Institut für Visualisierung und Interaktive Systeme
Abteilung Mensch-Computer-Interaktion
Universität Stuttgart
Pfaffenwaldring 5a
D–70569 Stuttgart

Diplomarbeit Nr. 3342

Searching the Real World using Stationary and Mobile Object Detection

Markus Funk

Course of Study:	Software Engineering
Examiner:	Prof. Dr. Albrecht Schmidt
Supervisor:	Prof. Dr. Lars Erik Holmquist M. Sc. Alireza Sahami
Commenced:	August 01, 2012
Completed:	December 20, 2012
CR-Classification:	H.5, I.4

Abstract

This thesis investigates a new form of search engine, which enables the user to search for objects in the real world, just like traditional search engines locate resources on the Internet. A search engine for the real world is a step towards an Internet of Things, where real-world objects become visible to computer systems. In order to being nonintrusive, the tracking of objects is done using visual object detection. It is examined whether instrumenting the environment or instrumenting the user is more convenient in order to ubiquitously integrate a real-world search-engine into the daily life of a user. To explore those questions, two prototypes are developed and two user studies are conducted. A stationary prototype called Antonius, which instruments the environment, is built. It implements a web-based frontend and a two-dimensional map for representing the location of real-world objects. As a result of the user study, a second mobile prototype called mobile Antonius is built, which instruments the user instead of the environment. It additionally implements a 3D model of the surveyed area to represent the location of sought objects. The results introduce three categories of users represented as personas, which outline the participants' thoughts. Although a visual object detection-based real-world search engine decreases the user's privacy, the user study showed that people are still willing to use such a system for the benefit of never losing an object again. As a result of this research, the mobile system is found to be more convenient regarding privacy and intrusiveness. As well as providing a useful service, the results reveal many promising application areas in personalization, targeting and ubiquitous computing.

Kurzfassung

Diese Diplomarbeit untersucht eine neue Art von Suchmaschine, die einem Benutzer ermöglicht Objekte in der realen Welt zu suchen, genauso wie traditionelle Suchmaschinen Ressourcen im Internet finden. Eine Suchmaschine für die reale Welt ist ein Schritt in Richtung Internet of Things, in dem reale Objekte für einen Computer sichtbar werden. Um unaufdringlich zu sein, wird die Erkennung der Objekte mit Hilfe von visueller Objekterkennung durchgeführt. Es wird untersucht, ob ein Ausstatten der Umgebung oder des Benutzers praktischer ist um eine Suchmaschine für die reale Welt allgegenwärtig in den Alltag eines Benutzers zu integrieren. Um diese Fragen zu untersuchen werden zwei Prototypen entwickelt und zwei Benutzerstudien durchgeführt. Es wird ein stationärer Prototyp namens Antonius erstellt, der die Umgebung instrumentiert. Er implementiert ein webbasiertes Frontend und eine zweidimensionale Karte um die Position eines realen Objekts darzustellen. In Folge der Benutzerstudie wird ein zweiter mobiler Prototyp erstellt, der anstatt der Umgebung den Benutzer ausstattet. Außerdem implementiert er ein 3D-Modell der überwachten Umgebung um die Position der gesuchten Objekte anzuzeigen. Die Ergebnisse stellen drei Benutzerkategorien vor, die als Personas dargestellt werden und dadurch die Meinungen der Teilnehmer zusammenfassend gruppieren. Obwohl eine visuelle objekterkennungsbasierte Suchmaschine für die reale Welt in die Privatsphäre eines Benutzers eindringt, würden Leute trotzdem ein derartiges System benutzen um ein Objekt nie wieder zu verlieren. Diese Arbeit zeigt, dass ein mobiles System angenehmer bezüglich Privatsphäre und Aufdringlichkeit empfunden wird. Neben der Bereitstellung eines nützlichen Dienstes zeigen die Resultate dieser Diplomarbeit einige vielversprechende Anwendungsgebiete in Personalisierung, Targeting und Ubiquitous Computing auf.

Contents

List of Figures

List of Tables

List of Algorithms

1. Introduction and Motivation

Search engines have made a major impact on the way people use the Internet. In the beginning of the World Wide Web, it was very difficult to locate useful information unless one had the exact URL of a page. Today, this is not the case anymore. With search engines becoming available, one does not need to remember the exact location of say an interesting article on the Internet anymore, because it is easy to find it again using a search engine. This has led to the fact that using a search engine is the starting point for most users when accessing the Internet [ZSA+10]. Search engines have made the Internet much more usable, and have led to the creation of entire businesses. Yahoo!, for example, started as a guide to the World Wide Web, where interesting pages were categorized and listed, and thus helped users to find web entities. Over time, the activity of using a search engine has melted this deep with the users' daily lives that even the German term for "to google sb/sth" was added to the German Duden dictionary.

But in the real world, things are not this easy, yet. Everybody has experienced losing an object or having difficulties in finding an object at the time it is needed. But what if people could use a search engine for real-world entities to search for entities, just like we can search for information on the Web? For example, what if we could pose a search query in order to find a lost object? Imagine, a user could run an app on a smartphone, search for the lost object and receive the location and directions towards the sought object. This would mean that real-world objects did not need to be "lost", and it would save endless hours currently spent searching for missing items.

Sensor technology is advancing and is becoming cheaper. Most smartphones today have several sensors built in. For example, a light sensor that can be used to brighten and dim the display according to the lighting conditions [Hol12] or a gravity sensor to see in which way the user is holding the device. With those sensors, computer systems become context aware [SAW94] and are able to adapt themselves to several situations. Reacting upon information, received from the surroundings can also be used as an input to trigger actions implicitly [Sch00]. This means that the user's behavior in his or her daily life can be used as an input for a computer without the user's knowledge or intention. A search engine for the real world would also have to constantly be aware of its surveyed area and it has to implicitly react upon changes of an object's location, for example saving the new location. One can easily imagine that a system like this could integrate physical real-world objects into a digital network of things. This could be a step towards a smart home scenario [CYH+03] where the user's home can detect a situation and can provide help. For example, when the user's refrigerator is low on milk, the milk can be ordered automatically.

With the commercial availability of smartphones, mobile product recognition applications appeared. A few examples are: The Amazon App[1], which evolved from the former SnapTell[2] App, and Google Goggles[3]. They all work the same way. A user takes a picture of a product with a smartphone in an everyday situation, for example a picture of a coffee machine, which he or she sees in an office. The application then sends the picture to a server, which is running an object detection algorithm to recognize the product and provides information about the product in return. In this example, the user could see the manufacturer, the description, the features and the price of the coffee machine. This approach can be viewed as posing a search query without having to insert keywords [YGTD05]. Another example of a mobile product recognition application is oMoby[4]. It provides the same features as the previously mentioned systems and enhances it with human operators (so-called real-time crowd-sourcing), which provide information if the algorithm is not able to recognize the product automatically. This is the case, when a user is taking a picture of a captcha and is sending it to the oMoby system [Hol12]. Those captchas are designed to being unreadable for a computer. Thus, the task is being forwarded to a human operator, who sends back the decoded text of the captcha. Those mobile product recognition systems take some time to recognize a product because the whole image has to be transmitted to the server. In [TCC+10] a system is being introduced, which compresses the image into its features and only sends a feature vector to the server. With this technology, the system is faster compared to the other systems, but only if it is able to recognize the product. If it is not capable of recognizing it, the task cannot be redirected to a human operator because the sent feature vectors are not human-readable.

Those visual product recognition systems are a great improvement towards the past, where the object to be recognized had to be equipped with a barcode in order to combine digital information with real world objects. The Webstickers project [LRH00], for example, enabled the user to print out a one-dimensional barcode onto a sticker and to apply it as a label onto any real world object. The barcode was associated with a URL, where additional information could be provided. Other established systems to combine digital information with real world objects are, for example, ISBN numbers to identify books or the recently popular QR-Codes to encrypt URLs.

Also, indoor location tracking has been a research topic for many years now. Over the years, many systems evolved and were published. The Active Badge Location System [WHFaG92] is an infrared-based system which enables tracking on a room level. The user is wearing a badge which sends a signal including an ID every 15 seconds. The signal is being detected by sensors which are installed in every room and being sent to a backend. Applications then can ask the backend for a user's current position. Another system, where the user is carrying a sensor tag, is the Active Bat system [HHS+02]. It uses a radio frequency to trigger a mechanism in the wearable bat. The bat then emits ultrasonic signals, which are received by sensors within the room. With the use of multilateration, the system can calculate the user's position and the user's direction of view. A system like this can be used, for example, to find a colleague within

[1] http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000291661
[2] https://itunes.apple.com/en/app/snaptell/id291920403?mt=8
[3] http://www.google.com/mobile/goggles/
[4] https://www.iqengines.com/omoby/

an office environment. The Cricket Location-Support System [PCB00] uses a combination of radio frequency and ultrasonic signals. The user is wearing a listener, which is able to receive those signals. The testing area is equipped with beacons that emit both types of signals at the same time. Radio frequency travels with the speed of light and ultrasonic signals travel with the speed of sound. Due to that, the listener can calculate the distance towards the beacon. This is similar to the common estimation of the distance towards a thunderstorm, where one has to count the seconds between the visual lightning and the sound of thunder.

This thesis focuses on combining indoor location tracking and product recognition to build a location-based visual real-world search engine. As a result, it is being discussed whether it is better to instrument the area of application in order to build a location-based real-world search engine or to instrument the user. Instrumenting the area of application results in installing cameras within an area of application, whereas instrumenting the user results in a user constantly wearing a camera. As a proof of both concepts two prototypes are being introduced, implemented, evaluated, and discussed. The main focus of the thesis is to evaluate the feasibility of an integration of a visual real-world search engine into the users' daily lives in the terms of ubiquitous computing [Wei91], meaning that the system will merge with the users' lives and will not be recognized as a computer system anymore.

Outline

The remainder of this thesis is structured as follows: Chapter 2 gives an overview about related real-world search approaches concerning stationary systems. The systems are assigned to categories and a concept of a stationary real-world search engine is introduced. Chapter 3 provides an overview about object detection algorithms and filtering techniques. Then, Chapter 4 introduces a stationary prototype called Antonius. Detailed information about algorithms, used frameworks, and the architecture is given. Chapter 5 evaluates the previously built prototype and discusses the results. As a result of the evaluation, Chapter 6 introduces a mobile prototype called mobile Antonius. Also additional related work is provided and the system is evaluated. Chapter 7 concludes the thesis and provides a short overview about the performed work. Finally, Chapter 8 suggests future improvements and ideas, which came up during the thesis.

1.1. Application Scenarios

During the preliminary stages of this thesis, the author conducted 5 informal interviews with friends to gather ideas where to set up a real-world search engine. In this section, the found scenarios are described and explained.

1.1.1. Scenario 1: Fridge Scenario

The system may be set up inside a refrigerator to keep track of the available food inside. It could be combined with a recipe-collection in order to determine which ingredients the user needs to cook a certain meal. The user could even check the contents of his refrigerator while being at a grocery store using a mobile device. This kind of system could also warn the user if a product is inside the fridge for more than a week and has expired.

1.1.2. Scenario 2: Storage Room

In the USA, it is common to rent a storage space to put unused items inside. Those storage spaces can be rented over several years and it is likely for people to forget what they put inside there. An object detection-based search engine could be installed inside such a storage room to keep track of the items that were put there over the years. Those storage rooms are typically located some miles away from a users home. The ability to access the inventory via the Internet may be a great improvement to the current situation. This could also be applied in the commercial sector, e.g. in a warehouse to keep track of a company's inventory. Usually, the companies use barcodes to track items within a warehouse. An object detection-based search engine could also detect barcodes and facilitate the workflow for a person working in a warehouse.

1.1.3. Scenario 3: Inside of a Home

The system could be set up throughout an apartment to keep track of important personal objects, for example an employee badge. Imagine a user is outside the apartment on his way to work. Suddenly, he recognizes that he didn't bring his employee badge. He might think that he lost his badge on his way to work and might panic about the fact that he lost it. In this case, a real-world search engine could clear his conscience. The user could run an app on his smartphone, which tells him that he just left the badge at home. Also, the system could provide a feature to define places, where specific objects should be. For example, if the user has a place where he always stores his keys, the system could warn the user if his keys are in a different spot. Such a feature could spare the user a couple of minutes in the mornings, which he otherwise might spend searching for his keys. A real-world search engine could even learn itself, if an object is always stored at the same position within the apartment and whether the object is being taken along with the user when leaving the apartment or not. In case the user is forgetting an object that is usually being taken along when leaving the apartment the system could warn the user.

1.1.4. Scenario 4: Tracking a Collection

As a result of the informal interviews, the participants told that they would want to use the system to track a collection. In Germany, it is a widely spread hobby to collect model trains. In the interview a male participant told that he is collecting model trains, while a female

Figure 1.1.: Photographs of the participants' collections - Left side: Model train collection of a male participant - Right side: Nail polish collection of a female participant.

participant told that she is collecting nail polish. The participants were asked for a picture of their collections in order to see whether the collections can be tracked visually or not (see Figure 1.1). The left image in Figure 1.1 shows a model train collection in a cabinet. The trains are placed inside the cabinet in a way that every train is fully visible for a spectator and for a camera. Also, the trains look very distinct and could be easily distinguished from each other. A visual tracking of the nail polish could be a more difficult task because the bottles do not provide enough unique features. Also, the way that the bottles are stored is not optimal for visual object detection, because some bottles are hidden by other bottles in front of them. A use case could be a user with a huge collection of model trains being at a garage sale and the user is not sure whether he already owns a model or not. The user could access the system, which is surveying his collection, via his smartphone and see if he already owns the model. In addition, the system could visually recognize the model being sold at the garage sale and tell the user if he already owns it.

1.1.5. Scenario 5: Tracking Persons

An object detection-based real-world search engine could also track persons within a defined testing-area. To achieve this, facial recognition and voice recognition techniques could be used in addition to recognizing objects. Benefits of a use case like this could be building a person-based location sharing system like Google Latitude but without using a smartphone. Another interesting use case could be setting up the system at the apartment door. When a visitor is arriving, the system could tell the user who is at the door.

2. Related Work

Today, search engines for the Internet are a well established part of the Information Retrieval field, and have become the foundation of successful companies such as Google or Yahoo!. Search engines for the real world is a newer concept and has only emerged recently with advances in sensor technology and recognition algorithms. On a conceptual level, they can be considered a part of the Internet of Things (IoT) vision, where real-world physical objects get connected and equipped with processing and sensing capabilities [MF10]. There are already a few real-world search engines. In this section, an overview of those engines is provided and the engines are being categorized. As a result, the author's system, which differs from other IoT-systems in that there is no need for any modification to the objects, and that the system can store previous states belonging to detected objects, is being introduced at the end of this section.

Real-world entities can be categorized as people, places and things [KBM+02]. Entities belonging to each category can be searched for and properties of those entities can be queried. This research focuses on searching for the location of things. The location-based search of people and places is already solved. In order to search for people, Google Latitude[1] can be used to view the location of friends in real-time. This can be useful to organize spontaneous meetings when a friend is around. The location-based search for places is also already being done by applications like Yelp[2]. Those applications can tell the user e.g. the nearest bar based on the user's position. It also uses a rating system to tell the user, which is the best bar around. In the following sections approaches towards location-based things are being introduced and described.

In this research, the real-world search engines are divided into two categories. First, search engines that are capable of searching for an object having a specific state (*state-based*), e.g. a sensor network enabling the user to search for a place that is warm. This could be useful for a user inside of his home looking for the best place to store a pastry. And second, systems that focus on determining the current location of an object (*location-based*), which can be useful to find lost objects.

Römer et al. [ROM+10] define 9 criteria for categorizing real-world search engines, which are capable of both performing a state-based and location-based search. In the following enumeration, which is based on [ROM+10], those criteria are described.

[1] http://www.google.com/latitude
[2] http://www.yelp.com

1. *Query Type*: It is being distinguished between an *ad hoc* query and a *continuous* query. An ad hoc query is returning a result immediately after the query was received. After returning the result, the query is finished. A continuous query on the other hand, is returning results as long as the query is open and is thereby enabling a live view of a sensor's data.
2. *Query Language*: There are two types of query languages: a *keyword-based* query language and a *complex* query language. In the keyword-based language, the user can just search for keywords, which have to be contained within a found entity. In contrast, a complex query language can also constrain the state of the entity that is to be found.
3. *Query Scope*: It is being distinguished between a *local* domain and a *global* one. The local domain is just capable of a small local setup, whereas the global one is capable of searching for things all over the world.
4. *Query Time*: Defines whether the queries just apply for the *real-time* state of an entity or if *historical* states are being taken into account as well.
5. *Query Accuracy*: Many algorithms use approximations to improve the response time of a query. When an algorithm uses this technique, it is called *heuristic*. In contrast, when an algorithm takes all results into account and does not use approximations it is called *exact*.
6. *Query Content*: There are 3 types of content. The *static metadata*, which can be queried (e.g. name or type of the entity), the *pseudo static metadata*, which can change every once in a while (e.g. the favorite place of an entity) and the *dynamic content*, which is the current state of a sensor (e.g. the location of the entity).
7. *Entity Mobility*: Some systems can handle *mobile* entities and other systems only support *static* ones.
8. *Entity State*: A state is a value on a *categorial* scale (e.g. it is "noisy" or "quiet") or on a *continuous* scale (e.g. the current volume in decibel)
9. *Target Users*: A search engine could be designed for *domain experts*, who can use SDK's and call API's in order to use the system, or on the contrary it could be designed like a commercial web search engine in order to support *end users*.

Figure 2.1 categorizes all systems that are introduced in this section and assigns them to the previously introduced criteria.

2.1. State-based Systems

In this section, state-based system, which are capable of searching for a specific real-time sensor state, are introduced.

Senseweb [KNLZ07] is a state-based system, which enables the user to query the location and the current state of a sensor. It is a distributed network of sensors having a central repository,

Dimension	Snoogle	MAX	OCH	DIS	GSN	SenseWeb	Dyser	Antonius
Query type	ad hoc	ad hoc	continuous	both	ad hoc	ad hoc	ad hoc	ad hoc
Query language	keywords	keywords	keywords	image	keywords	keywords+geo	keywords	keywords
Query scope	local	local	global	local	global	global	global	local
Query time	real-time	real-time	real-time	both	real-time	real-time	real-time	both
Query accuracy	heuristic	heuristic	heuristic	heuristic	exact	exact	exact	exact
Query content	pseudo static	pseudo static	dynamic	dynamic	static	static	dynamic	static
Entity mobility	mobile	mobile	mobile	mobile	mobile	mobile	mobile	mobile
Entity state	categorial	categorial	categorial	continuous	categorial	both	categorial	categorial
Target users	end users	end users	end users	end users	experts	experts	end users	end users

Figure 2.1.: The introduced systems categorized based on the criteria. Source: [ROM+10] - edited

where metadata of each sensor are being stored. Senseweb has a central repository that is part of the system, which stores all metadata for each sensor. When a user wants to search for a specific sensor, he or she can query the central repository. The system does not update the current state of the sensor, so queries can only encompass the static metadata. Once a suitable sensor has been found, the current state and further changes of the state are being sent directly to the requester. Senseweb also introduces a sense gateway to provide a uniform interface for all sensors in the system. This means that all sensors can be accessed in the same way. An owner of a sensor might not want to share all of the data with the rest of the world. Therefore, the sense gateway also controls the data, which is made available to the public, and provides privacy options. Senseweb also contains a mechanism for mobile sensors - a mobile proxy. This proxy can handle highly mobile sensors, because it provides a general interface even when the sensors are only available for a short time. The mobile sensor can update its location at the mobile proxy while the proxy's location stays the same. Using this proxy, a requester can work with a static address or location while the sensor itself is highly mobile.

