then creating a widget from this movement. This approach is after all not incorrect, but not scalable for further use cases. I believe this will not be a common issue when it happened only once during the usability testing. However, it is important to have this user's behaviour in mind if it repeats in the future.

Concerning the composition of the traffic scenarios, I noticed that the first two participants did not know how to create a nested expression (how to expand the tree). The reason was that the plus sign in the empty cell had a low opacity (and full opacity on hover). Another problem

Figure 5.3: A camera preview of the roundabout from the second test scenario. In the upper picture, a participant drew zones for independent entry and exit zones as expected. In the bottom picture, another participant drew one big zone covering all entry and exit zones with traffic in order to obtain one number for the total traffic volume.

came up with deleting the nested expression when the same two participants did not know where to find this option. The option was hidden in the 'expression menu' on the right of each expression. During the testing, I realized how to quickly fix these UX issues, so I increased the opacity to full and added the delete icon directly on the left side of each expression (instead of the drag icon and not into the root expression) as visible in the picture 5.4 on the right side. My expectations were fulfilled when all the subsequent participants did not have a problem with creating a nested expression as well as deleting it.

Finally, the last issue, linked to the counting of the traffic volume at the roundabout, is the user's understanding of the composed traffic scenario. In Figure 5.4 on the left side, there is an example of the correct scenario query generating a correct statistics if used in widgets. On the right side, there is an interpretation of a participant who thought the scenario will generate the correct numbers if used in a widget. The problem is that the participant ignored the existence of 'virtual brackets' for binary expressions and did not understand well the logic

principles. This participant even asked me after the composition „*How do I know that the scenario I composed would give me the correct number?*“. One possible solution is to highlight all zones and lines included in the given scenario via the same color. Another solution is to offer the feature of 'preview the scenario' to the user. This feature could 'replay' a movement of an exemplary vehicle fulfilling the scenario while highlighting appropriate expressions during the movement. Nevertheless, the general visualization of the sequence of movements and events with logical operators in the scene does not have a trivial and only one correct solution.

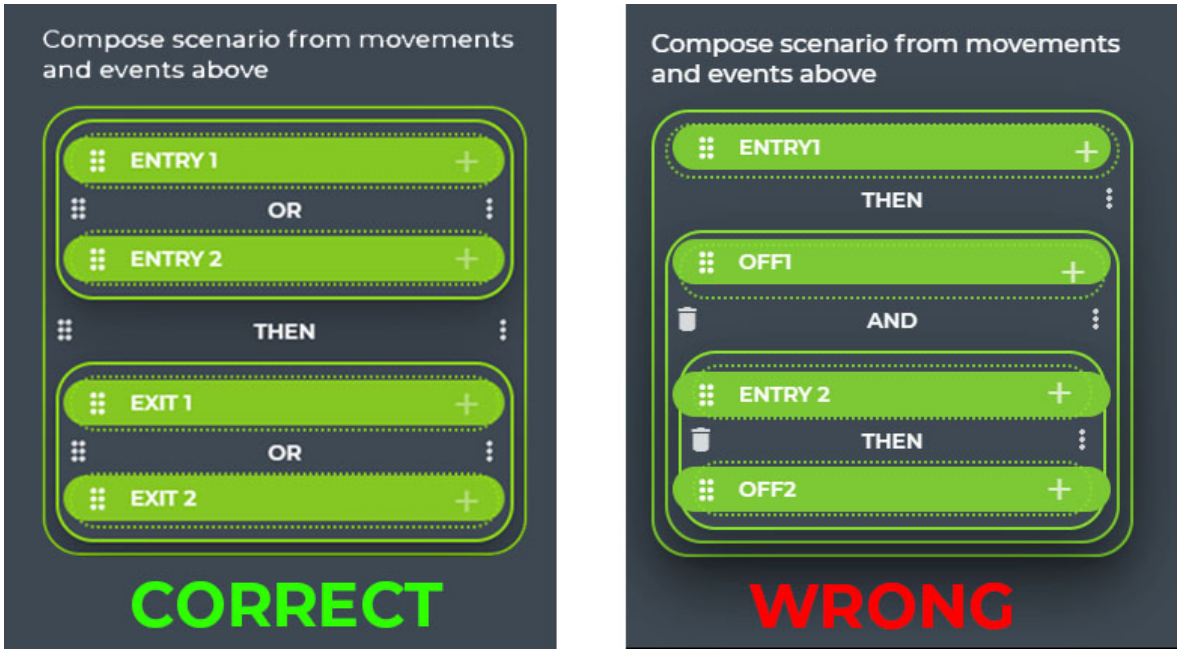

Figure 5.4: Creation of the traffic scenario from the second test scenario. On the left, there is one possible correct solution. On the right, there is a scenario (composed by a participant) that will produce an incorrect result regarding the total traffic count on the roundabout.