Global Sensor Networks (GSN) [AHS07] is a piece of middleware that connects sensors around the globe. It uses virtual sensors as a level of abstraction. The system does not distinguish between what is behind a virtual sensor. A virtual sensor could be a cell phone, a physical sensor or a combined data stream unifying several and diverse sensors. There is only one virtual sensor connected directly to the system. This node is called the sink node. All other sensors, which are connected to the same local network, are in a local peer-to-peer network with the sink node. All communication is led through the sink node. A user can search for a specific sensor either by addressing the sensor's ID or searching for keywords. The search for an applicable sensor, which matches the sought keywords, is being done using a peer-to-peer approach starting at the sink node. GSN hosts several virtual sensors within a GSN container running as a task on a computer. As GSN is a distributed network, the containers also build a peer-to-peer overlay to be connected to each other via the Internet. The system additionally supports mobile sensors using the transducer electronic data sheet (TEDS), which is a mechanism to describe sensors automatically. GSN builds a virtual sensor for every mobile sensor that appears using the description which is being pulled from TEDS. If a sensor disappears the corresponding virtual sensor is deleted. Compared to Senseweb, which has a central repository of metadata, GSN is more scalable because of the distributed approach. On the other hand, when a certain sensor is being searched for, many other nodes

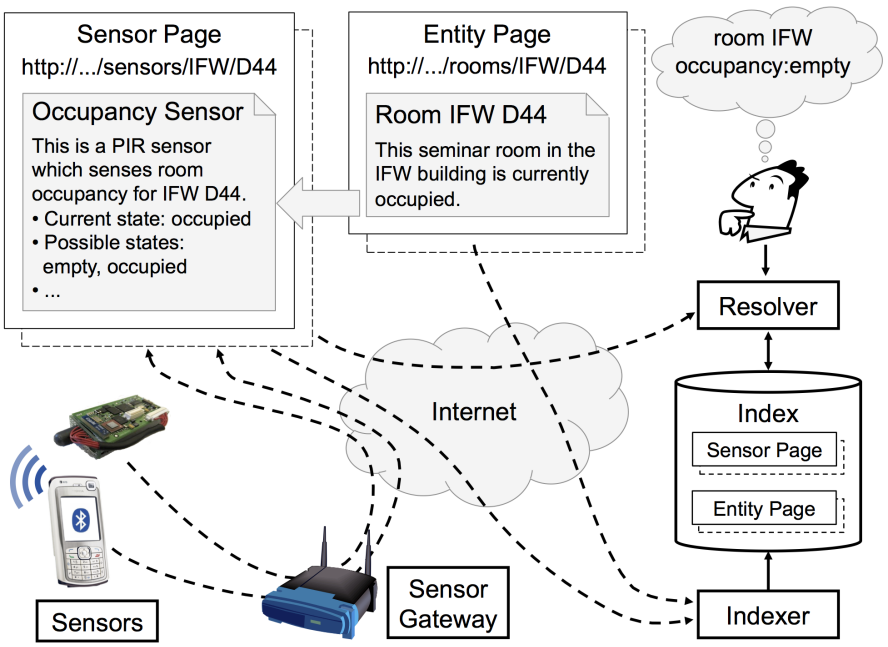


Figure 2.2.: The architecture of Dyser and a user posing a query for a free room. Source: [ORM+10]

are involved in the discovery. GSN also does not support content-based search, e.g. querying a sensor that has a certain current state because each current data packet would have to be sent to all sensors involved in the search [ROM+10]. As a result, both systems have a uniform way to access diverse sensors and provide a way to deal with mobile nodes.

The Dyser prototype [ORM+10] [ROM+10] is the first real-world search engine allowing the user to search for sensors by their current state. Dyser even combines the current state of the sensors with real-world entities (people, places and things [KBM+02]) to make the search less technical (e.g. just searching for a specific sensor value) and more user-centric. As a result, a mapping is introduced to combine specific sensor values with everyday search-terms. A user could search for a conference room that is empty (see Figure 2.2). Dyser then checks all places tagged as a conference room, having a PIR sensor[3] with a current state "free". For each sensor, a corresponding sensor webpage, which contains a description and information about the current state and possible other states, is being created. This webpage can be accessed via a URL. The system also has an entity page for each real-world entity combining one or many sensors to a real-world entity (for example the conference room). Dyser also has a central index of all sensor and entity pages, called the resolver, which is used for the lookup. A user can query the resolver for a real-world entity, for example a "conference room", which currently is in a "free" state. The resolver first checks all known entities for the keywords "conference

[3]A PIR sensor can detect movement within its reach.

room" and then determines whether the entity can attain the state "free". If it found a match, the resolver queries the candidate for its current state. As soon as it found k entities having a matching current state, a list of those entities is returned to the user. To address scalability issues, Dyser doesn't query all sensors which can match a posed query, but it implements a prediction model following heuristic assumptions to query only the sensors having the highest probability to currently hold the desired sensor state. Compared to Senseweb and GSN, Dyser introduces a significant improvement, which is the possibility to search for current sensor states. Senseweb and GSN only provide the possibility to search for static metadata. Dyser also has its downsides because it is bad at dealing with mobile sensors. Every time a new sensor needs to be integrated with the system, its sensor gateway needs to create a sensor page and an entity page. Dyser could fix the mobility problem by combining the sensor and the sensor gateway as suggested in [ORM+10]. Like Senseweb, Dyser implements a central directory, where the lookup is done. In contrast, GSN solves this problem by implementing peer-to-peer techniques. Overall, each of the 3 systems provides a uniform way to access heterogeneous sensors.

2.2. Location-based Systems

This section introduces systems, which are location-based. Location-based systems are capable of determining the location of sought real-world objects.

An approach for building a location-based real-world search engine is Snoogle [WTL10]. In this system, the objects are also getting equipped with a marker. The markers are carrying a textual description of the entity onto which they are placed. This way, the system can deal with new objects and does not require registering the object at some centralized database. Snoogle has a two-tiered architecture consisting of index points (IP) and key index points (KIP). An IP is responsible for a domain, e.g. a single room. The IP can see all markers that are within the IP's range. The KIP can see all IP's, which are inside the global network. When a new object comes into the network, the object's marker sends its description to the IP, which is responsible for the area where the object is located. Then, the IP sends this message to the KIP. Both process the message and build an internal database in order to being able to react to queries. The user can query a specific IP to check if an object is located in the IP's area or the user can query the KIP to search for the object globally [ROM+10].

The MAX system [YSM05] also uses markers, which are put onto objects, containing a textual description of the object. MAX uses a three-tiered hierarchy of sensors. The lowest sensors are the substations, which survey furniture within a room, e.g. a table or a shelf. When an object is being placed on a shelf, the sensor is able to recognize the object's marker. Therefore it knows that the object is on the shelf. The second tier consists of the so-called base stations. Those base stations are bound to a geographical space [ROM+10] such as a room or a car. The base station is able to communicate with the substations within the base station's area in order to find objects. The highest tier is the MAX server, which is able to communicate with all base stations. The MAX server can trigger global searches including all rooms covered by the system. MAX uses a pull-based approach to query results. The user can select which base

stations a query should cover. Then, a query is being sent to all selected base stations. Those base stations check within their area of reach if any object matches the keywords contained in the query. When one or multiple results have been found, the base stations sort the matches and return the top k hits to the MAX server and the user. Compared to the Snoogle system, which updates the information of the overlying layer every time something changes, the MAX approach is not scalable for global networks [ROM+10]. Snoogle's push-based approach does not involve each object every time the user poses a query, and is therefore a better solution regarding scalability.

Considering the accuracy of both system's queries: Snoogle uses Bloom filters to shorten the queries, which means that the information sent to the IP's is hashed to keep the communication cost low. MAX on the other hand does not use a mechanism to shorten the queries. Therefore, the MAX queries are 100% accurate. The Snoogle queries can contain false positives (meaning that an object was found at a location, where it is not located) but the communication cost is lower, which makes the queries faster [WTL10].

In the objects calling home (OCH) project [FBMK08] [FBRK07], Bluetooth markers are used to keep track of the objects. Those markers are placed onto the object and have to be supplied with a battery to be able to send a Bluetooth signal. The mechanism for discovering an object is the idea of the user running a Bluetooth scan constantly. Whenever a user's phone found a tagged object, the application tells a backend via an event that the object is in range. The application also tells the backend when an object is being moved out of the phone's Bluetooth-range or when the location of the phone changed. In this way, the detection of an object's location is being distributed to many phones. This crowd-sourced approach covers a larger area the more users participate. OCH also introduces a user index, where each registered object is assigned to a user to whom the object belongs. This resolves some privacy issues, because only the owner could be able to search for his or her object. OCH also uses an approach to not query all phones in order to find an object. It uses some heuristics to guess the participating phone, which can most likely detect the sought object. Such heuristics could be [ROM+10] : 1) the object might be close to it's last seen location. 2) The object is likely at a location which its owner has visited recently or 3) where the owner spends a lot of time. Taking those assumptions into account, only a few phones have to be queried in order to find an object. Compared to the MAX project, where every object is queried, the OCH project is more efficient concerning the number of sensors involved in a query. Compared to MAX and Snoogle, the OCH project differs in a few points. The object's marker does not hold its description and its keywords - It just stores an ID. This takes the computational burden away from the nodes and towards the backend, which saves the battery of users phones. The second new concept, which is introduced by the OCH project, is the mobility of sensors. The sensors do not have a fixed location anymore. A surveyed area is divided into fixed cells, where a phone can be in. Whenever a phone changes its cell, the backend is being notified. With this approach, there is a possibility that not every cell of the surveyed area can be queried at any time. For example in an office scenario: If there is a meeting, everybody will be in one room. This disables the ability to search other rooms during the meeting. But the system can still use the last known locations to estimate the position, even if there is no sensor around.

Every system introduced before requires the user to put a marker onto each searchable object in order to register it with the system. Finally, Distributed Image Search (DIS) [YGM08] allows searching for objects without equipping them with a marker. DIS uses a distributed network of camera sensors to detect objects. It uses the SIFT algorithm [Low04] (an object detection algorithm - more see Chapter 3) to extract features from live images and compares them to the features extracted from a user submitted image. Therefore, DIS calculates the SIFT descriptors of every captured image and builds an inverted index in every sensor-node to be able to compare it to other descriptors. DIS also enables the user to search within a history of previously recorded images. When a user poses a query, he is sending an image to a centralized mediator, which calculates the descriptor of the sent image. Then, the descriptor is sent to the sensors, which compare it to their inverted index in order to find matches. The nodes send a ranked list back to the mediator, which builds a global ranked list and retrieves thumbnails from the top k results. Those top k results are being displayed to the user. DIS is capable of both ad hoc queries and continuous live queries. In the ad hoc mode, the system only takes the historical images into account. In the continuous mode on the other hand, the sensors notify the mediator every time a live image matches the sought image. DIS also suggests implementing a motion detection technique to only process potentially interesting images. If there are no changes within a scene, the system does not need to analyze the objects anymore. This can save a lot of energy and keeps the computational burden low. Compared to the Snoogle system, DIS uses the same approach of storing results in an inverted index to prepare for queries. In contrast, the MAX system queries each node on demand. DIS can be assigned to a push-based approach of dealing with results. When a result is found, it is being pushed either to the local database or to the mediator in case a query is opened.

2.3. Concept

This section introduces a new concept for a location-based real-world search engine, which relies on object detection to identify real-world entities. The system is described from a conceptual and a user point of view.

2.3.1. Procedure and General Idea

The goal of this chapter is to build an object detection-based real-world search engine, which is instrumenting an area in order to be able to locate objects in the user's physical environment. To achieve this, the idea is to equip an area with high-definition cameras, which are constantly surveying everything within it. The system will then be able to compare the images coming from the live feed with a collection of images of previously registered objects. Those images can be stored in a database. The system should check periodically for changes in the environment and keep track of the registered objects' availability. The user should be able to query information about all registered objects using a simple search interface. Information that could be provided is the current location, previous locations, a timestamp indicating when the object was last seen or the current availability of the object. Also the user should be able to register documents with the system. The system should implement OCR (Optical Character

Recognition) to facilitate the registering process. With the use of OCR, some searchable terms can be auto-detected and need not to be added manually.

This concept is also integrated into Figure 2.1. It is ad hoc, keyword-based and targets the end users, because it queries a database, which can be accessed by the end user via a frontend. Because the system knows last seen locations and the real-time location of an object, the concept complies with both query time categories. The queries to the database do not use heuristics and are based on static, pre-defined keywords. Entities are mobile and can change their location within the equipped area. An entity's state is being represented categorial, based on the camera that currently sees the object.

The described concept is similar to the previously introduced DIS system, because it relies on visual object detection. However, DIS does not build a centralized database of locations and does not have any knowledge about objects. It just compares features of images. Although, it is possible to search in previously recorded images, DIS does not store previous locations assigned to each object. It only stores previously recorded images. The described concept will speedup the search for previous locations a lot, because the features do not have to be compared each time the user poses a query. Also, the described concept provides a keyword-based search, which is supposed to be more user-friendly and less computationally demanding. In contrast, the DIS project uses images as an input, which have to be recorded, transmitted and analyzed every time the user poses a query.

2.3.2. User Point of View

From a user's point of view, this real-world search engine looks the same as a text-based search engine. It has to provide a search interface and a way to display found results. Compared to a commercial text-based search engine like Google or Bing, a real-world search engine using object detection needs to implement a mapping between text-based search queries and stored images [ROM+10]. However, the initial system is not able to recognize all objects that the user is looking for. Therefore, another requirement is a means to register custom images of real-world objects with the system and to associate searchable terms with it. According to [ZSA+10], meta-data for a real-world objects have to contain an ID, a name, potential aliases, a type, potential subtypes and a description. A user will have to provide this information when registering an object with the system.

Taking all the requirements into account, a typical user workflow should look like this:

1. Register an object: Provide keywords and images of the registered object.
2. Search for a registered object by using the previously defined keywords.
3. Display a list of results matching the query, and additional information for each found object.

3. Object Detection

This chapter discusses two common feature-based object detection methods. The object detection methods provide the technique to implement a markerless discovery of real-world entities. An entity is being detected by the way it looks, when it is placed within the camera's field of view.

All feature-based object detection methods, which are used and described in this thesis, follow the same basic principle. They divide an image into a set of unique points or regions called "interest points" [BTVG06] or "key points" [LCS11]. Those interest points are the most distinctive points in the image, such as corners or borders of objects [BTVG06]. A specific object detection method consists of a detector and a descriptor. To detect an object, we also need a matching-algorithm and a filtering-algorithm. Hence, the standard procedure is the following:

1. Detect interest points within an image.
2. Describe the found interest points uniquely.
3. Match a found set of interest points against another image's set of interest points.
4. Filter the found results to increase the probability of good matches.

The first step of each method is to find interest points using a *detector*. The quality of a detector is determined by the repeatability of finding an interest point under a different angle and by the robustness against image transformations or scaling. A perfect detector can find an interest point from each angle and even under different lightning conditions. The robust detection of distinct interest points is fundamental for object detection, because thereby a single interest point can be found accurately among a large number of interest points within an image [Low04].

Once the interest points have been detected, they have to be described and stored in a data structure. This is called a *feature descriptor*. Usually, interest points are stored within a vector, which holds all necessary information. This is why such a vector is also called feature vector [BTVG06]. A feature descriptor needs to be unique for each stored point to make an accurate matching of the found interest points possible. Also, the descriptor needs to use a fast data structure in order to being able to access the values quickly. Descriptors can also provide help with detecting errors or preventing noise while storing representations of interest points [BTVG06].

To compare two sets of interest points, a *matching-algorithm* has to be used. The matching-algorithm is not part of the used object detection method and can be chosen independently. A common matching algorithm is the KNN-matching-algorithm (described in Section 3.3).

Although a descriptor strives for the unique representation of two different interest points, it can never be guaranteed that two different interest points are represented the same. Those errors can be detected by using *filtering-algorithms*. The filtering-algorithms, which will be used in the prototype, are described in Section 3.4.

3.1. Speeded Up Robust Features (SURF)

Speeded Up Robust Features (SURF) is a scale- and rotation-invariant object detection method. SURF consists of a detector and a descriptor, which are described in the following sections. Both following sections are based on [BTVG06].

3.1.1. Detector

The detection of the interest points within the SURF method is based on the Hessian Matrix H. Therefore, the detector is called Fast-Hessian detector. The Hessian Matrix H in a point $\mathbf{x}$=(x,y) within an image I at scale σ is defined as follows:

$$H(\mathbf{x},\sigma) = \begin{pmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{pmatrix} \tag{3.1}$$

where $L_{xx}(\mathbf{x},\sigma)$ is defined as the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with the image I in point $\mathbf{x}$. The same is valid for $L_{xy}(\mathbf{x},\sigma)$ and $L_{yy}(\mathbf{x},\sigma)$ [BTVG06].

It uses the determinant of H for both locating and scaling the interest points. To simplify the calculation and making it faster (like it is being done in [Low04]), an approximation with box filters is being used to approximate the Gaussian second order partial derivatives. In Figure 3.1, an example of those box filters can be seen. The detection of the points can be seen as a pyramid, because the size of the box filter is increased step by step starting from a 9x9 filter and then adding 6 pixels at each new level.

In the SURF algorithm, the entities of the matrix are being approximated using a box filter. The approximations are named D_{xx},D_{xy} and D_{yy}. Leveraging the matrix's symmetry and using a weight of 0.9 for balancing issues (as suggested in [BTVG06]) the approximation of the Hessian determinant can be calculated as shown in Equation 3.2

$$det(H_{approx}) = D_{xx} \cdot D_{yy} - (0,9 \cdot D_{xy})^2 \tag{3.2}$$

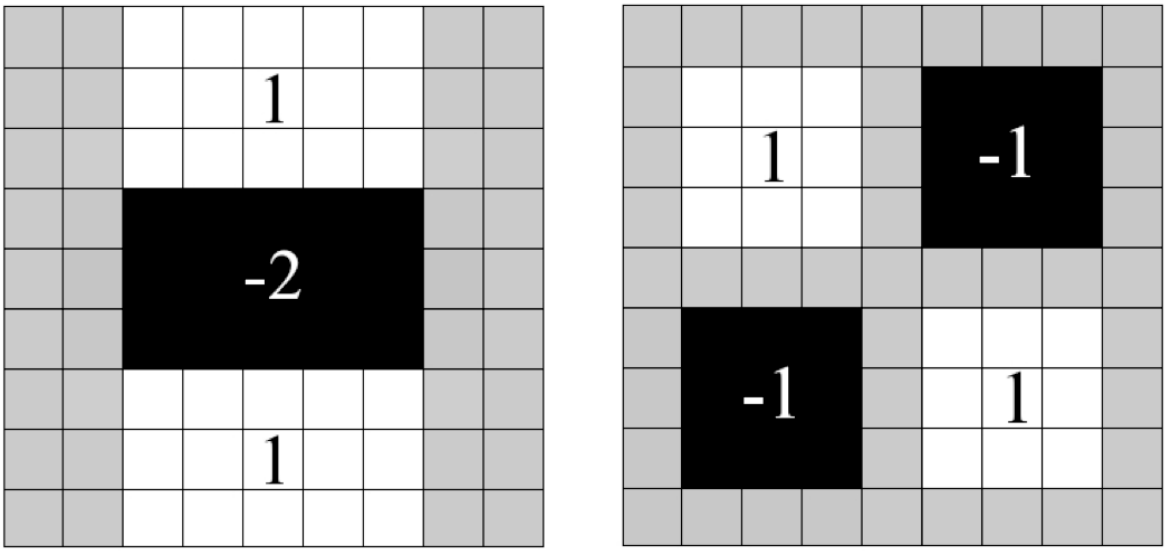


Figure 3.1.: Box filter used within SURF to approximate the Gaussian second order partial derivatives in y-direction (left) and in xy-direction (right). Source: [BTVG06]

3.1.2. Descriptor

The SURF-Descriptor introduces a way to reproducibly construct a vector, which can store information about a certain interest point. The construction of the descriptor consists of two parts: First, finding an orientation based on the information of a circular neighborhood around an interest point. After that, the descriptor is being built based on a square-shaped part of the interest point's neighborhood using the previously calculated orientation.

The orientation is calculated based on the Haar-wavelet[1] responses, which are collected within a circular neighborhood around the interest point. Those responses are collected in both x and y direction and are being weighted afterwards. The circular neighborhood is divided into parts with an angle of $\frac{\pi}{3}$. The sum of all responses is calculated for every part to estimate the dominant orientation. The longest vector gives its orientation to the interest point. With this orientation the SURF algorithm becomes rotation-invariant.

To describe an interest point, a square-shaped area is defined around it. This area is oriented according to the previously calculated orientation. The defined area is divided into 4x4 square-shaped sub-regions. SURF, again, uses the Haar-wavelet responses to collect information about the neighborhood. The Haar-wavelet responses are oriented in horizontal (d_x) and vertical (d_y) directions based on the previously calculated orientation of the interest point. For each sub-region a descriptor vector v (see Equation 3.3) is being calculated. A graphical

[1] A Haar-wavelet is a simple transformation, which is used in image compression. It was first proposed in 1909 by Alfred Haar within his PhD-thesis.

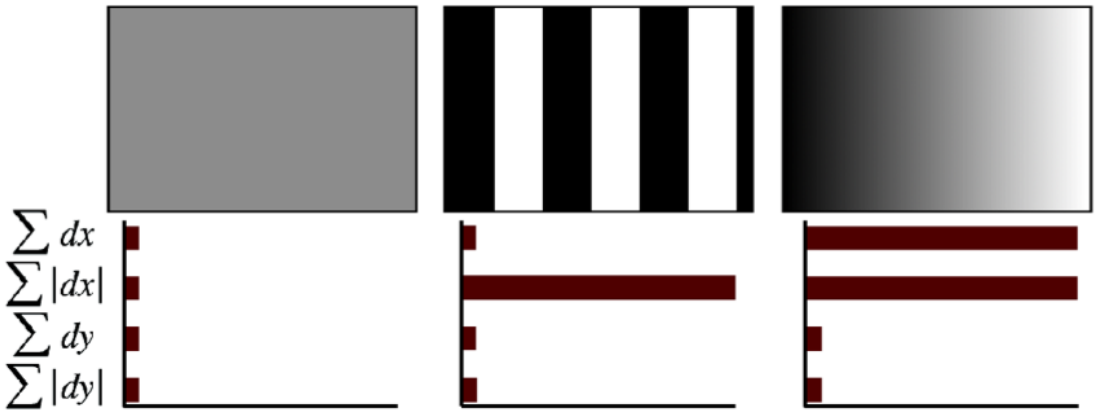


Figure 3.2.: A graphical representation of a sub-region's descriptor vector according to the sub-region's content. Left: The sub-region does not change - the vector's values are low. Middle: Changes in x direction influence the value of $\sum |d_x|$. Right: If the intensity in a horizontal direction is changing gradually, both $\sum |d_x|$ and $\sum d_x$ are high. Source: [BTVG06]

representation of a vector v can be seen in Figure 3.2. Because SURF uses 4x4 sub-regions, this results in a descriptor length of 64 values.

$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right) \tag{3.3}$$

3.2. Binary Robust Invariant Scalable Keypoints (BRISK)

Binary Robust Invariant Scalable Keypoints (BRISK) is an object detection method, which is scale- and rotation-invariant. Due to its binary nature, it promises to be orders of magnitude faster than SURF. The following subsections are based on [LCS11].

3.2.1. Detector

In order to detect an interest point within an image, BRISK uses a scale-space interest point detection algorithm. This algorithm approximates the real scale of an interest point from a scale-space pyramid, which is build within the algorithm. The scale-space pyramid consists of n octaves c_i and n intra-octaves d_i for $i = \{0, 1, ..., n-1\}$, where c_0 is the source image. The intra-octaves d_i are calculated by downscaling c_i by the factor of 1.5 and the next octave c_{i+1} is calculated by downscaling it by a factor of 2^{i+1} (see Figure 3.3).

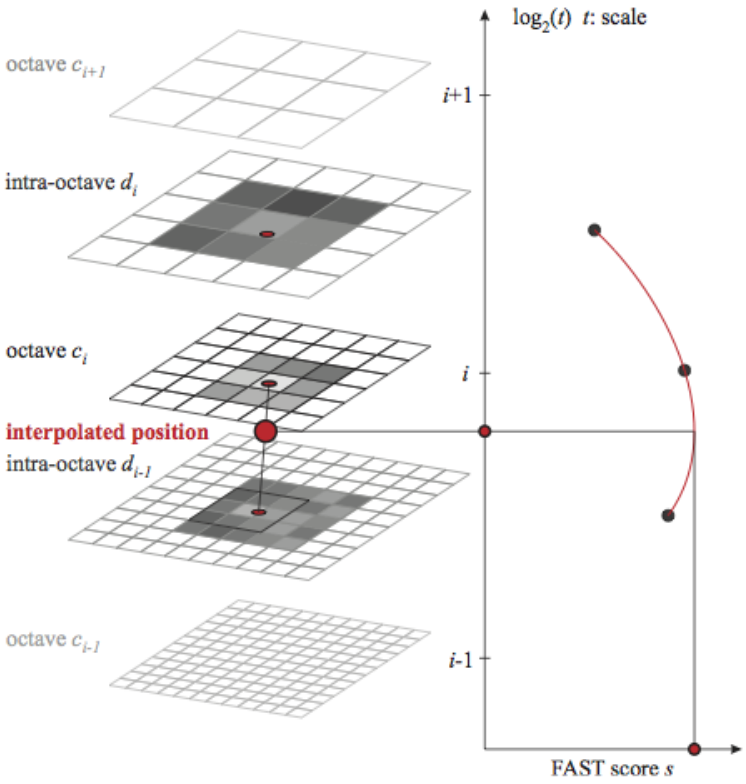


Figure 3.3.: Detection of interest points using octaves and neighbor intra-octaves. Source: [LCS11]

BRISK calculates FAST scores [RD06] to measure the uniqueness of an interest point candidate using the FAST 9-16 detector. This detector is used for each octave and intra-octave with the same threshold. The FAST score of the interest point candidate's area and its 8 neighbors needs to be higher than a threshold value which determines whether a candidate point is considered an interest point or not. In a second step, the same area's FAST scores of the corresponding intra-octave layers will need to be lower. When both conditions are true, the candidate point is considered an interest point. The areas that are surrounded by a border in each layer in Figure 3.3 are the neighbors which are used to calculate the interest point's FAST score.

3.2.2. Descriptor

The BRISK-Descriptor was built in order to be able to compare descriptors very fast with each other. Therefore, the BRISK-Descriptor is a binary string that holds brightness comparison information for a set of point-pairs belonging to an interest point. This mechanism was inspired by the BRIEF-Descriptor [CLSF10]. The matching can be done very fast, because the brightness comparison uses a bitwise XOR operation.

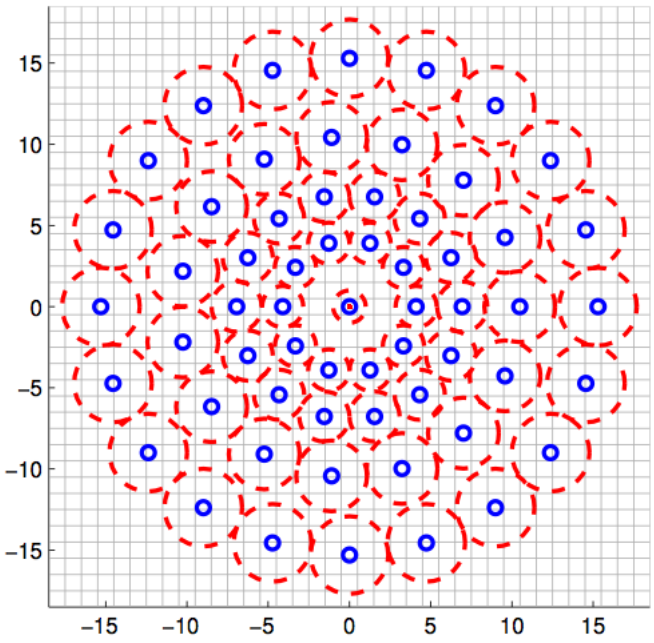


Figure 3.4.: The sampling pattern which is used to describe an interest point in BRISK. N=60 sampling points are being used around the center, where the interest point is located. Source: [LCS11]

To calculate the differences in brightness inside the area, BRISK uses N=60 points, which are located within a circle around the interest point (see Figure 3.4), and compares them with each other. Also a Gaussian smoothing is being applied to prevent aliasing[2]. The BRISK descriptor distinguishes between short- and long-distance pairs. Short-distance pairs have a maximum distance, which is allowed to be in-between. Long-distance pairs have a minimum distance. The long-distance pairs are used to calculate the orientation of the keypoint. In Equation 3.4 all gradients $g(p_i, p_j)$, where p_i and p_j are long distance pairs and L is the set of all calculated long-distance pairs, are being added to achieve rotation-invariance.

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{|L|} \sum_{p_i, p_j \in L} g(p_i, p_j) \tag{3.4}$$

After the interest point's orientation was calculated, all sampling points involved in the description of the interest point are rotated in order to be rotation invariant. Then, the short-distance pairs are being compared regarding their intensity. A bit-vector b with the length of 512 bit is being built from all point-pairs (p_i, p_j) within the short distance sampling

[2]Aliasing is an effect which can occur in signal processing, when a wrong sampling-rate is being applied to an analogue signal. The resulting digital signal of two different analogue signals may look the same.

circle. The value of a bit in b is 1 at the bits, where the intensity of p_j is greater than the intensity of p_i, and 0 otherwise.

3.3. KNN-Matching

BRISK and SURF return a number of descriptors which describe found interest points. In order to compare those descriptors to each other, a matching-algorithm is needed. In this research, the k nearest neighbor (KNN-) algorithm was chosen, as suggested in [Lag11]. The algorithm returns the k nearest neighbors for each given interest point. To match two sets of interest points against each other, the algorithm has to be called twice. First, to calculate the corresponding neighbors from image A to image B and vice versa. For the number of neighbors to be found $k = 2$ is being used as suggested in [Lag11]. This returns a set of at most two candidate points for each given point.

In this research, the OpenCV 2.1 implementation of the KNN-algorithm is being used. This implementation uses a brute-force matcher for both BRISK and SURF, which means that k nearest neighbors are found by trying all possibilities. This results in a runtime of $\mathcal{O}(n^2)$. There are approaches to reduce the runtime of the KNN-algorithm using randomized KD-trees as suggested in [SAH08].

3.4. Filtering

As mentioned before, the KNN-matching may return two corresponding candidate points per interest point. And of course, some false-positive detections could have occurred. Therefore, we need a filtering algorithm to filter outliners and incorrect candidate points. In this section a symmetrical check, a ratio check and the RANSAC-algorithm are introduced. Those algorithms can be used within a prototype in that order.

3.4.1. Ratio and Symmetrical Check

In a worst-case scenario, the KNN-matching will return twice the matches for both images than the number of interest points. Hence, we need to filter false matches from the found set of matches. Both ratio and symmetrical checks are based on [Lag11].

In the ratio check, the algorithm compares the distance of the two nearest neighbors for each interest point to each other. If an interest point has only one nearest neighbor, it is treated as an error and the point is being removed. The ratio check follows the assumption that nearest neighbors who are close together are most likely errors. The algorithm divides the distances of the candidate points to the interest point by each other and checks whether the ratio is under a certain threshold (within the prototype, the threshold is 0.65). If both nearest neighbors are sufficiently far away from each other, both candidate points pass the test.

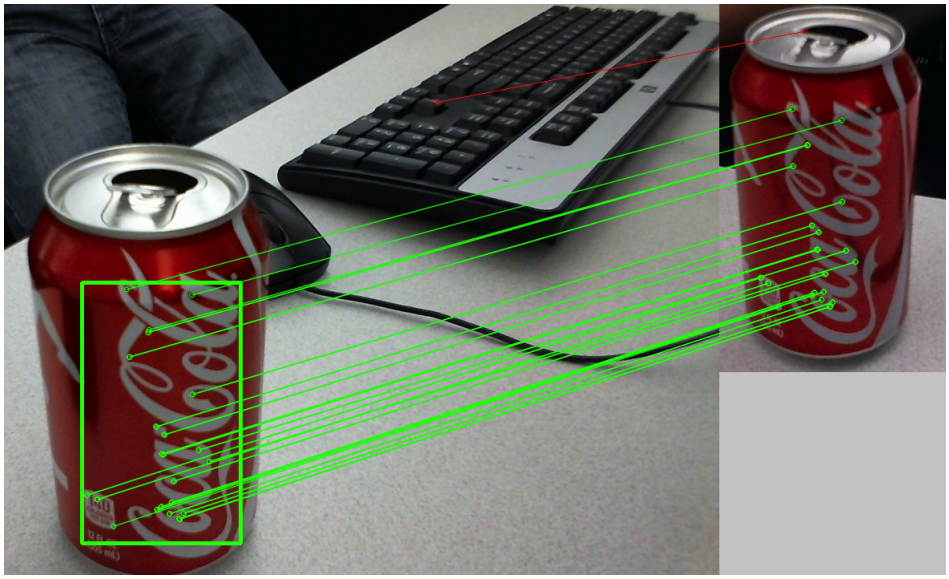


Figure 3.5.: A screenshot showing found interest points. The green lines are symmetrical correct matches. The red line is an interest point which failed the symmetry check.

The symmetrical check takes into account that there are two sets of points: the points from image A matched to image B (P_{AB}) and the points from image B matched to image A (P_{BA}). The algorithm now checks if each point in P_{AB} and in P_{BA} is the best matching candidate point for each other. For example, if there is a point $x_1 \in P_{AB}$ that is the best match for point $x_2 \in P_{BA}$, but x_2 is the best match for $x_3 \in P_{AB}$, x_1 will be deleted from P_{AB}. With this algorithm both P_{AB} and P_{BA} are symmetrical consistent with each other. This means, if a function F from $P_{AB} \rightarrow P_{BA}$ would exist, F would be bijective. Figure 3.5 shows an example of a pair of points that is not symmetrical correct. The red line depicts the false pair matching a key on the keyboard to the lid of the can. This error is being filtered out using the symmetrical check, because there exist better points for matching the lid of the can in the left picture than the key on the keyboard.

3.4.2. RANSAC

Random sample consensus (RANSAC) was first introduced by [FB81]. It is an algorithm to detect outliners within a set of data values. The algorithm consists of three steps: First, it randomly selects a subset of data values from all given values. Then, the algorithm assumes that all those values are correct and builds a model. Now, this model is being used to find all points, which are within the same error-tolerance than the model from all given values. This set is called the consensus set [FB81]. When the number of values within the consensus set is greater than a certain threshold, the consensus set's model is supposed to be valid for the

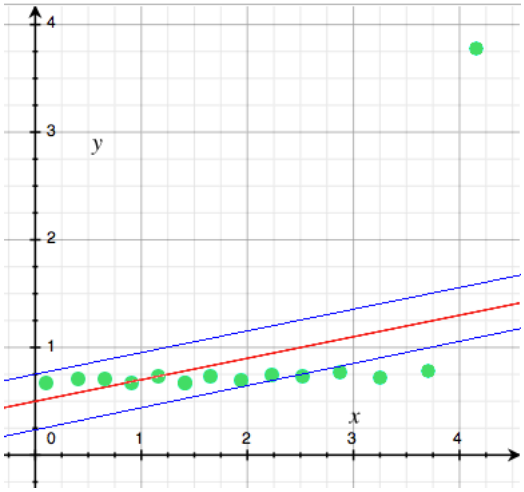


Figure 3.6.: A random selection of data values within the RANSAC-algorithm. The last value (green dot) is an outliner that distorts the model (red line) and causes it to reject some correct values.

given values. If not enough data values support the model the algorithm randomly selects a new subset of data values and tries again to build a valid model.

In Figure 3.6 a randomly selected subset is shown. The red line is the model that is calculated from this subset. The last value is an outliner distorting the model. The blue lines show the borders, which determine whether a point is supporting the model or not. The last three valid points are not supporting the model because the outliner has distorted the model. In this scenario, the threshold will not be reached, because too many points do not support the model. The algorithm will try again.

4. Antonius: A Search Engine for the Real World

This chapter introduces a prototype called Antonius[1] as a proof of concept for a stationary real-world search engine. The Antonius prototype was developed by the author during an internship at Yahoo!-Labs. It implements the key requirements of an object detection-based real-world search engine, which were introduced in Section 2.3.

4.1. Architecture and Component Overview

Antonius consists of 3 components: A frontend providing a GUI, a backend running the visual object detection and a shared database that connects them (see Figure 4.1).

The frontend component is an Apache Tomcat webserver (version 7.0.28) running Java Servlets to provide a web-based UI, which is platform independent and also available on mobile devices. Also, a web-based frontend can be accessed with multiple clients at the same time. This increases the flexibility while using the system. A user can access the frontend on a local desktop computer and at the same time on a mobile device, like a tablet. For generating and designing the GUI, Java Server Pages (JSP) is being used. Both servlets and JSPs are able to query the shared database in order to read information from it. Servlets are also capable of writing information to the database, e.g. when registering new objects.

The backend component is a console application written in C++, which runs in a progressive loop. In this loop, the backend iterates over all available images in the database and compares them to the live feed taken from the webcams, which are connected to the host machine. In

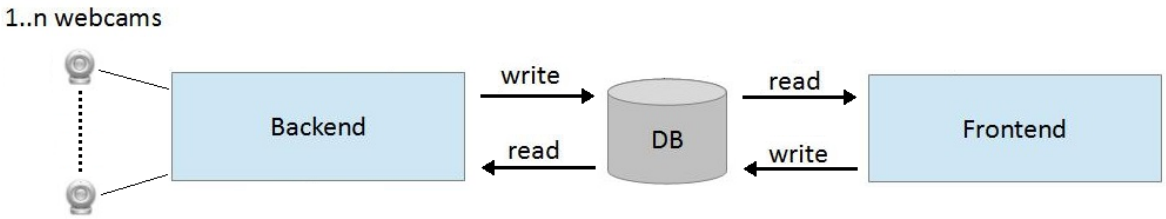


Figure 4.1.: A rough sketch of Antonius' architecture.

[1]The name Antonius was inspired by the historical figure St. Anthony, who is considered the "Patron Saint for lost items" in the Catholic faith.