Some minor UX issues and their possible solutions are listed below.

- Indication for selected zone into a movement or a stay event is not visible enough. A possible solution is to increase the opacity or add a bold border when selecting a graphical object.
- The dropdown for camera selection in the toolbar on the 'Describe page' is not visible enough. Addition of the green background (as the 'new widget' button has) might solve the issue. However, in comparison to a frequency usage of the 'new widget' feature, the 'change camera' feature is used much less.
- The meaning of the time granularity for 'average' statistics is not clear enough. After a discussion with one participant, we agreed that a change of the title would be sufficient to understand it next time. [4]

[4]Selected time granularity adjusts the average step length. For example, the average amount of vehicles in a six-hour-long video footage with one-hour granularity will be calculated as the total number of objects divided by six.

Regarding the curious issues and ideas, one participant expected the system will support all sorts of shortcuts as well combinations by default. Video Insights support the basic shortcuts compliant with accessibility standards, such as pressing the escape, delete and enter key. However, keyboard combinations are not relevant for this type of application and therefore not supported yet. Another idea brought by one of the participants is to implement the 'not' logical unary operator into the scenario composition . I can imagine a small number of use cases where this might be useful, on the other side it would add another layer of complexity into the design and therefore I will not incorporate this idea into the design unless I hear this requirement from the frequent users of the system.

To conclude the *Findings* section, I present the picked participant's post-test answers for each question below

What was the biggest problem for you during the test?

- „*I think it wasn't a problem in the UI but mostly that I had to think what I want in advance regarding the scenarios.*"
- „*I did not understand that it is possible to draw a movement across only one zone.*"

What did you like the most about the system?

- „*The traffic scenarios and the actual power to combine the movements and events into a logic expression. Also, the UI is nice, the drag and drop reminds me of block programming.*"
- „*When defining the scene I knew where to put the zones, where were the traffic flows, thanks to the trajectories and even when I never saw the inspected intersection or roundabout with my own eyes*"

Can you imagine yourself using this system for traffic data collection and analysis if needed?

- „*Yes, definitely, it provides me with all what I need for my traffic surveys.*"
- „*Yes, in a few minutes, I was able to obtain the traffic counts and much more!*"

5.6 Conclusion

The qualitative usability testing with in total 9 participants succeeded and helped to improve the UX of the system according to the expectations. I verified my theories regarding the usability of some features from the design phase. Specifically, I validated that every new user interacting with a new type of system and interface has to be guided and provided by a descriptive but brief tutorial with a lot of examples, including pictures or gifs.

Regarding the findings, I divided them into three categories - normal, minor and ideas and discussed them accordingly. Besides the negative findings, I would like to highlight some positive outcomes, such as creating movements and widgets went flawlessly as well as everybody intuitively knew that scenarios should be composed via drag drop. Also, I edited some cosmetic issues on the fly and observed the success already with the next participants.

Regarding the future plans, I will inspect the usability problems I found and modify the system accordingly, realize another round of a usability testing after the modification and just after then approve it for the official deployment to production.

Chapter 6

Conclusion

In the last chapter, I am going to summarize my achieved results with regard to the task assignments and discuss the future work within the topic of traffic data visualization and filtering as well as the visionary ideas regarding this topic.

As the first task, I successfully collected the requirements needed for the future system specification and the design of the user interface. I analyzed the current state of the system Video Insights, performed in-person interviews with more than 80 people of the target audience and besides that, I analyzed solutions similar to Video Insights focused on visualization of extracted traffic data via computer vision. Based on the research, I identified the main scenarios the future UI should handle as well as the system entities and the functional model containing all use cases the UI of Video Insights should support. Afterwards, I defined the spatiotemporal data the detected objects from camera recordings hold and formulated the tasks from the visualization-related use cases. Consequently, I presented and selected some visualization methods for the classified tasks.

In the second part of the thesis, I focused on the description of the Video Insights architecture, including all major components responsible for the traffic data extraction, in order to understand how to incorporate the future user interface into the existing system. Subsequently, I sketched the low-fidelity prototypes for two non-trivial use cases and evaluated them according to Nielsen heuristics. Findings from this evaluation were used for the second high-fidelity prototyping round. Finally, I managed to incorporate the proposed design into the existing Video Insights web application. In the end, I conducted the qualitative usability testing with 9 participants to verify that the new UI is both usable and satisfactory for the target group requirements I collected at the beginning. After the usability testing, I modified the UI based on the findings to improve the user experience of the web application even more before its official deployment to production.