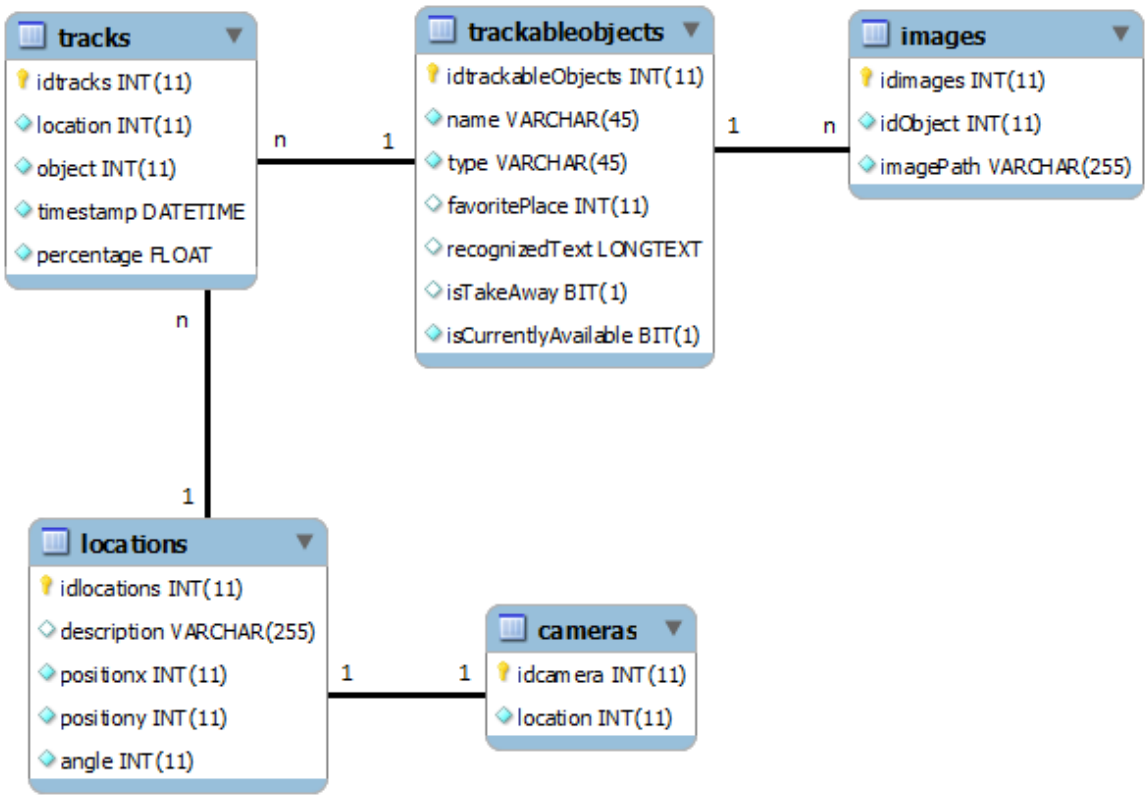


Figure 4.2.: The Entity-Relationship-Model of the shared database.

this prototype, the backend is only capable of comparing one image at a time to one frame of a live feed.

Although Antonius runs on a single host machine, it was developed with scalability in mind. Additional hosts, which run an instance of the backend component, can be added to increase the surveyed area. Due to the modular structure of the components, multiple frontends are possible as well.

4.2. Database

The Antonius prototype uses a MySQL Server to provide a shared database that is reachable via network in order to support distributed backend components. In this section, the database schema (see Figure 4.2) will be described.

All cameras, which are used by the system, have an entry in the *cameras* column. Every camera has one *location* associated with it. The location entry determines the position of the camera and the horizontal angle in which the camera is set up. The surveyed area is, therefore,

modeled using a grid layer, where every grid element within the area has a unique x/y position. The location entry also holds a textual description of the location, which is being shown to the user. A description text could be "left side of the desk" or "right side of the shelf". This description is there to help the user finding the objects faster because the user is familiar with a description like this and it would be hard for the user to map x/y coordinates to the surveyed area.

The table *trackableobjects* represents a real-world object, which is registered with the system. It stores the name, type, favorite place (i.e. where the object is most often seen) and the OCR-recognized text. It also holds flags to indicate whether an object is taken along with the user or not and a flag showing the current availability of an object within the surveyed environment. For each stored object, there are one or multiple entries in the *images* table. Multiple images per registered object can be useful for storing different reference images, e.g. images showing the object from different angles or under different lighting conditions.

Antonius also stores a record of where a real-world object has been seen, which is providing a history of an object's previous locations. This record is being stored in the table *tracks*, where each entry represents a match, which was detected by one of the connected cameras. In an ideal situation, such a track can be saved multiple times per second. To prevent the *tracks* table from getting jammed, Antonius implements a cleanup routine. It reduces the entries according to a threshold value every 24 hours. A suitable threshold could be, for example, 10 tracks per object. Then, the 10 newest tracks will be kept and all older tracks will be dropped. The dropped entries are analyzed to figure out, whether an object is being taken along with the user or not and if the object has a favorite location, where the object is usually stored. The flags in the table *trackableobjects* are being set accordingly.

4.3. User Interface and Map

With the frontend the user has the opportunity to register new objects with the system. Therefore, the user needs to provide a name, a type, and an image of the object. As an example, the registration of a Coke can is described. The name-parameter is the name of the object, in this case 'Coke'. The type-parameter is another searchable term, which should facilitate the task of searching for an object. Because with adding the type as a searchable term, the user has two searchable terms per object. In case the user forgets the exact name of an object, which can occur when having a large number of objects, the user is able to search for a more general term - the type of the object. In this example, the type would be 'can'. The first image of the object should face the front of the object, because it is being used to represent the object in the map of the surveyed area (see bottom part of Figure 4.3). After that, the system tries to recognize any text, which is written on the object, using the Tesseract OCR engine [Smi07]. After an object is registered with the system, the user has the opportunity to add additional images to the object. That can be useful to provide images facing the object from a different angle or showing the object under different lightning conditions.

The user interface extends the standard Yahoo! Search interface by adding another search-tab called 'Home' (see Figure 4.3). Because this prototype was built within an internship at

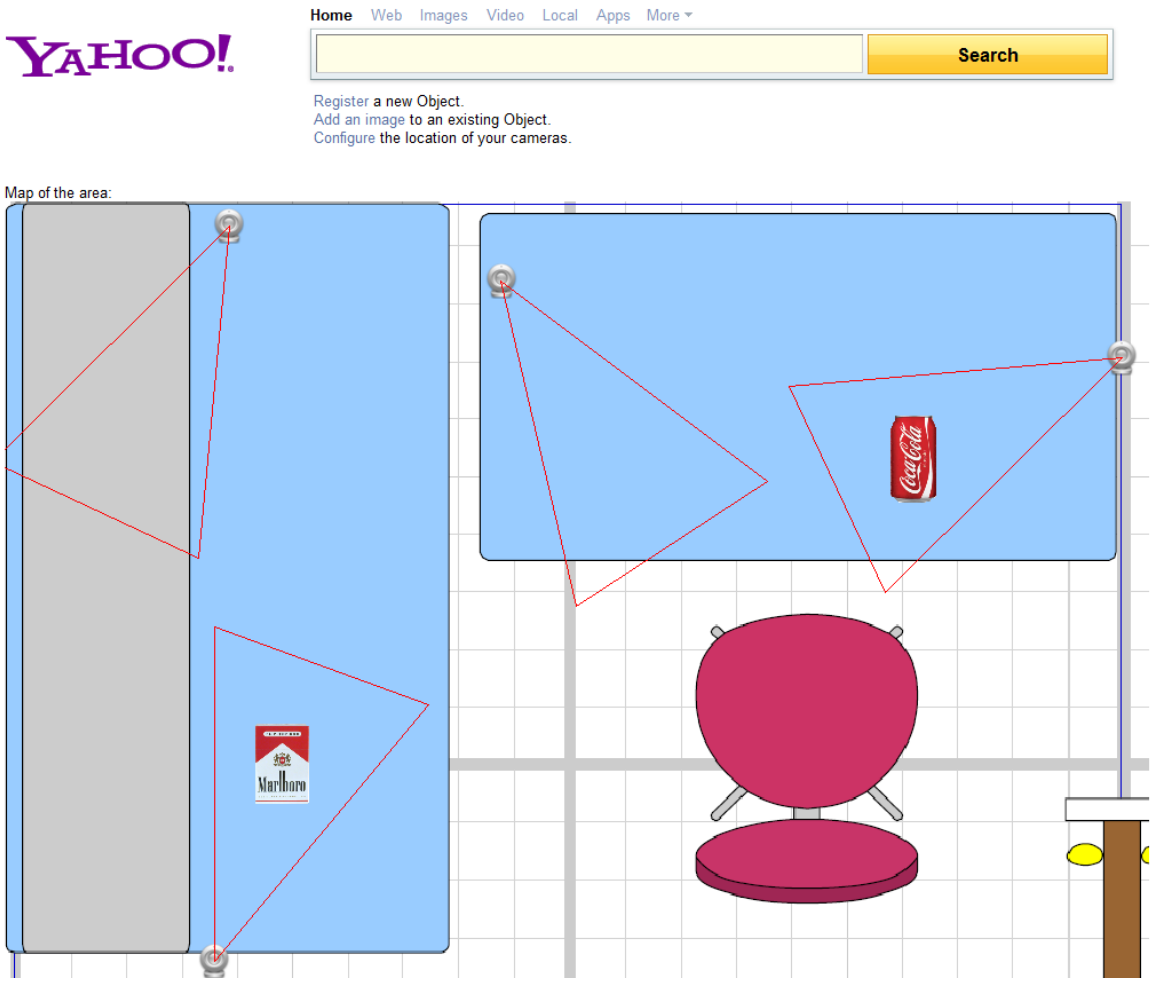
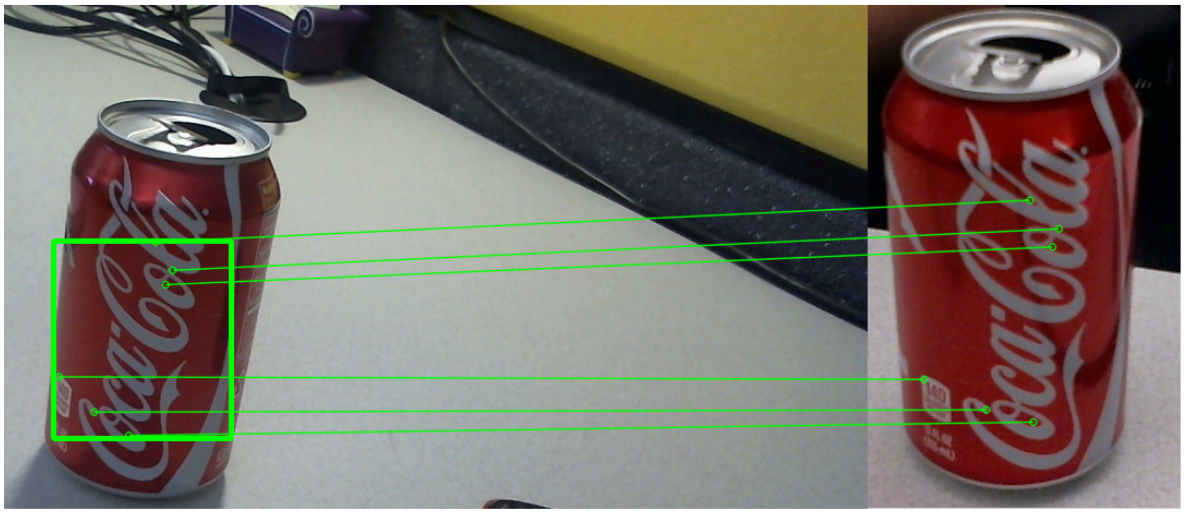


Figure 4.3.: The user interface with a map of the surveyed area within the Antonius prototype.

Yahoo!-Labs, the author chose to integrate the UI into the Yahoo! Search interface. This causes the Yahoo!-related user to feel more familiar with the UI. With the newly added 'Home' category, the user can search for the name, type or recognized text of a registered object. When a result is found, the system displays an overview of the object containing its current availability, a screenshot of the last detection, the last 10 locations, a last seen timestamp, and the current location (see Figure 4.4).

The second big part of the user interface is the map (see bottom part of Figure 4.3). The map shows a 2D representation of the surveyed area with all objects that are currently available within the area. The map was added to map the x/y coordinates of each object onto a visual representation of the area, which the user can understand easily. In this prototype, the map, the location and the angle of the cameras have to be provided manually. The backend estimates the distance from the camera to the object, based on the distance of the found feature points in the live image. This algorithm is described in the next section.

ID: 26
Name: Coke
Recognized Text: Coka Cola
Location: Right Table Right Side
Currently available: YES
Last Seen: 16:34:00 2012-08-13



Last 10 Tracks:

Location:	Timestamp:
Right Table Right Side	16:34:00 2012-08-13
Right Table Right Side	16:33:09 2012-08-13
Right Table Right Side	16:31:40 2012-08-13
Right Table Right Side	16:28:36 2012-08-13
Right Table Right Side	16:26:49 2012-08-13
Right Table Right Side	16:17:23 2012-08-13
Right Table Right Side	16:16:36 2012-08-13
Right Table Right Side	16:15:50 2012-08-13
Right Table Right Side	16:15:02 2012-08-13
Right Table Right Side	16:14:14 2012-08-13

Figure 4.4.: The results of a search show where the object currently is, a picture of its location, and a history of the last ten known positions.

4.4. Algorithms

The backend runs in a progressive loop, called the main loop (see Algorithm 4.1). The algorithm iterates over all objects that are registered with the database. Because the objects are queried from the database at the beginning of each loop, the algorithm can react to changes in the registered objects during runtime. Then, the available images per object are queried and it is being iterated over them. The algorithm takes each image and initially calculates the interest points from it. In later iterations, the previously calculated interest points are being used. After that, the loop iterates over all available cameras and grabs a live image. The algorithm takes this live image and matches it against the current object's features, which were calculated

Algorithm 4.1 The algorithm of the mainloop in pseudo code

```
procedure MAINLOOP
    for all obj ∈ trackableObjects do
        for all img ∈ obj do
            features ← EXTRACTFEATURES(img)
            for all camera ∈ connectedCameras do
                foundCount = 0
                for all i ∈ triesPerObject do
                    frame ← GETLIVEIMAGE(camera)
                    found ← IMAGEMATCHING(frame, features)
                    if found = true then
                        foundCount++
                    end if
                end for
                if foundCount > triesPerObject/2 then
                    WRITETRACKTODATABASE(obj,camera)
                    UPDATEOBJECTSTATUS(obj)
                end if
            end for
        end for
    end for
end procedure
```

before. This step is repeated a user-defined number of times (triesPerObject) to achieve a better success rate and to be more stable against falsely detecting an object. In this prototype the 'triesPerObject' was set to 3. When the matching is done, the algorithm checks if 50% of all tries found the object. If this is the case, the track is valid. It is being written into the database and the current object's status is set to "currently available".

If the system is constantly surveying the area, the tracks-table in the database can easily be jammed by too many entries. Therefore, Antonius implements an algorithm to periodically check whether some entries can be dropped or not. The algorithm (see Algorithm 4.2) is being run when an entry, which was recorded more than 24 hours ago, was found in the database. Then, all entries in the database belonging to a specific object are counted. If the amount of entries is greater than a defined threshold (e.g. 10) the tracks are being analyzed. The system checks if the object is always at the same place (has a favorite place, where the user puts it every time, for example when arriving at home) and if the object is used to being taken along with the user. Then, the unnecessary tracks are dropped from the database to keep it neat and fast.

To insert a thumbnail image into the 2D map, an x/y coordinate has to be calculated for each found object. Because the distance from the camera to the object may vary for each image, Antonius cannot calculate the exact distance. Therefore, the distance is estimated based on the found feature points in the live image. The system tries to draw a bounding box

Algorithm 4.2 The algorithm to keep the tracks-table clean

```
procedure CleanupDatabase
    if currentTime > oldestTrackInDatabase + 24h then
        for all track ∈ tracksInDatabase do
            if count(track) > tracksThreshold then
                CheckAndUpdateFavoritePosition(track)
                DeleteUnnecessaryTracks(track)
            end if
        end for
    end if
end procedure
```

(see Figure 4.4), which includes all found feature points, around the object. Then the system compares the size of the bounding box to the size of the reference picture in the database and calculates a percentage. If the bounding box has the same size than the reference image, the system assumes it is in the middle of the cameras field of view. If the bounding box is smaller than the reference image, the object is further away from the camera. If the object is very close to the camera, the system may only capture a part of the object resulting in a small bounding box. But this smaller bounding box only covers a part of the reference image and not the whole image. In that case, the system recognizes that the object has to be very close to the camera and calculates the coordinates accordingly. The camera's field of view was determined empirically. Here, the field of view is defined as the maximum distance, where it is still possible for the camera to recognize an object using visual object detection. Using the Logitech C525 HD webcam and the algorithms implemented in the Antonius prototype, the field of view is 58,4 cm. Also the maximum angle of sight was determined with an experiment. The angle was calculated placing an object at the most right place at the maximum distance within the field of view. Then, the object was moved towards the most left place along the field of view line. After that, the resulting angle was calculated using the most left and the most right place. In the experiment, this angle was 40°. As mentioned above, the positions of the cameras and their angles within the area have to be provided manually. Using this information and combining it with the calculated field of view and the angle of sight, the cameras and the found objects can be drawn into the map.

4.5. Frameworks

The Antonius prototype uses a lot of different frameworks and technologies, which are introduced and described in this section.

In order to efficiently recognize objects within live images, the Antonius prototype uses all algorithms, which were introduced in Chapter 3. The detection and matching in the backend is being done using OpenCV (Version 2.1) and BRISK/SURF (see Chapter 3). OpenCV[2] is an open source collection of Computer Vision algorithms. An implementation of the SURF algorithm is included within this library and was used for the prototype. The user can choose whether the SURF or the BRISK algorithm should be used for the object detection. The BRISK algorithm is too new to be included in OpenCV, but its authors provide a reference implementation of the algorithm[3], which can be integrated into OpenCV. The detection and description of the interest points is being done using either BRISK or SURF. After the detection, a KNN-matching is performed. Then, ratio and symmetrical filters are being applied to the result set and finally the RANSAC-algorithm eliminates the last remaining outliners. This chain of validation algorithms minimizes the chance of a falsely found match in case the object is not present. Because the backend is designed to run quietly, it only comes as a console application. But to be able to understand and debug the backend, the user needs to be able to see, what the program is currently doing. Therefore, the backend implements a debug-GUI, which can be switched on and off, using OpenCV's "highgui"-package. With this package, the current live-image and the current database object can be displayed to the user and even the found matches can be drawn into those images to make them visible.

The last undefined parameter within the detection process is the number of found interest points (k), which are necessary to consider an object as found. In empirical tests, the author found out that $k = 3$ and below is too error-prone against false positives. With $k = 7$ and above, it is hard to identify any object, because if the angle and the lighting conditions are not exactly the same as within the reference image, the average number of found interest points is below 7. Especially, when it comes to objects where only few unique points can be found by the detector. For example, when using an image of a black iPhone 4 photographed from the front side, finding sufficient interest points is hard. Within the Antonius prototype k is chosen to be 5, because it is enough to filter false positives and it is not too high to falsely reject found objects. To additionally decrease the error rate, the object detection algorithm iterates 3 times over each frame taken from the webcam. If 2 out of 3 times the object was detected, it is considered found.

Another important fact is that the user needs a desktop computer to run Antonius with more than 2 webcams. This is due to the throughput of a USB hub. Using a high-resolution webcam, a standard USB hub is only capable of transmitting the data of 2 webcams at the same time. If the system should use 4 webcams, like in the usecase scenario (see Section 4.6), the user has

[2] http://opencv.willowgarage.com/wiki/
[3] http://www.asl.ethz.ch/people/lestefan/personal/BRISK

to connect 2 webcams at the front USB hub of the desktop computer and 2 webcams at the back USB hub. Using this setup, the webcams don't jam the USB hub.

A problem that occurred while building the system is that the Logitech C525 HD webcam's driver is built for the use of a single camera only. When connecting 2 or more Logitech C525 HD webcams, the cameras will interfere with each other and none of the webcams will be recognized by the system anymore. To fix that problem, the Videoinput library[4] is being used. This library provides a wrapper code for the webcams which can query live images from more than one connected webcam using the DirectShow framework. To use this library, the Windows SDK, which contains the DirectShow framework, has to be installed on the machine running Antonius.

To connect the frontend and the backend, a MySQL Community Server (Version 5.1.66) is being used. Because this server is a standalone database, it can run on any host, which is connected to the frontend and the backend. The user has to provide an IP-address, a port, a username and a password to access the database. Because of this property, one can think of a big database, which is located in the Internet and is providing a big crowd-sourced set of known objects.

The frontend is an Apache Tomcat (version 7.0.28) webserver, which was integrated into the Eclipse IDE. The Tomcat webserver provides webpages to conduct the search and display the results. This design allows the user to access the system using different, and especially mobile, devices. The Antonius prototype is capable of OCR using the Tesseract OCR engine, which is delivered as a C++ library. As Antonius uses a Java/JSP frontend, a Java JNI wrapper called Tesjeract[5] has to be used. The OCR has to be done at the frontend, because this is the place where the images are being added to the system. Whenever the user uploads an image of an object, the OCR is applied to recognize text. If a text was found, the user can adjust the found text because there is a possibility of Tersseract recognizing wrong characters. After that, the recognized text is written into the database. This enables the user to search for the OCR-recognized text as well.

4.6. Usecase Scenario: Desk in an Office

The scenario, which was used to test and evaluate the system, is an office scenario. In this scenario, four webcams are being set up to survey the objects on two desks. The left side of Figure 4.5 shows an overview of the area and the cameras, as they are set up in a demo cubical. There are two desks inside the demo cubical. Each of the desks is equipped with two cameras to survey a large part of the desks. Each camera is put on top of a box to maximize the camera's area of sight (see right side of Figure 4.5). In this picture, the camera is able to track the green mug and the Coke can. If the camera would not be placed on top of a box, the nearest object would cover too much of the camera's field of view, which results in not being able to detect other objects in the area.

[4] http://www.muonics.net/school/spring05/videoInput/
[5] http://www.ohloh.net/p/tesjeract

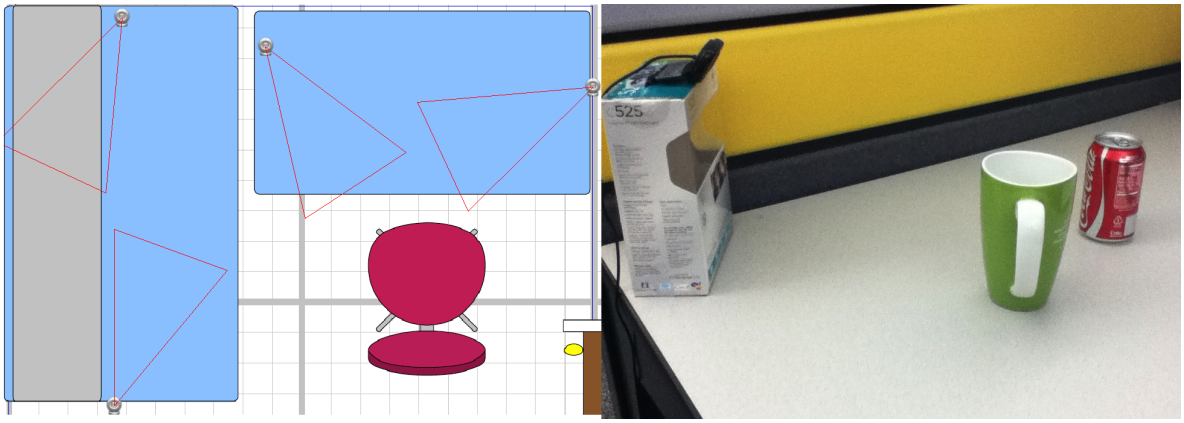


Figure 4.5.: The map of the office scenario (left) and one of the cameras used in this scenario (right).

A typical use case for this scenario is a lost object, for example a pen. If the pen was registered with the system, the user can search for it in case he cannot find it any more. In an office scenario, it occurs that a pen gets covered by a stack of papers. The Antonius system can tell the user the last known location of the pen. The user sees that this location is now covered by a stack of papers. Based on that information, the user now might remove the papers and find his pen. Another use case for this system is to take pictures of documents, e.g. a research-paper. Using OCR, the system is capable of searching for words within the paper. This means, if the user only remembers some keywords within a registered research-paper, the user can search for it, too.

There are objects within an office that always have the same location. For example, the stapler is always to the right of the monitor. Over the time, Antonius will learn that the stapler is always in that spot and can notify the user when the stapler is missing or was found in some other place. This way, it helps to keep the office area organized. Another important use case is that Antonius is capable of identifying objects, which the user is taking along with him when leaving the desk. Such an object could be a smartphone or a keycard. In a later version, the system could notify the user whenever he is leaving his cube and is not taking those objects with him.

4.7. Performance

The performance object detection algorithms within the Antonius system is tested using their speed and their recognition rate. The speed measures the time until an object is being detected. The recognition rate measures the fact that an object is detected when it is placed within the camera's field of view.

Resolution	SURF	BRISK
1280x720	2163.51	83.76
960x720	1540.99	76.16
640x480	701.70	49.43

Table 4.1.: Time in [ms] to do the extraction of the interest points.

Resolution	SURF	BRISK
1280x720	93.33%	80.00%
960x720	100.00%	93.33%
640x480	100.00%	80.00%

Table 4.2.: Percentage recognized matches. Object: Coke.

4.7.1. Speed

In order to maximize the performance of the system, the performance of both SURF and BRISK within the Antonius prototype was compared using different resolutions while extracting the interest points. Antonius uses Logitech C525 HD webcams to capture images. As a reference image, a picture of a Coke can with the resolution 283x460 was used. Table 4.1 shows that BRISK outperforms SURF by more than an order of magnitude at all chosen resolutions.

4.7.2. Recognition Rate

An experiment to measure the recognition rate of both SURF and BRISK was conducted. The experiment uses two different sample images to find the object within the sample image in a video stream. The first sample image is an image of a Coke can, facing the Coke logo from a straight viewpoint. The second sample image is an image of a cigarette box taken from a 45° angle, which the author assumes makes it harder to recognize. Both objects were placed 30 cm away from the camera and positioned in the middle of the camera's field of view. All experiments were conducted 15 times. The percentages that are shown in Table 4.2 and in Table 4.3 are the average percentages calculated from all 15 runs.

Table 4.2 shows that SURF has a higher recognition rate than BRISK when using a straight sample image. It can also be seen that the resolution 960x720 has the highest recognition rate comparing both algorithms with this sample image. Table 4.3 shows that both algorithms are able to detect an object from an image, which was taken from a 45° angle. But the recognition rate is lower compared to using a straight sample image. From this experiment can also be seen that BRISK is not suitable when using a 640x480 resolution combined with an angled sample image.

Resolution	SURF	BRISK
1280x720	86.67%	80.00%
960x720	80.00%	86.67%
640x480	60.00%	26.67%

Table 4.3.: Percentage recognized matches. Object: Box of Cigarettes.

5. Evaluation

In this chapter, the prototype is evaluated. The evaluation does not only focus on the prototype itself. It also focuses on the form of interaction and the concept behind an object-detection based real-world search engine. In the user study, the participants are being interviewed after using the system to get a more detailed idea of the participants' thoughts. This chapter describes the conducted user study and discusses its results.

5.1. User Study

A user study was conducted in order to evaluate the Antonius prototype and the method of implicitly interacting with ubiquitous cameras. The report was written as suggested in [FH03]. First, the apparatus and the procedure are being described and then the results are being shown.

5.1.1. Participants and Apparatus

A total of 8 users (5 male, 3 female, average age 26.25, SD=7.8) participated in the study. All participants were Yahoo! employees and interns, who have an IT-background and can be seen as experts. A mailing list was used to recruit the participants. All participants volunteered to take part in the study.

As a testing environment, the office scenario (see Section 4.6) was used. The scenario covered two desks using four Logitech C525 HD cameras. Both the Antonius frontend and backend were running on a single desktop computer. The participant was sitting in front of the desktop computer to be able to control it. The facilitator was sitting next to the participant, but not in reach of the desktop computer and not within the cameras' fields of view. Before the participant entered the scenario, the backend was started and running within the progressive loop. The frontend was opened in a web browser and the main page was opened.

5.1.2. Procedure

First, each participant was informed about the cause and the contents of the study. They were also told that all results would be published within this thesis and within a scientific paper. Once the participant agreed to those terms, the facilitator gave an introduction into the system and told the participant about the object detection-based approach for a real-world search engine which was implemented in the prototype. Prior to the study, each participant was

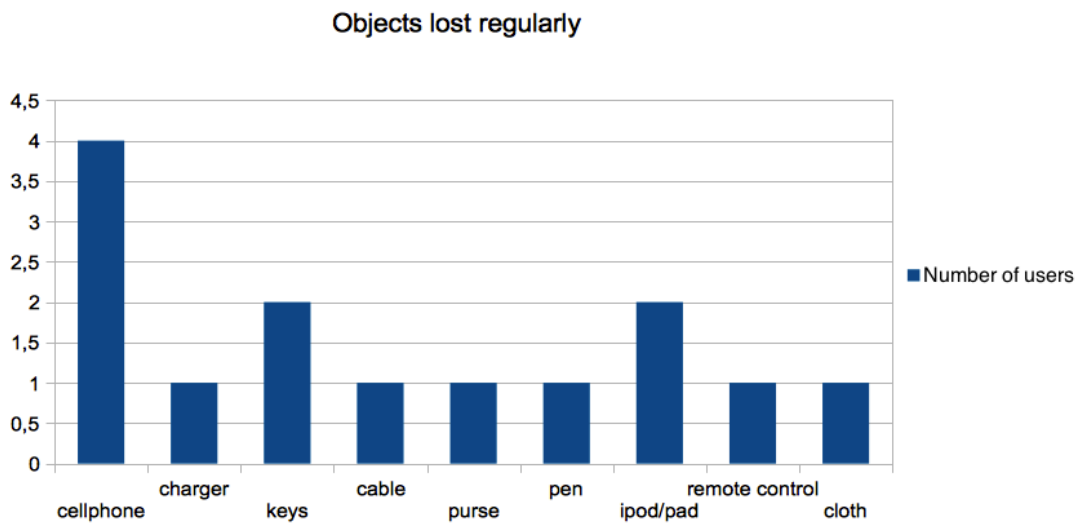


Figure 5.1.: An overview of the participants' lost objects.

asked to bring a personal object. The participant had to register the brought object with the Antonius system by taking a picture of the object and then submitting the picture to the server. Along with the picture, the participant had to provide two keywords: A name and a type of the object. Throughout this registering process, help was provided by the facilitator whenever it was needed. After the object was registered, it was placed somewhere within the surveyed area. Because the system needs some time to iterate over all cameras, a short questionnaire about the demographics was asked and the first interview was conducted (questions can be found in Appendix A.1). The first interview was about the habits of the participants concerning object placement within their living area and their daily search engine usage. After the interview was conducted, the system should have found the brought object and the participant is able to search for it using the web-based frontend. After that, the participants had some minutes to get to know the system and to play around with it. The participants were asked to alter the location of the object and search it again. During the whole process explanations were provided by the facilitator. When the participants were familiar with the system, a second interview was conducted. In this second interview, the participants were asked about their opinion on the system, the form of interaction, and their thoughts about using the system in their daily lives (questions can be found in Appendix A.1).

Prior to the study, the following hypotheses were assumed:

- A mobile system would be more convenient and will be accepted more likely.
- Objects that are taken along with the user are always stored at the same position.
- A graphical representation of the location is easier to understand than a textual representation.

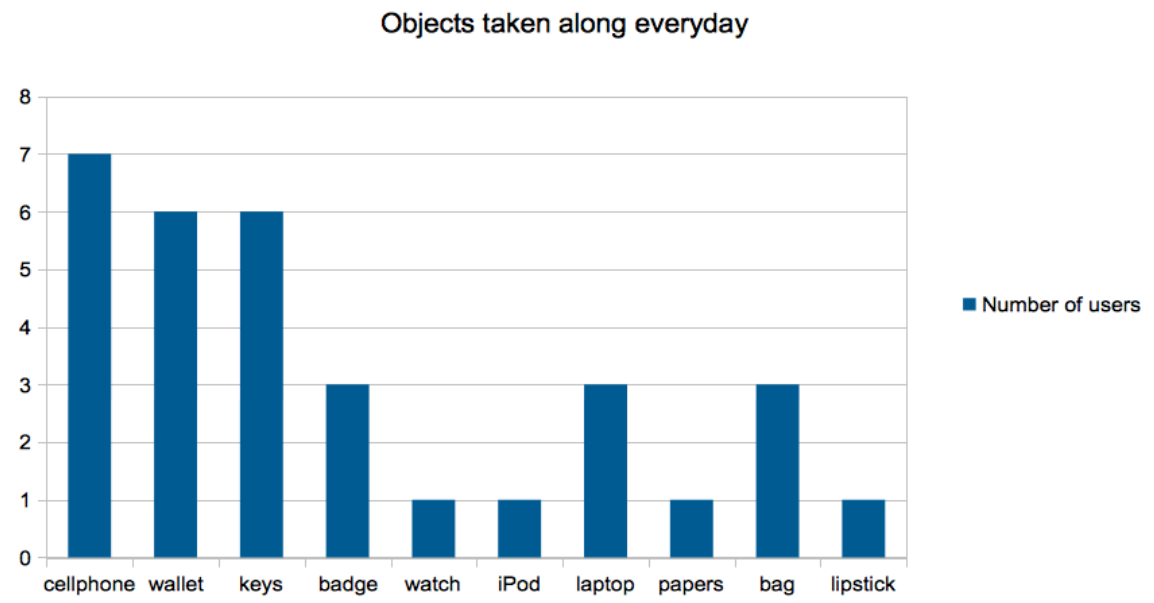


Figure 5.2.: An overview of the participants' objects which they take along with them everyday.

5.1.3. Results

Based on the first interview, the participants use a search engine like Google or Bing on average 17,63 times on a workday and 10,25 times on the weekends. 62,5% of the participants claimed to have trouble finding an object on a daily basis, whereas 25% of the participants stated their object loss rate to be pretty much never. One participant admitted to search for objects once a week. Overall, it takes the participants between 5 and 10 minutes to relocate their lost objects. When being asked for their method of finding lost objects, each participant mentioned the backtracking approach, where the participant tries to remember where he or she last saw the object and then revisits the places, he or she has been since. In case the participants lost their phone, they will call it in order to find it. The participants were also asked which objects they tend to lose regularly. In Figure 5.1 an overview of those objects is being shown. 50% of the participants lose their phone on a regular basis. The rest of the lost objects are rather small objects or other electronic devices. The participants were also asked which objects they take along with them every day. Figure 5.2 depicts that almost every participant is carrying a cellphone, a wallet and keys along with them. 37,75 % of the participants are also taking an employee badge and a bag with them every day. Then, the participants were asked whether they place those objects always at the same spot when they are at home or not. 87,5% stated that they place at least some of those objects at the same spot every day. 62,5% of the participants even place all objects which they take along with them every day at the same spot when they are at home. The participants, who store some or all of their take-along objects at the same place, where asked in which room they store those

objects. 85,71% answered that this place is the bedroom. The rest of those participants place those objects in the kitchen.

The second interview, which was a more detailed interview where the participants could talk about their feelings more freely, showed that there are several types of people when it comes to privacy and setting up the system in their living area. All interviews were recorded, transcribed and affinity diagrams were built from them. Figure 5.3 shows a combined affinity diagram from the best of all users' responses during the interviews. In the following, the responses are being grouped and described.

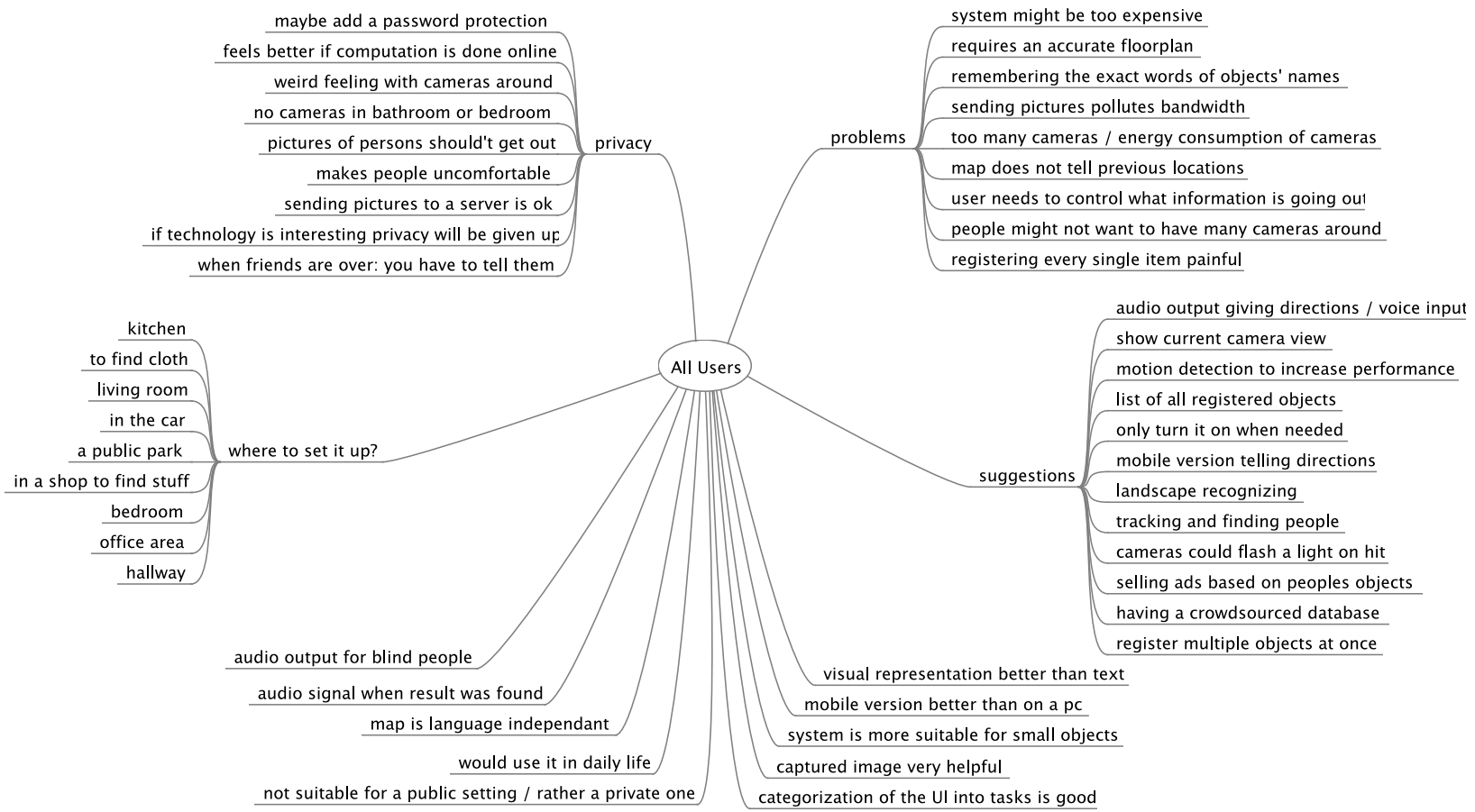


Figure 5.3.: Combined affinity diagram from all users' responses.

Additional Use Cases

After the users learned about the system in the user study they were asked which additional scenarios they can imagine for using the Antonius system. A suggestion was to use it as an inventory management system within an industrial environment. A company could install the system in a warehouse and keep track of items just by their visual appearance. This could supersede the barcode system, which is very common in warehouses. As a downside, objects that look the same could not be distinguished from each other. Many users also stated that they would use a system like this in their car. The participants tend to leave a lot of objects in their cars and later they don't remember where they put them. Another participant even thought in a bigger scale and suggested to install the system in a parking lot in order to help people finding their cars. The most common suggestion was to install the Antonius system in a person's kitchen. The system could keep track of all the ingredients available and even help the user in putting together a shopping list. The participants said that they spend a lot of time going through all the groceries to find items that they are out of.