6.1 Future Work

Regarding the future design, I will focus on generalization of the user interface so that it is usable not only in traffic-oriented sphere. I want to design a system that might work for several other use cases where data collection from camera recordings might be useful. For instance, retail is a huge domain where installation of surveillance cameras is taken into account nearly automatically nowadays. Moreover, stores as Amazon Go, where customers are charged in real time when they put a new item into their basket, have a great infrastructure of cameras that

could be used for various use cases. One of them could be re-identification of customers across multiple cameras and therefore even tracking of these customers when appearing on different cameras in the defined space. Discovering shopper's behaviour could be another use case to convert browsers to buyers (similarly as Pygmalios [1]). However, implementing these features would require to launch the second instance of Video Insights using another set of terminology because finding correct terms that would fit and had the same meaning in both (quite different) fields is not an easy task. Moreover, this re-identification feature and the other would also require research into the AI component of the solution to be able to offer results with high accuracy.

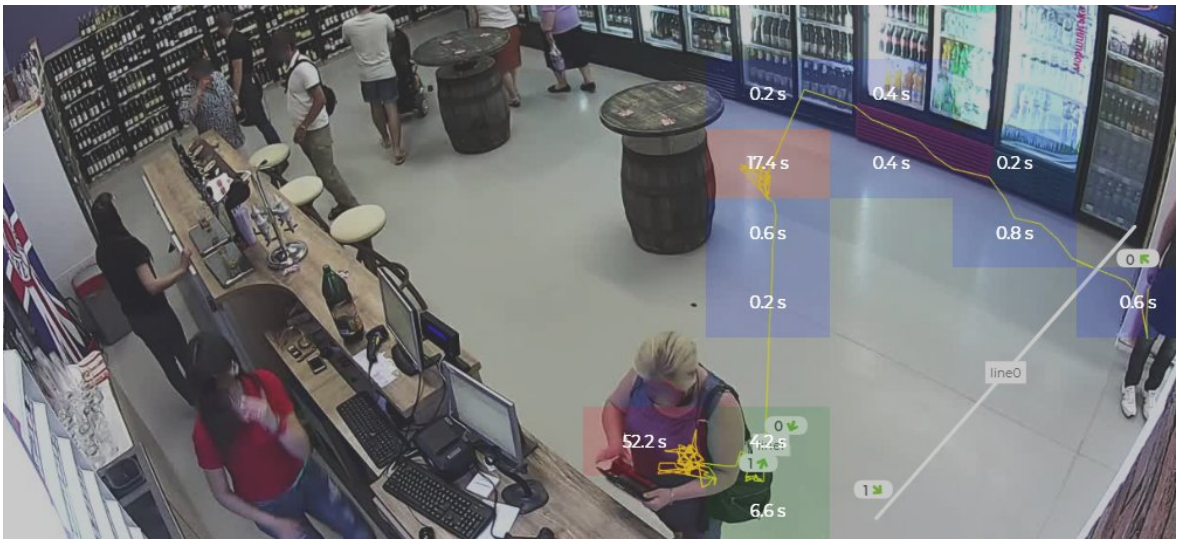

Figure 6.1: Hold-up heatmap generated by Video Insights showing average hold up time in specific areas and a trajectory of a specific person in the store.

Similarly to retail, a security domain provides a continuously growing infrastructure of surveillance cameras. Therefore, I imagine a lot of new use cases that could be incorporated into Video Insights or similar systems one day. However, the desired time delivery of the results in this field might be an issue. The results would have to be delivered in real time or at least close to that due to a simple fact. It is useful to detect an event (in comparison to an object) as soon as possible so that competent people or chained systems can react to these events in time.

Coming back to the field of traffic engineering, a big step forward can happen if the already mentioned re-identification of objects among multiple cameras is going to reliably work one day. The information about vehicles appearing in multiple cameras in a given order introduces a new UI task, such as visualization of these vehicles in context, e.g. in context of a map, might represent one possible solution. Besides other camera parameters, the geographical location of the static camera might help to understand the 'bigger picture' when trying to identify some traffic behaviour. As for reporting such a feature, it is necessary to generate some kind of a cartography report or an origin-destination report where the origin and destination would be represented by cameras instead of zones drawn into the scene as it is right now.