Data Representation

The Antonius system is representing the results in a details-page in text form. This means that every object's location is described based on the room's properties and furniture. A location's description is for example: "Top shelf to the right" (see Figure 4.4 in the previous chapter). Additionally, Antonius offers a visual representation of the location of the objects that are currently available. This representation shows a map of the surveyed area and draws thumbnails of the objects into the map (see Figure 4.3 in the previous chapter). The participants were asked for their opinion on both representations of the location. 100% of the participants liked the graphical representation within the map a lot better. Some stated that they had to think a lot to transfer the textual representation to the layout of the room. Another participant was experiencing difficulties in understanding which piece of furniture in the surveyed area might be called the "shelf".

When being asked about the map, all participants said it was easier for them to understand where the object is located because they are already familiar with the room's floor plan. Another advantage of the graphical representation is that it is language independent. A non English-speaking user would have to ask what a "shelf" is. But with the graphical representation the user could figure it out by the room's layout. The details-page also provides an image of the found object. Some users mentioned that this image of the object was really helpful for them in order to find it, because there were other objects next to the sought object. They could use the neighboring objects to estimate the location. Also, there might be a big or shiny object next to it, which can be found very easily. This way, the user can estimate a rough area and can narrow down the search.

Upon being asked about other ways to represent the location, some users answered with audio output. Especially in stressful situations, an audio interface could be appropriate. A participant described a situation, where he is just about to leave the house and he cannot find his MP3-player. He could imagine asking the system: "Where is my MP3-player?" and the

system will answer: "In the living room on the table." An audio interface would also be great for blind people. A system like Antonius could see for them and facilitate their everyday lives. On the downside, an audio output would also have the problem, where a user is not being able to distinguish between two descriptions of objects, like in the textual representation.

Interaction with the system

Another goal of the user study was to learn about the participants' opinions on implicitly interacting with the system. When first being asked, the participants were a bit scared, because a system is interacting with them and their environment all the time. A user does not have to control the system because the system does its task on its own. The second new thing for the participants is the ability to search for real-world objects using a computer. A participant told that he used to separate between the real world and the digital world. With a system like Antonius his "digital world would merge with the real world". Another participant said that a system like this could change the way people behave. They might not put very much effort into organizing their real-life objects anymore because they are able to search for them again.

Towards a mobile system

When being asked about accessing the Antonius system via a mobile device like a tablet or a smartphone, all users liked the idea. 87,5% of the users would even like a system, which is accessible via a mobile device better than the one where the user needs a desktop computer. All participants could imagine a stressful situation in which it is too much effort to start a desktop computer, and it would be more convenient to just run an app. For example, when a user is short on time and just stopping by his apartment to grab a hat. If this user had the ability to run an app on his phone, he could save some time. He could even access the system via smartphone while not being at home and save even more time, because he will know the exact location of his hat even before he arrives at home. Another participant was thinking about the kitchen example, where she could access her inventory of ingredients while being at a super market. Another example showing the benefits of a mobile version is a situation, where the user is outside and is not sure whether he has lost an object or has just forgotten it at home. The user could use the Antonius system to clear his or her conscience, because the object was just forgotten at home and not lost.

Problems

During the user study, the users also mentioned some problems they see, when using the system. The following section describes those problems. The first and most obvious problem is that there are too many cameras involved in the process. A participant said, "If you need four cameras to survey a cubical - how many will I need for my apartment?". The cameras will pollute the living area and might create a weird atmosphere. Also, the many cameras will consume space e.g. on top of a table or on top of a shelf. When this many cameras run

constantly, the energy consumption will be very high and the system will be very expensive to run. Also, the cameras will cost a lot of money, which makes the system even more expensive. Another participant was worried that this many cameras need to be connected to a server and that there will be cable everywhere in her apartment.

The participants liked the idea of the map, which shows thumbnails of found objects drawn into a floor plan of the area. But to be able to draw a map of a user's apartment, the system needs an accurate floor plan in the first place. Therefore, providing the system with the necessary information could be cumbersome. Many participants suggested that the map should be the main way to access information. A problem with the map is that previous locations cannot be seen because only currently available objects are shown. Another participant would have liked a zoomable map (like Google Maps) instead of a static image. In this way, the user could see an object's location more clearly. Also the system would be more scalable, because when there are many objects, the map might get polluted.

Some participants were experiencing difficulties in remembering the exact name they gave the registered object. To facilitate the search of an object, the user can also search for the type of the object. But this might cause the same problem. For example, when a can of Coke was registered: Did the user define the type "can" or did he define the type "beverage"? The participants suggested to add a list that displays all registered objects to provide a mechanism for the user to choose which object he or she want's to search for.

When thinking about setting up the system locally, a participant was worried that he needs to register all objects he owns with the system. In case he uses an online server to do the matching, this problem would not occur. A possible solution would be, to deliver a predefined crowd-sourced database of common objects for local servers. This would help the user because he wouldn't need to take a picture for each object he wants to add.

Imagine a use case, where the objects are in a closed dark room, for example a fridge or a drawer, the cameras will have no chance to recognize any object at all. The areas would have to be equipped with a source of light to enable tracking in those rooms. Another idea for the system is using an online server to do the image matching. A participant had concerns about polluting the bandwidth when using such a system. A system that uses an online server needs to send images to the server periodically. Especially in an apartment complex, where there is a shared Internet connection, a system like Antonius could jam the bandwidth.

The problem which all of the users pointed out is the privacy issue. Some users feel weird when being surrounded by cameras all the time. A participant mentioned that due to the implicit interaction with the system, he does not feel in control anymore. He would like to have a mechanism to control which information is going out to the server and which is being kept inside of his home. He is afraid that a company might know all of the products the user has. A company could use this information to display advertisement tailored to the user. This might be good for the company, but according to this participant, people would be worried about this.

Suggestions

When the participants understood the system and its principles, because they had given the idea some thought by answering the questions about the system, they were asked if they have suggestions to improve the system. This section lists and describes those suggestions.

A participant complained about the registration process for objects and said that it is too much effort to register each single object. He suggested implementing a mechanism to register multiple objects at the same time. A user could take a picture of a set of objects and draw bounding boxes around the object he wants to describe. This will result in a lower resolution of the object's sample image but it would facilitate the process for the user. This is, because the participants see taking the image as the most cumbersome part of the registration process. When thinking about a mobile system, a participant suggested using Augmented Reality techniques to give directions towards the sought object to the user. A user could have an application running on a smartphone and the application could be using the camera feed to determine the location and the orientation of the user. With that information, the application could display directions (e.g. red arrows towards the sought object) into the camera feed as an alternative way of representing the object's location.

As mentioned before, the energy consumption of many cameras running all the time could be very high. With this in mind, a participant suggested to use motion detection as a trigger for the cameras. The cameras could go into a standby mode until a motion detection sensor registers activity in the surveyed area. This would require additional hardware sensors. As the image-matching increases the computational burden, the frequency of scheduling an image-matching task could be optimized, too. The cameras could also be used to do motion detection, but they have to be running to do this. When a camera detects movement in its surveyed area, it could schedule a new image-matching task. This would keep the computational burden of the server low and the algorithm would only be ran when it is needed.

Another participant suggested to track persons as well. This would result in a higher privacy infringement but the benefits the users are getting from it would be higher, too. With being able to track persons, the system would know about a person's location. But this is only the beginning. The system could see which objects the user is interacting with and could estimate the user's activity from this data. With this information the system could suggest objects that the user might need for his current activity and facilitate finding those objects. Over a period of time, the system could learn about the user's habits and could suggest objects to interact with, based on those habits. Another improvement for the system could be displaying a list of recently searched objects and a list of objects, which were recently used and have changed their position, respectively.

A few participants criticized the many cameras and suggested movable cameras to increase the cameras' area of sight. A camera, which is modified to be able to rotate, could enable a 360-degree view around its position. This could allow viewing the same object from different angles. The user could chose from many views to find the angle from which the user can recognize the location best. Some participants would also want a possibility to view the live

feed of a camera as some sort of security system. Because since they already have the cameras installed, they would also want to use them for those purposes.

5.2. Discussion

The results of the user study show that although the users feel weird when being on camera, they could imagine using a system like Antonius to facilitate their daily lives. When being asked about specifics of a potential system, the opinions vary. Some users would like to have an online computation of the results and others do not want pictures of their homes to get out to a server at all. But a local computation brings up trust issues, too. Either way, when using such a system, the users have to trust somebody with their pictures. Either they have to trust a company or they have to trust a roommate or partner.

Another main emphasis of the study was the representation of location data within a search engine. All users found the 2D-map representation of the location data to be more convenient and easier to understand than the textual representation. This supports the initial hypothesis. The author assumes that compared to mapping a 2D representation, the cognitive load is higher when having to map a textual description into a 3D representation of the testing area. Most users said that it was easier for them to understand the 2D map.

As a result of the study, all users would have liked a mobile access to the frontend better than always having to access the system using a desktop computer. This confirmed the hypothesis that a mobile access to the system would be more convenient and would enable a more flexible use of the system. A user also would have liked a mobile version to facilitate the registering of images. An app could combine the tasks of taking a picture, transferring it to the machine, uploading it to the server and describing the uploaded picture. Therefore, it makes the system more easy to use. When asking the participants about the usability and the grouping of the information, the participants found it distracting to leave the map view in order to see detailed information about a sought object. A possible solution would be to extend the map with layovers, which provide detailed information when the user is pointing the mouse over an object within the map.

The last hypothesis was that users always store objects that they are taking along with them every day at the same place when being at home. The study showed that 87,5% of the participants store at least one of those object always at the same location. This confirmed the hypothesis. This fact should be considered in future attempts, because those objects are the ones, which are searched for the most.

The results of the interview-based user study may be biased because all users have an IT background and may be more open towards new technology than the average user. In the future, more diverse participants should be recruited to take part in the study. Choosing an open-ended interview-based approach with fixed key-questions for the study was a good choice for evaluating a prototype, which introduces a new way of interacting with a computer. The participant can speak what's on his or her mind freely without having to fear restrictions. Compared to letting a participant complete a questionnaire with predefined answers, which

would be good for a statistical evaluation, an open-ended interview enables the facilitator to additionally find out why a participant thinks a certain way. If something is unclear, the facilitator can just ask to explain a thought more deeply. On the downside, with an interview-based study the facilitator can bias the results because he or she can lead the participant towards a desired answer by the way the questions are being asked. In contrast, this cannot happen in a questionnaire with predefined answers but this would hold back valuable additional thoughts about the system. Also, a facilitator can be influenced by his own interpretations of previous interviews. This could result in posing a follow-up question not as open as it used to be in previous interviews, but posing it into the direction of a previous interview's results.

Many users stated that there are too many cameras involved in the process and that this many cameras make them feel uncomfortable. With running many cameras other issues appear, for example a high energy-consumption or the high cost of buying the cameras. Furthermore, the approach of having stationary cameras everywhere the system should work is not particularly scalable. As a result, the idea of having just one mobile camera, which is attached to the user, came up. This idea resulted in another prototype, which was implemented and evaluated (see the following Chapter 6).

5.2.1. Limitations

A big issue concerning the Antonius prototype is the spatial limitation. In order to work, the system requires webcams in every spot it should cover. A Logitech C525 webcam has a field of view[1] of 58,4 cm. This means, if a user wants to cover a table with 1,2 m length and 80 cm width, he or she has to use two webcams. To be able to retrieve live images at all times, the cameras have to be running and connected to the system continuously. Due to the bandwidth limitations of a USB 2.0 bus, only two cameras can be connected to one bus at the same time. Using a desktop computer with a front-side and a back-side USB bus, this results in 4 cameras per computer. In order to integrate more than 4 cameras with the system, other computers have to be used in addition. Those computers can also be connected to the shared database and work independent from other backend entities.

Another limiting factor is the number of objects registered at the database. Each object adds another iteration to the main loop and increases the time, which the system needs to react upon changes to an object's location. This is the same for additional webcams. If there is only one webcam connected to a backend instance, the system will be much faster compared to working with four webcams. Adding parallel processing to the algorithm could solve this problem.

When two items look the same, the system cannot distinguish between them. For example, if there are two Coke cans within the testing area, the system is capable of recognizing both of them. But it is not capable of telling which one is can A and which one is can B. In order to do that, identically looking objects have to be marked visually [Kir95].

[1] The field of view is defined as the area in which it is possible for the BRISK/SURF algorithm using our parameters to detect a sought object.

The implementation of the measuring of the object's distance towards the camera could be more precise. The current algorithm only estimates the distance by comparing the size of the bounding box to the size of the reference image. An infrared distance measuring, like it is being used in the Microsoft Kinect, could be used in further systems.

Using visual object detection is always dependent on the scene's lighting conditions. When the room is darker than it was when the reference picture was taken, an object might not be recognized anymore. A user could add several reference pictures taken under different lighting conditions. But this will result in more effort for the user when registering an object and a longer runtime for the current algorithm.

In order to being able to recognize an object from any angle, several reference pictures have to be taken, too. For example, a rectangular box has to be photographed from each of its 6 sides. When the system knows how the object looks from each side, it is not important anymore which side is facing the camera.

5.3. Personas

The answers of the second interview from the previous section regarding privacy and places where to set the system up were gathered and grouped to build 3 different personas [LWJH06]. Those personas represent different user groups. Each group is represented by a fictional person, who is being described accurately to be able to understand the interior motives driving that person and possibly leveraging them in order to create a better system. In the following paragraphs, those personas are being introduced and described. The names of the persons are made up.

The first person is called Carsten. He is male, 30 years old and is working in finance. In his job, he is surrounded by computers and cutting edge technology. Therefore, Carsten does not have a problem with being on camera all the time. Also, he thinks that he appears to be perfect from the outside and has no problem with him being recorded constantly. Carsten believes that interesting technology like the Antonius prototype will encourage people to give up aspects of their privacy for the benefit they get from using it. He would definitely prefer a cloud-based server to a local server to do the image matching. According to his opinion, a cloud-based server has more advantages compared to a local one. Besides, Carsten thinks a local server also has security issues. For example, when another person checks the local recordings and discovers Carstens little white lies. It would be the worst thing, if a friend of him would be able to see that he didn't have to work in the evenings but he was just too lazy to go out. Therefore, he prefers a cloud-based server and is totally fine with pictures of him being sent to the Internet as long as his face is blurred out. But when it comes to track persons instead of objects, even Carsten disagrees. "You have to draw the line somewhere.", he says and smiles. A person like him would set up Antonius in every room of his house. Especially bathroom and bedroom usage is fine, because once he owns the system, he wants to use it for everything. Even for little things, like tracking his toothpaste in the bathroom.

John is a male, 45 years old IT-professional. He is spending much of his free time on the Internet and the rest of his time with his family. As he is working in a technology-related environment, John was glad to test the Antonius prototype. He believes that in 20 years almost everybody will have a system like this in their homes and that people will get used to this technology. John sees himself as an early adopter and is perfectly fine with having cameras in his house. He want's to use the system only within a local environment, which means that no data will go outside to a company. Because of his experience in working in IT, he knows what can be done with a huge amount of personalized data. Therefore, he does not want a company to have data about his home. Unless a big company clearly states, what they are allowed to do with that data and until when they keep the data, John will not use an online server. When being asked about his opinion on tracking people within a home, John was excited. He could imagine tracking the location of his children to always be sure where they are, but only if the tracking would be computed on a local server. He does not want any information to leave his house. Even blurring the faces would not be enough because potential attackers could still figure out how many people there are in the house. John would set up the system in places he is at the most. Those places are his car, the kitchen and the living room. He would think twice about installing cameras in a more private room like the bedroom or the bathroom to assure each family member's privacy.

The last person is called Jessica. She is 25 years old and is currently studying business economics. Her hobbies are going out and trying the latest fashion trends. When Jessica first interacted with the Antonius prototype, she had a weird feeling because she was constantly on camera. She is not sure whether she wants to set the system up in her home or not because being constantly surrounded by cameras makes her feel uncomfortable. If she had to set it up somewhere, it would be in less private places outside of the house, for example in the garage or in the car. Jessica is absolutely uncomfortable with setting up the system in her bathroom or bedroom. She would not want a system to take pictures of her in those rooms at all. When she was asked to imagine, what needs to be improved in order for her to use the system, she answered that she needs to be in full control over the system. This means, there has to be a switch to turn off the system whenever Jessica wants to have some privacy. Jessica also would rather use an online system than a local system because of the benefits she gets from using an online system. "It is weird enough to be on camera all the time, why should I not take all the benefits I can get?", she said. Her biggest fear concerning a local system is, trusting the person she is living with. Her boyfriend could access the local server in order to spy on her and find out, what she was doing the whole day. An online system would make her feel more comfortable in that issue. Nevertheless, she has to be sure that no pictures of persons go out to a server and that all pictures, which are being sent to the server, are being deleted immediately. Her opinion on the system is that no matter how big the privacy issue is, people still will use it in the end. The situation Jessica can imagine to use the system in her daily life is to find cloth in her apartment.

6. Mobile Antonius: Towards Pervasive Search

This chapter introduces a mobile object detection-based real-world search engine as a result of the previous stationary system's evaluation. First, related mobile systems are introduced and described, and then a prototype called mobile Antonius is described and implemented. The reminder of this chapter evaluates and discusses the prototype.

6.1. Related Work: Mobile Systems

This section enhances the previously introduced related work (see Chapter 2) by other mobile systems. All introduced systems are capable of detecting an object and locating it within a scene. They are designed to keep track of objects, and therefore making activities and changes in location visible to the computer [FPR05].

In [Kir95] it is discussed that human beings group items to being able to remember their location and to work with them in a more organized way. This observation could lead to the conclusion that objects, which are grouped together for certain tasks, are most likely next to each other. For example, if a system is only able to detect one of the objects belonging to the same task, other objects belonging to that task have a high probability to be in that area, too. The author divides the structuring of space into three task-depending categories. Long-term structuring is used to store tools, which are needed for a task near the place where the task is being performed. For example storing all the cooking utilities in the kitchen. A medium-term task is, for example storing all dirty dishes inside the sink to remind oneself that those dishes should be cleaned soon. An example for a short-term task is placing all ingredients to make a sandwich onto the worktop in order to have them in reach when needed. The area is divided into regions, which are being categorized. An area could be on top of a table within a surveyed area or inside a certain drawer. A real-world search engine could make use of this fact and first search in areas related to the (long- medium- or short-term) task, which is currently being performed.

In [Ped01] it is assumed that objects don't move by themselves. Every time an object changes its location, a user is involved in the process. This is why it is only necessary to equip the user instead of the whole testing area. The system MagicTouch [Ped01] equips the user with an RFID tag reader, which is placed directly onto the user's hand. Also, a location transmitter is put onto the user's hand. RFID tags are put onto searchable objects. When a user grabs an object, the reader on the user's hand is reading the RFID tag of the object. As soon as the user drops the object, the RFID tag is out of reach and the system updates the object's location to the current location of the user.

Also in [SGM00], a wearable RFID reader is built. The reader is integrated into working gloves in order to recognize objects, which the user is interacting with. The system uses passive RFID tags, which do not need a battery. Therefore, the tags only have to be put onto the objects once and do not have to be maintained. The authors introduce a way to enable implicit human computer interaction, which means that the computer recognizes every day behavior of the users and uses it as an input. As a use case, the system opens a URL according to the object the user is currently interacting with. This could be a website about cooking when holding a pan or opening a blank document in a text editing software when holding a pen. The system does not track the location of the object - It just detects the object that the user is interacting with.

In [FPR05] a wearable glove called iGlove, which is equipped with an RFID reader similar to [SGM00], is built. iGlove is a modified bicycle glove which does not restrict the user as much as the working glove used in [SGM00]. Another improvement is that the iGlove gives the user more freedom because it is wireless. It uses passive RFID tags instead of active ones because the passive tags do not need any power supply. On the other hand, the passive tags have a limited range and the reader needs to be aligned to the tag in a special way. But the passive tags are the better choice because of their low cost and not needing a power supply. As a result of their research, the authors introduce another wearable prototype called the iBracelet. The RFID reader is integrated into a bracelet, which is worn by the user. Using a bracelet instead of gloves is more convenient for the user because wearing gloves causes sweat and gloves interfere with the user's tasks. The iBracelet has a range of 11 cm in which it can detect objects. When an object is bigger than 11 cm the user has to take care to get in range of the RFID tag. When it comes to activity detection, a user could also touch or get within the reach of an object by accident and a false activity could be recognized. But since this thesis focuses on locating an object, this is not a problem. Neither the iGlove, nor the iBracelet implements a mechanism to track and store the location of the used objects.

Another RFID-based object detection system called BodyNets is introduced in [PMB11]. Compared to the previous systems, BodyNets introduces a zone-based localization approach, where there are RFID antennas in each zone. When an RFID tag is recognized by an antenna, the system knows in which zone the object is currently at. The BodyNets system's focus is to detect users' activities, based on the objects they are interacting with and the order they are using them. The use case of a medical setting where the doctors are treating a trauma patient is introduced. Also, the system should detect objects which are carried around but not being used, e.g. a doctor's stethoscope. This system uses the idea of dividing objects into different categories to make assumptions about them. As a result, it is being distinguished between objects that are carried around by the user and objects that are only used for a certain task.

The SearchLight project [BSS04] is an approach to visually track objects using optical markers, which can be recognized by an Augmented Reality toolkit. The testing area of SearchLight is equipped with a camera and a projector in the middle of the room. The camera first scans the environment and tries to recognize each marker inside the room. When a marker is found, the camera stores its tilt and pan angle in a database. When a user wants to look up a sought object, the system queries the tilt and pan angle from the database and applies it to the projector which highlights the sought object, and therefore facilitates finding it. The

authors of SearchLight describe the initial scan process to take about 1 hour. This can be explained because they used USB 1.1 technology and the research is from 2004. The system also implements movement detection to recognize when parts of the surveyed area have changed. Using this technology, the system does not need to scan the whole area but only needs to scan the part of the area, where the movement was detected. Nevertheless, the system requires each searchable object to be equipped with a marker and each room needs to have a camera and a projector. A benefit of SearchLight is that no 3D model of the area has to be created because the system only uses the camera's pan and tilt angle.

Another more scalable approach to track the indoor position of a user is using optical markers to determine the room in which the user is currently at. In [RS03], a few rooms are being equipped with room-markers. A user is wearing a helmet, which is equipped with a camera. Whenever the camera recognizes a marker, the system knows to which room the marker belongs. Therefore, it knows the user's position. The user is also wearing a HMD (Head Mounted Display), which is able to display information directly into the user's field of view. Using a HMD, an Augmented Reality experience is achieved. Their system distinguishes between room markers, which are used for determining the user's location, and object markers, which are used for identifying objects. Two applications, which use this system, are being introduced. An indoor navigation system called Signpost, which displays the shortest directions towards a destination, and a library information system called ARLib, which displays information about books that are inside the user's field of view. With this technology, the location of an object can be narrowed down to a room. When placing more markers in each room, e.g. on each shelf or table, the system could determine an object's location even more accurately. Compared to SearchLight, this approach is more flexible in dealing with changes in an object's location. SearchLight would have to rescan the rough area trying to search for the marker, while ARLib just has to identify a marker within the user's field of view. The time to initially setup the system is about the same for both systems because a user has to place markers on each object, which is to be tracked, in both cases.

One can imagine that besides a visual detection of objects or markers, a visual detection system can also be capable of recognizing a scene. In [TMFR03] a nameless mobile system, which can recognize previously recorded scenes and objects within those scenes, is introduced. For example, when a user carries the system into the kitchen of an office building, the system compares the visual features of the kitchen to the sample pictures, which were provided previously. When it recognized a scene, the system knows where it is and starts to detect objects within the scene. For example, if the system knows how the coffee machine looks like, it can detect it and combine it with the previously recognized locations. The system now knows the object's location. In contrast to SearchLight and ARLib, this system does not need any markers to work. The effort a user has to expend to initially setup the system is about the same. Because with ARLib, the user has to put markers into each room that is to be recognized and has to register them with the system. While in this system, the user has to take pictures of each room to be recognized and then tell the system which picture belongs to which room.

In [CGKM07], a system is introduced which tracks objects and determines the objects location visually without the use of markers. It uses the SLAM (Simultaneous Localization and

Mapping) system [Dav03] to build a 3D map of an area using the feature points taken from a video stream. SLAM starts with a few previously known features and constantly detects new features in the neighborhood of the known features. When the camera moves, it tries to detect known features again to estimate the new direction and position of the camera. In the meantime, it builds a 3D map of the area, where the features have a location within the map. This SLAM 3D map is being combined with SIFT [Low04] object detection in order to detect objects visually. The system uses the SLAM scene detection to recognize the location and uses the SIFT features to compare it to previously recorded sample images of those objects. When an object was found, the location is written into the SLAM map.

Compared to SearchLight and ARLib, this system is the most promising one, as it does not need markers to recognize objects. While ARLib uses markers to identify the location, SLAM uses the visual features of the environment to recognize the location, similar to the nameless system introduced in [TMFR03], which is described in the previous paragraph. Compared to this previous system, SLAM is more convenient because the user does not have to provide pictures of the scenes to be recognized as the system creates a map at runtime. Thus, it promises to be easier to setup and more convenient to use. On the downside, to track an unknown part of the room, the camera has to be moved very carefully because the SLAM system needs to find known feature points in order to connect the new parts of the map to the previously known ones. Also, the system has to run constantly in order to have information about the location. Compared to the previous system, where the user can enter information describing a scene along with just a picture of the scene, here the user would have to specify the exact position of the scene within the map in order to provide additional information.

6.2. Concept

As discussed in Section 6.1, objects are usually moved when a user is involved in the process [Ped01]. They do not move on their own. Therefore, it might be enough to just instrument the user instead of the whole area. This concept aims at equipping the user with a wearable web camera in order to find out which objects he or she is interacting with. Just like in the stationary case, the system should be able to compare images taken from a live video feed to a set of registered images. This approach also does not require equipping sought objects with markers. Whenever an object is recognized by the system, the object's position is being saved. Because the system is mobile, there has to be a mechanism to detect the camera's position and orientation within a testing area. Based on the camera's position and the distance of the object towards the camera, the position of the object can be calculated.

The evaluation of the stationary system showed that a graphical representation of the location is better and easier to understand than a textual one. Therefore, a graphical representation should be built to support this prototype. A web-based search engine, like it is implemented in the Antonius prototype, is not required. Compared to a stationary system like Antonius, a mobile system will be cheaper because the system only needs one sensor to work properly - the one carried by the user. Imagine, the stationary system should cover an apartment. Assuming five rooms and ten cameras per room, this results in 50 cameras within the apartment. This

would be too expensive for a regular household to expense. In the mobile system, the number of cameras is being reduced to one, which is much cheaper. Also, as a result of the stationary system's evaluation, the users feel weird when being surrounded by this many cameras. In this mobile system, the user is wearing the camera. It might not feel unpleasant for the user because he or she might not be aware of the camera anymore. And the fact that it is just one camera, compared to 50 cameras, might feel less weird. This idea only works with the assumption that all persons who can move objects in the testing environment will carry a sensor. Otherwise a change of an object's position will not be recognized by the system. This could result in false location information because the system will still assume the old position.

6.3. Prototype: Mobile Antonius

This section introduces a prototype called mobile Antonius and describes its architecture and the used technology.

The general idea is to have a mobile object detection-based real-world search engine, which can be worn by the user. To build this system, a means to determine the position of the mobile camera within a room is needed. To accomplish this, the OptiTrack hardware and its Tracking Tools software are being used. OptiTrack is a motion-capturing and object-tracking system consisting of 6 infrared-based cameras, which are installed firmly at a room's ceiling. It can determine the exact position and orientation of a 3-dimensional marker within a testing area within an accuracy of millimeters. To combine the marker, the camera and the user, both camera and marker are glued to a hat (see Figure 6.1). The used camera is not wireless. Therefore, the cable is lead towards the back of the hat in order to not distract the user. A 5-meter USB extension cable is used to guarantee the mobility of the user within the testing area. The live feed taken from the camera and the position taken from the OptiTrack system are combined by the mobile Antonius software and a graphical representation is calculated from this data. Although the system becomes dependent on the OptiTrack system, which consists of even more cameras than the stationary system, an approach towards a mobile object detection-based real-world search engine is built.

6.3.1. Architecture and Component Overview

To implement the concept of a mobile object detection-based real-world search engine, a two-tiered architecture was designed. The first tier, which is called the backend, does the object detection and tracks the position of the user's head (see left side of Figure 6.2). The second tier, which is called the frontend, does the graphical representation of the data (see right side of Figure 6.2). Both tiers are connected through an UDP-streaming API called EIToolkit [Hol07].

The backend component of mobile Antonius runs an altered version of the main loop of the stationary system. It is capable of a fixed number of pre-defined objects, which was chosen to be three in the built prototype. The main loop iterates over those objects and compares their features to the features of the image taken from the live video stream. To detect and extract

Figure 6.1.: The mobile Antonius prototype - a combination of a webcam, a marker and the user.

the features of an image, the mobile Antonius prototype runs the BRISK object detection algorithm. The BRISK-algorithm was chosen because it performed best in the stationary prototype. A delay could be critical in a mobile scenario because the user could look at an object for just a moment. If that moment is used to calculate the features of a previously recorded image, the object may not be detected. Mobile Antonius also implements a callback for the OptiTrack-system. Its Tracking Tools software provides streaming of a marker's position in real-time. Every time the system receives a new location, a synchronized singleton instance is being updated to hold the newest location. When a match was found in the live stream, the algorithm gets the most recent location from the singleton instance and combines it with the ID-property of the found object. Both values are encoded into an EIToolkit packet and are sent to an instance of the frontend.

The frontend is running a 3D-representation of the testing area using the OGRE 3D graphics engine. It uses the ID, location and orientation of the marker, which were sent to the frontend from the backend via EIToolkit packets. The ID tells to which object the received location belongs. The object is being moved to the new location within the graphical representation. It stays there until a new location is received. The OGRE 3D engine is also capable of setting an orientation of the object. Mobile Antonius estimates the location of the object to be 40 cm in front and 45 cm below the marker's position (the values were determined empirically). This is

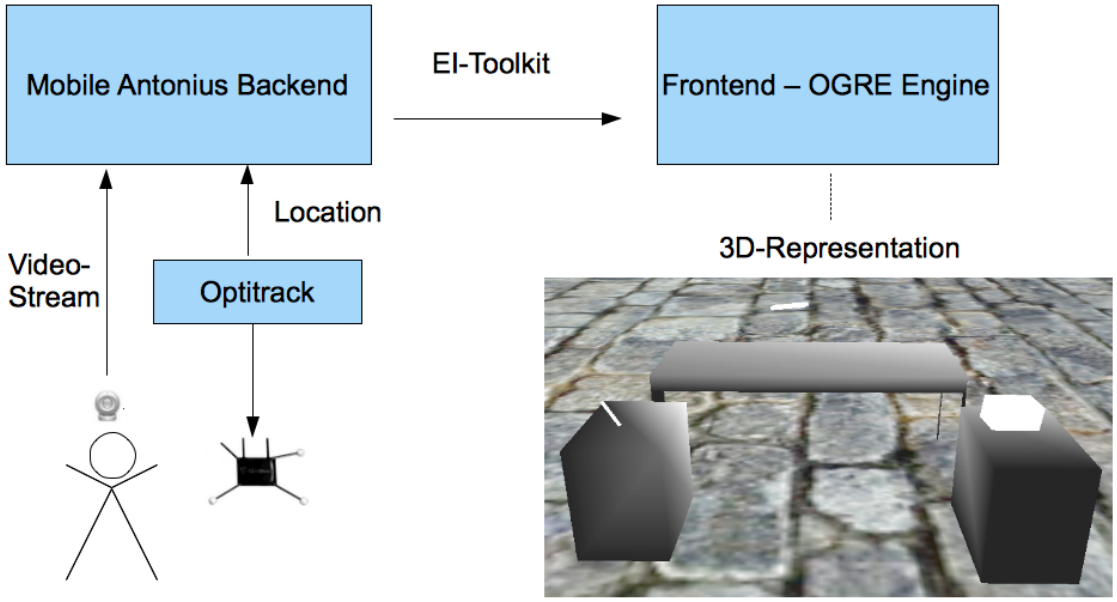


Figure 6.2.: The architecture of the mobile Antonius prototype.

the normal distance from the object to the marker that a user normally has when holding an object.

Due to the two-tiered design of the components, there can be multiple frontends interacting with the system. Each frontend has to run the frontend application, which receives the packages. Because the EIToolkit is broadcasting the packets via UDP, all clients in the same LAN will receive the packets. This can be useful for scenarios, where there have to be multiple screens showing the 3D-representation of the room.

6.3.2. Frameworks and Technology

The mobile Antonius prototype reuses some technologies and frameworks (e.g. BRISK and OpenCV) of the stationary Antonius prototype but also introduces some new technologies and frameworks. Those new technologies and frameworks are being described in this section.

Compared to the previous Antonius prototype, mobile Antonius uses OpenCV 2.4.0., which introduces a few changes in the API concerning the Brutforcematcher. To get it to work, a special 'legacy' header has to be included. In order to use BRISK in debug mode with Visual Studio 2010, one has to build debug libraries of the third-party 'agast' library.

The communication between the frontend and the backend is being done via a framework called EIToolkit [Hol07]. It is a communication framework, which is designed to connect different combinations of software and hardware via language independent network protocols.

A sender gathers the information that is to be shared with another component and builds a string from it. The resulting string is transformed into a network packet. In this case, a sender is used to pack and transfer the location data coming from the OptiTrack system. The EIToolkit is capable of the UDP and TCP transfer protocol. For this project, the UDP protocol was chosen to enable multiple recipients. The EIToolkit also provides a receiver class, which extracts the packet and rebuilds the string that was sent previously. With this toolkit, the location information can be shared with multiple receiving components, which do not need to be connected to the OptiTrack system. They just have to be connected to the same network.

The determination of the location is done using the OptiTrack[1] system and the corresponding software TrackingTools. TrackingTools needs to be calibrated once using an OptiTrack calibration wand, which has to be moved around the testing area to calculate the cameras' positions. Once calibrated, the system can determine the location of a marker inside of the testing area within an accuracy of millimeters. The calibration can be saved within a project file, which has to be loaded when starting TrackingTools. The TrackingTools software can also determine the orientation of the marker. This is used to detect the user's direction of view. Once the marker is tracked by the OptiTrack system, TrackingTools can stream the location data via a multicast protocol. OptiTrack provides an SDK, which can receive those multicast packages and read the location and orientation values. This SDK is called NatNet SDK[2]. Mobile Antonius uses the NatNet SDK (version 2.2) and implements a callback, which is being called every time a multicast packet is received. The system creates a separate thread, which only listens for multicast packets. When a packet is received, the callback writes the location and orientation into a location-handler, which provides the location and orientation to the rest of the program.

To simulate the testing area in a 3D-representation, the OGRE engine[3] (Version Vc10 1.8.1) is being used. OGRE is an open-source 3D graphics engine, which supports OpenGL and DirectX. It comes with an implementation of a sandbox, where the user can move the camera freely through the modeled area. The mobile Antonius frontend uses this sandbox to display a model of the testing area (see Figure 6.3 on the right). The model was created using Google SketchUp 8[4]. With the help of this tool, a 3D model of a table and two cupboards was created. Google SketchUp is also able to measure the model's size in meters, which enabled the creation of an accurate model of the testing area. 3D-modelling programs are working with intermediate file formats, whereas OGRE engine works with its own optimized file format called "mesh". Therefore, a number of transformations have to be done to convert the Google SketchUp file format to a mesh-file. Google SketchUp can export COLLADA (".dae") files. Those COLLADA files have to be imported by the modeling software Blender[5] (Version 2.64a), which can export them in order to create Autodesk 3ds Max files (".3ds"). Then, the Ani2Pov[6]

[1] http://www.naturalpoint.com/optitrack/
[2] http://www.naturalpoint.com/optitrack/products/natnet-sdk/
[3] http://www.ogre3d.org/
[4] http://sketchup.google.com/intl/en/download/
[5] http://www.blender.org/
[6] http://texel3d.free.fr/projets/ani2pov/

Figure 6.3.: Left side: The testing scenario in a lab environment. Right side: The corresponding 3D representation of the scenario. The colored circles show the correlation between a real-world object and its representation within the model.

tool, which can create different 3D model files from 3ds-files, is being used to create an XML representation of mesh files (".mesh.xml"). Last, to convert the mesh.xml file into a mesh file, OGRE's OgreXMLConverter is used. Those mesh files have to be placed into the models folder of the OGRE-SDK. After those steps the OGRE engine is able to display the created models within its sandbox.

The OGRE engine is capable of running user defined code before calculating a frame. The frontend implements a singleton instance of a location manager, which holds the location and orientation of the user for each object which is registered with the system at the time the user sees the object. When a location-changed packet is received, the location manager is updated as well. Before calculating a frame, the system reads the location and orientation information for each object and draws its model on the corresponding place within the sandbox. With this procedure the frontend always displays the most recent location and orientation for each object.

The representation of the location is done using x, y and z coordinates. Each unit in the coordinate system represents one millimeter. The OGRE engine has a different coordinate system. To convert the meter unit representing the real world into the simulated unit within the OGRE engine, the real unit has to be multiplied with 2.5. In order to represent the orientation within the model, OGRE engine uses a technique called quaternion. A quaternion can represent the orientation within a 3-dimensional model using 4 variables called qX, qY, qZ and qW. Those values are calculated by the OptiTrack system, which uses the same representation of the orientation. The system knows the users position and the users orientation at the time the user saw the object. The object is being drawn into the model using the user's orientation and a distance of 40 cm in front and 45 cm below the user's position.

6.4. Evaluation

To evaluate the mobile Antonius prototype, a user study was performed. As a method, the focus group [Gib97] evaluation was chosen because it can be done in a short period of time and it enables the participants to talk about the system without constraints.

6.4.1. Participants and Apparatus

In the focus group study, 5 participants (4 male, 1 female, average age 23.4, SD=1.02) took part. All participants are acquaintances of the author and were recruited randomly at the university. They all study computer science and can be categorized as IT-experts. All participants volunteered to take part in the study. The focus group session took about 60 minutes. First, the facilitator explained, what is being done with the collected data and that an audio file of their responses is recorded during the study. When all participants agreed to that, pictures of the stationary Antonius system were shown to the participants and the functionality and concept of the system was explained. They were told that the stationary system needs many cameras and that it is more efficient and cheaper to have a mobile system. Subsequently, the mobile system was shown and explained. The facilitator put on the system's hat and moved a box from a big table to a small table and showed, how the system displays the location of the object live in the 3D model. Then, the hat was given to a participant to try the system himself. After that, the pen object was introduced by the facilitator to show that the system is capable of multiple objects. When the group was familiar with the system, an open interview with key-questions was conducted. The key-questions can be found in Appendix A.2. After the interview, the recording was transcribed and the generated ideas were summarized.

6.4.2. Results

The participants would use a system like mobile Antonius in every day situations to track rather small objects because those are the ones which they have experienced difficulties in finding. As a use case, the participants thought of a shopping scenario, where the user is wearing the system while being at a supermarket and receiving real-time information about products the user is currently looking at. This information could be price, ratings, a description and offers based on the current product and could be displayed on a smartphone or in a head-mounted display. Another use case could be just locating objects in a confusing situation, like in a supermarket. The user is not familiar with the arrangement of the objects within a shelf and it may take the user a while to spot a sought product. Mobile Antonius could spot a sought product faster than the user and could provide help, where to find the product. The system could also be combined with a shopping list in order to match objects that are already in the shopping-basket with the objects that are on the shopping list.