[1] More about Pygmalios <https://pygmalios.com/>

The topic of visualization of the data extracted from camera recordings offers a plenty of applications in various spheres of life. Nevertheless, majority of these applications require high-accuracy and fast algorithms (or at least x times more powerful GPUs) to provide the best results in reasonable time. After both of these prerequisites are fulfilled, it is meaningful to think about extending the current user interface by these new features. In any case, the extension of the UI by any of the visionary features just mentioned should not pose a problem thanks to the scalable and easily extendable user interface I designed and implemented in the Video Insights system.

Bibliography

[1] AKCELIK, R. Glossary of Road Traffic Analysis Terms. Available from: <http://www.sidrasolutions.com/documents/aaglossarytrafficanalysisterms.pdf>. Accessed on 05.04.2019.

[2] ARNOWITZ, J. – ARENT, M. – BERGER, N. *Effective Prototyping for Software Makers*. Morgan Kaufmann Publishers Inc., 2006. Accessed on 07.04.2019.

[3] BUHAUG, H. – URDAL, H. An urbanization bomb? Population growth and social disorder in cities. *Global Environmental Change*. 2013, 23, 1, s. 1 – 10. ISSN 0959-3780. doi: https://doi.org/10.1016/j.gloenvcha.2012.10.016. Available from: <http://www.sciencedirect.com/science/article/pii/S095937801200129X>. Accessed on 03.12.2018.

[4] CANDELA, I. et al. Using Cohesion and Coupling for Software Remodularization: Is It Enough? *ACM Transactions on Software Engineering and Methodology (TOSEM)*. 2016, 25, 3, s. 1–28. Accessed on 07.04.2019.

[5] DUMAS, J. – DUMAS, J. – REDISH, J. *A Practical Guide to Usability Testing*. Intellect, 1999. Accessed on 10.05.2019.

[6] ENDSLEY, M. R. – BOLTé, B. – JONES, D. G. *Designing for situation awareness: an approach to user-centered design*. Taylor & Francis, 2003. Accessed on 05.04.2019.

[7] EVAN NISSELSON, S. S. A. H.-S. 45 Billion Cameras by 2022 Fuel Business Opportunities. Available from: <https://cdn.filestackcontent.com/ZXNI76xKSKqqU7JJuahl>. Accessed on 08.12.2018.

[8] GOODMAN, E. – KUNIAVSKY, M. – MOED, A. *Observing the User Experience, 2nd Edition*. Morgan Kaufmann, 2012. Accessed on 07.04.2019.

[9] GOOGLE. Protocol Buffers documentation. Available from: <https://developers.google.com/protocol-buffers/>. Accessed on 17.03.2019.

[10] MUNZNER, T. *Visualization Analysis and Design*. University of British Columbia, Department of Computer Science, 2014. Accessed on 03.01.2019.

[11] NIELSEN, J. Usability Inspection Methods. 1994. Heuristic Evaluation, s. 25–62. Accessed on 04.01.2019.

[12] NIELSEN, J. *Usability Engineering*. Morgan Kaufmann Publishers, 1994. Accessed on 10.05.2019.

[13] RON GLAZ, J. R. World Population Growth. Available from: <https://ourworldindata.org/world-population-growth>. Accessed on 03.12.2018.

[14] SEOW, S. C. Information Theoretic Models of HCI: A Comparison of the Hick-Hyman Law and Fitts' Law. *Human–Computer Interaction*. 2005, 20, 3, s. 315–352. Accessed on 04.01.2019.

[15] SHNEIDERMAN, B. – PLAISANT, C. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. 2004.

[16] UNITED-NATIONS. World Urbanization Prospects: The 2018 Revision. Available from: <https://population.un.org/wup/Publications/Files/WUP2018-KeyFacts.pdf>. Accessed on 03.01.2019.

[17] WWW-CONSORTIUM. Relationship to the World Wide Web and REST Architectures. Available from: <https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest>. Accessed on 17.03.2019.

Appendix A

Nomenclature

ACID Atomicity, Consistency, Isolation, Durability

AI Artificial Intelligence

ANPR Automatic Number-Plate Recognition

API Application Programming Interface

ATSPM Automated Traffic Signal Performance Measure

AWS Amazon Web Services

CI Continuous Integration

CRUD Create, Read, Update, Delete

DB Database

GPL General Public License

GPU Graphic Processing Unit

HTTP Hypertext Transfer Protocol

JSON Extensible Markup Language

MVC Minimum Viable Product

REST Representational State Transfer

RPA Robotic Process Automation

SQL Structured Query Language

UCD User Centered Design

UI User Interface

UX User Experience

XML Extensible Markup Language