When being asked about, if they would be able to find a sought object with the help of the 3D representation, all participants agreed. The participants suggested to highlight sought objects in the 3D representation in order to distinguish them from other objects. This could be for

example a red circle around the object or an additional light pointing at the object. In a home scenario, the participants could imagine having a terminal, which runs a 3D model. To make the system more mobile, a head-mounted display or a smartphone app running the model and additionally pointing out directions towards the object would be possible as well. Compared to the 2D map, which was part of the stationary Antonius system, all participants preferred the 3D representation of the testing area used in the mobile Antonius prototype. They claimed, it is easier to map a 3D representation into the real world than a 2D map. Furthermore, a 3D representation is better to determine the height of an object within a room. A 2D map only provides x and y coordinates and is not giving height information. Therefore, it is more difficult for a user to discover, whether an object is at a higher position, for example on a shelf, or it is lying on the floor. Compared to the textual representation, all participants would also prefer the 3D representation. The participants said that the textual representation is only giving a rough idea of where the object is at and it is more difficult to understand, what is meant with the textual description of areas. They also mentioned that an audio output could be built to support blind people. Mobile Antonius could help a blind person to find objects and provide directions towards the sought object, and therefore increase his or her quality of life.

The participants were also asked to compare the mobile Antonius system to the idea of the stationary Antonius system. Some of the participants told, it would feel strange to have cameras all over the apartment. Another participant could imagine a hybrid approach, where important areas, like an office, could be monitored constantly using stationary cameras. Then, the user wouldn't have to wear the hat all the time while working. In a bigger area, for example within an apartment, a mobile system could be used. The stationary cameras could work together with the mobile camera in order to extend the stationary application site. Some participants prefer the stationary case, because wearing a hat all the time would be inconvenient. Others prefer the mobile case, because they can take off the hat when entering more private areas like the bathroom. When the private time is over, the user can enter the bathroom wearing the camera and keep track of bathroom related objects. With a stationary system, the user would not be able to track objects in the bathroom, unless he or she gives up his or her privacy.

Because the hat can be worn differently each time, a hat may not be the best choice to wear a system like mobile Antonius. The position of the marker is different with each user because the shapes of the users' heads differ. Equipping a more fixed construction, like a helmet or glasses, with the system would be more robust. Each user is wearing glasses the same way. Therefore, the orientation of the marker and the position of the camera would not change for a different user. Female users, who might not want to wear a helmet or glasses, could wear an earring equipped with a camera. Another suggestion was to equip the hat with more than one camera in order to being able to have a 360° view around the user. Adding this, the system could even see behind the users back and tell him, when he is looking for something in the opposite direction. Currently, the prototype is connected to the computer with a USB cable. A further improvement could be to equip the wearable system with a Wi-Fi connection in order to transfer the video to the computer without a cable. To register objects with the system, the object's UPC (Universal Product Code) barcodes could be used. Information about the object could be gathered from the Internet. Maybe even a reference picture could be fetched

online. To improve the object detection, the system could use color histograms in order to previously filter not matching colors and save computation time. This step has to be done prior to the matching because both BRISK and SURF work on gray-scale images.

The participants also discussed about the technology changing the users' lives. When a system like this could find all real-world objects, the user wouldn't have to organize his or her real-world objects anymore and just keep all objects on piles. The system would help a user finding an object again. This would be bad for a person, who does not want to use the system but has to live together with a person using the system.

6.5. Discussion

The study showed that both the mobile system and the stationary system have their advantages and disadvantages. But the mobile system would be more accepted in a user's daily life because having the ability to not wearing the system in private situations, the user feels to be more in control. Also, a mobile system is cheaper and more scalable than a stationary one.

The results also reveal that a 3D representation is easier to understand for a user than a 2D or a textual representation. That is, because the user is familiar with a 3D representation of a room. When reading a 2D or a textual representation, the user has to put cognitive effort into mapping the representation into a (3D) real world.

One can argue that feature-based optical tracking systems and real-world search engines existed before and the combination of both was also introduced before. But implementing a combination of both within a mobile, wearable and working prototype is a new idea. Also, with the use of the OptiTrack system, mobile Antonius is more accurate than previous systems. Because the accuracy is within millimeters, it could now be possible for a robot to fetch a sought object automatically and a human being would not be needed anymore. This could make fully automated and marker-less warehouse scenarios possible.

But there are also problems with the system. Using the OptiTrack system, the cameras need a clear sight to track the marker. In a normal scenario, this is not a problem because the user is wearing the marker on his head. But if he is crawling under a table, the system is not able to track the marker anymore. An integration of the camera and the marker into other wearable items might be cumbersome because the user could stand in the visual line between the marker and the OptiTrack cameras. Therefore, applying the marker onto the users head is the best way. Also, mobile Antonius is not scalable because it uses the OptiTrack system for determining the indoor location. There have to be OptiTrack cameras everywhere, where the system should be used.

A focus group study was conducted to evaluate the mobile system. This approach brings advantages and disadvantages [Gib97]. The main advantage is that a focus group study can be conducted in little time because all participants take part in the study at the same time. Also, a participant's suggestions and opinions have to be explained more precisely, because other participants can question them. A focus group-based evaluation is a qualitative approach, which cannot be controlled. In an ideal case, participants also discuss with each other about

the topic. This is why the facilitator cannot influence the participants this much with posing questions, like he or she could in one-on-one interviews. Also, a participant's opinion can inspire another participant, who can improve or extend an idea. On the other hand, a participant's opinion can also influence another participant. For example, if 4 out of 5 participants agree on something, the remaining participant might just agree to the others instead of telling his or her own opinion and then having to defend it in front of the others. This is why quantitative results from focus group studies could be biased because opinions might be influenced by other participants. Compared to a one-on-one interview, a shy participant can hide among the other participants. Those participants have to be asked directly for their opinion in order to integrate them into the discussion. Also, when a participant does not feel confident about the discussed topic, he or she might not join the discussion and rather act passively. This is the same with personal or sensitive information. Participants will reveal sensitive or personal information more likely in a setting, where there is just the facilitator and the participant. Other participants might not keep the information as confidential as the facilitator. Another problem is that there could be personal differences between participants, of which the facilitator does not know. This could also influence a participant's behavior during a focus group study.

6.5.1. Limitations

With a system like mobile Antonius, the author believes, a step towards a smart home was performed and the gap between the physical and the digital world was shortened. However, there are a few limitations, which are described in this section (some of them are inspired by [Ped01]).

- Mobile Antonius has a problem, when the user is moving several objects at once because the current algorithm is only capable of tracking one of them. In an everyday office situation, this would restrain a user's productivity. If a user moves multiple objects at once, the system might not recognize that the user moves other objects as well and thereby, gets into an inaccurate state. The wrong location of additional moved objects can only be corrected when the user looks at the object coincidentally.
- The system only can survey a limited area because it needs OptiTrack cameras to determine the indoor location. When a user leaves the application site, the system is not able to tell an object's position anymore. Also, the camera is connected with the computer through a 5 meter USB extension cable. Therefore, the area cannot be wider than the length of the cable.
- Only users, who are wearing the system, are allowed to move objects. Otherwise, the changes in location are not being updated. For example, in an office environment, the cleaner would have to wear the system, too. In a daily life situation within an apartment, this restriction would also prevent visitors from moving any object.
- A user has to look at an object, when changing its location. For example, when a user is carrying an object behind his back, the system will not be capable of tracking it. Furthermore, if a user is dropping an object and is not looking at it anymore, the final

location of the object will not be tracked by the system. It will assume that the object is still in the air, where the user held the object last.

- The distance which is shown in the model between the user and the object is currently a fixed distance away from the marker. The system does not distinguish between an object that is far away from the camera and an object that is very close to the camera. Also, the length of the arm changes for each user. Both problems could be solved using, for example, infrared distance measuring tools.
- The detection of an object is always dependent on the lightning conditions within the room. If a reference picture was taken under a very bright light, the system will have difficulties recognizing the object inside a dim room and vice versa.
- A reference picture only depicts one side of an object. The system can only recognize an object, which is facing the camera in the same way the object in the reference pictures does. To recognize an object from multiple angles, multiple reference pictures have to be supplied.
- The position of an object within the model is represented with a coordinate. For smaller objects, like a pen, this is not a problem. But for larger objects, like a huge box, this can cause problems with accurately displaying the object within the 3D representation.

7. Conclusion

This thesis showed that an object detection-based search engine for the real world is feasible, even without the use of marker technology or intrusive tags. Although, visual object detection engines already existed, this thesis successfully introduced a new way of constantly and implicitly interacting with a system by building a search engine which not only can detect objects, but also can remember last seen locations to facilitate finding a real-world object. The main goals of this research were, to find out whether users will accept an integration of an object detection-based real-world search engine into their daily lives or not, what benefits the user is gaining from using a system like this, which representation of an object's location is most understandable and most suitable for everyday situations, and whether it is better in terms of both acceptance and efficiency to instrument the application site or to instrument the user.

As a proof of concept a stationary prototype called Antonius was implemented using a scalable architecture. The user study showed that people are fascinated by the technology and the idea of a real-world search engine, and that they would use such a system in their daily lives. However, the constant surveillance of an area inside a user's home with web cameras also brings up privacy issues. According to the user study, some users do not want images of their home being sent to a server, while others would give up parts of their privacy for the benefit of never losing an object again. Also, participants stated that the idea of having each room of a user's home equipped with web cameras would feel weird for them. Furthermore, needing this many cameras to cover a whole room, like the Antonius prototype, is not very scalable. Thus, the idea of a mobile prototype came up. Another result of the study was that each participant found the 2D graphical representation of the location easier to understand and more convenient to use than the textual representation. Hence, the author decided to try out an even more graphical approach for representing the location, which is representing it within a 3D model.

To prove the mobile concept, another prototype called mobile Antonius was implemented, which combined the idea of a wearable mobile camera and a 3D representation of an object's location inside a mobile object detection-based real-world search engine. A focus group user study showed, that the 3D representation of an object's location is even easier to understand than a textual or a 2D map representation. Furthermore, the idea of a mobile prototype was more appealing to the participants because they feel in control of a wearable system. If they do not want to use it, for example in the bathroom, they can just take it off.

This research shows that users are likely to accept an object detection-based search engine for the real world when it is clearly stated, what is being done with their data and when they feel in control of the system. The representation of an object's location is most understandable,

when being displayed in a 3D model because the user does not have to map the textual representation onto the real world anymore. In stressful situations, an audio representation of the location is suitable as well because the user may not have the time to operate the system. Instrumenting the user with a mobile system is better in both acceptance and scalability. The focus group study showed that users feel weird when constantly having stationary cameras around them. In contrast, a wearable and mobile system can be taken off whenever the user wants some privacy. In terms of scalability, both stationary and mobile systems are dependent on stationary hardware, but the mobile approach has more opportunities in getting rid of OptiTrack by using alternative indoor location technologies like WiFiSLAM. The main benefit a user gets from using a real-world search engine is, being aware of each object's location independent from where the user currently is. Maybe a system like mobile Antonius will have an impact on how people live their lives, like the mobile phone did a few years ago. Perhaps people will not need to organize their living area anymore because finding an object will be just one search-query away.

As a result of this research, a lot of great ideas for using an object detection-based real-world search engine were generated. From using the system while being at a supermarket to check the fridge at home in order to see what is inside, to getting directions towards a shelf, where a sought product is at, many areas of applications were covered. The most promising idea is to use the system in order to help blind people. Mobile Antonius could identify objects in their daily lives and use an audio output to tell them which objects are within their reach. Furthermore, it could provide directions towards a sought object in order to facilitate finding it. It could be used as a tool, which is seeing for them and is additionally capable of remembering everything it ever saw.

As systems like Antonius become more pervasive, it will become possible to locate and track virtually every object in the user's environment. In addition to providing real-world search, this will also facilitate completely new forms of personalization and targeting of content, as well as allowing for more relevant advertisement. As we move beyond the known online Internet into the new Internet of Things, the opportunities may be just as revolutionary as when the first search engines entered the market some decades ago.

8. Future Work

In this chapter, ideas for future improvements of the system are introduced. All introduced ideas came up during the work on this project and should provide inspiration for future attempts on a visual real-world search engine. The introduced ideas can also be used to extend both the Antonius and the mobile Antonius project, and are therefore categorized into a stationary and a mobile approach.

8.1. Stationary Case

The data structure used within Antonius could be extended towards accepting more general search terms. The type attribute (compare with Section 4.2) is a precursor to this attempt, which enables the user to search for the type of the object in case he or she can't remember what the object was called exactly. This approach can be extended by adding a tree of synonyms and hyponyms to the database structure. Providing this information would be too much effort for a single user to spend each time an object is registered with the system. The system could provide synonyms and hyponyms automatically using Wiktionary [ZMG08]. This information can be pulled from the Wiktionary API by providing just one keyword. The resulting synonyms and hyponyms could be stored in the database and thereby, improve the search towards being more user-friendly.

In the current stationary Antonius system, the location and orientation of the cameras have to be given to the system when setting it up. This means, the person who is setting up the system has to measure the position of the camera to figure out its position within the coordinate system and has to measure its orientation towards a defined point in degrees. To facilitate this process, the cameras could be used to recognize their position on their own using visual scene recognition. This could work the same way as the camera detects objects. The live video is compared to reference images and when some unique interest points were found, the system can estimate the cameras position. Of course, when first installing the system, the user then would have to provide reference pictures of the area. But with this approach, the system could handle spontaneous changes in the cameras location without having the user to provide information. Another approach for facilitating the calibration of the cameras could be to use an indoor location system. This would make the system more flexible and robust to changes of the position of the cameras. But on the other hand, the system would become dependent of the used indoor location system, for example OptiTrack.

In the user study, some participants claimed that it is too much effort to register every single object that should be trackable. An approach to solve this problem could be creating a huge

crowd-sourced database which is accessible via the Internet. In an ideal case, the system would know every ordinary object and the user just has to register unique personal items. A system like this could learn from the crowd. When a user is registering a bar of chocolate, he could be asked, if this is a private item or not. If not, the inserted data and the reference picture could be uploaded to a global database. Based on this data, every other Antonius system, which is connected to the Internet, could detect a bar of chocolate that looks the same than the one on the reference picture. Previously introduced systems like oMoby, Google Goggles or the Amazon App show that this is feasible. Another way to get reference data could be crawling the web for reference pictures and information. For example, items sold on eBay have keywords in the title, a description, and a reference picture, too. This data could be used to learn additional objects automatically.

A problem of the Antonius system is that it is estimating the distance from the camera to the object based on the proportion between the reference picture and the bounding box, which is drawn around the object in the live image. To measure this distance, an infrared-based distance measuring system like the Microsoft Kinect[1] could be used. Because the Kinect uses infrared, multiple Kinects may interfere with each other. If a system using multiple Kinects should be built, a consecutive approach will be necessary.

To facilitate finding objects in the real world, visual pointers could be used to light up a sought object or area, like it was done in the SearchLight project [BSS04]. This would solve the problem of users who do not understand which camera is meant by a given textual description. To achieve this, the camera has to be combined with a pointing device, for example a laser pointer. When a user is sending a query for a sought object, the camera that most recently saw the object could activate its laser pointer and point onto the sought object. The user would just have to follow the ray to find the object. Another more general approach would be to equip the cameras with a red light. When receiving a query, the camera which most recently saw the object could flash its light to give the user a rough idea where to search for the object.

The most important suggestion for extending the system is to build a smartphone app. An app could combine the process of taking a picture, transferring it to the computer, uploading it to the system and adding a description. This would facilitate the registration process a lot because a user does not need to connect his or her camera or phone to the computer each time he or she wants to add a picture. An app also could provide help in pre-analyzing the picture and tell the user, if the recorded picture is not sharp or bright enough. The BRISK and SURF algorithm also runs within an app. Therefore, the app could also provide information about how many feature points can be found in an image and whether the photographed object is unique enough. The 2D map of the testing area, which appealed to the participants, is also much effort to initially set up. The system could be extended in order to help the users in creating a 2D map of the application site. There exist Augmented Reality tools to draw a floor plan with small effort, for example MagicPlan[2]. The user has to walk around with a smartphone and visually mark corners of an apartment. A floor plan of the apartment is being

[1] http://www.microsoft.com/en-us/kinectforwindows/
[2] http://www.sensopia.com/english/index.html

drawn, based on the defined corners. A tool like this could also be integrated into an Antonius smartphone app to make providing a map of the application site more convenient.

8.2. Mobile Case

In order to reduce the amount of cameras to a minimum, the system has to get rid of the OptiTrack system. A promising way of achieving that is to use an indoor location system like WiFiSLAM[3]. It estimates the position of a user's smartphone by looking at the relative strength of Wi-Fi signals. The accuracy of a Wi-Fi-based system is within a few meters [Far12]. This is a deterioration compared to the OptiTrack system because OptiTrack can measure the location within millimeters of accuracy. But the ease of use and the mobility the system gets from using a system like WiFiSLAM is promising. A user would not have to wear a marker anymore and the system could be used in rooms, which are different from a predefined testing area equipped with cameras. WiFiSLAM's demo video[4] shows that calibration of new areas can be done within a minute. An existing floor plan of the area can be imported via a camera. WiFiSLAM's app supports printed floor plans like an emergency exit plan or even hand drawn sketches of a floor plan.

Another promising improvement for an object detection-based real-world search engine like mobile Antonius, could be integrating it into a wearable system like Google's project glass[5]. Project glass is a personal computer integrated into glasses, which provide an Augmented Reality experience for the user in his or her daily life. The idea behind project glass is that the user is constantly wearing the glasses throughout the whole day and the system is supporting the user with features, like taking a picture of what the user is currently looking at, or getting directions while being on the move. Project glass could run an application similar to mobile Antonius and track the location of objects the user is interacting with. Integrating the idea behind mobile Antonius into Google's project glass would solve the problem of users feeling weird when having a web camera on their hat because it is integrated into their glasses. The problem of determining the indoor location of the user is not solved by using project glass. But a combination of project glass and WiFiSLAM could be done as well.

The mobile system could also be extended by a distance-measuring tool, for example a Microsoft Kinect. Currently, the system uses a fixed distance from the marker's position on the user's hat to draw the object inside the 3D representation. With determining the distance, the system becomes more accurate. But a distance-measuring sensor could also be used to measure the distance towards the walls of the room and create an indoor map of the area - just by wearing the system. Imagine, all previously mentioned suggestions are being combined to create a wireless, Wi-Fi-signal-based and wearable device, which can measure distance and create indoor maps. The data collected by many users' devices could be combined to create a

[3] http://www.wifislam.com/
[4] http://www.youtube.com/watch?v=B_GdXp_Swjs
[5] http://g.co/projectglass

crowd-sourced approach for an automated indoor cartography system, like OpenStreetMap[6] does outdoors with submitted GPS data.

In a future project, multiple mobile Antonius devices could be interconnected. This would result in having a more up-to-date location, because more users are recognizing more objects. Also, this would enable a user to get directions towards an object, which the user did not previously see, but was detected by another system. Of course, a combination of both stationary and mobile Antonius systems could be promising as well. A stationary system could cover an area, where there are many objects and where privacy issues are negligible, for example an office area. A mobile system could cover a more private area, where the user does not want stationary cameras around, for example inside the user's apartment. When those systems are connected to a server, a user could search for an object's location within all areas he or she usually resides in and even would be able to see a live image of objects that are located within the stationary environment.

[6] http://www.openstreetmap.org/

A. Appendix

A.1. User Study Procedure and Questions

The following is the procedure of the Antonius user study, conducted in August 2012.

1. Tell the participant about what the user study will be like (audio recording / use results for paper/evaluation/diploma thesis). Let participant agree to those terms.
2. Tell what Antonius does. (Real-world search engine/ finding lost objects/ visual object detection / OCR / map and UI/ explain 2 use cases: pen under a stack of papers & document)
3. Ask for brought user object
4. Take picture of the object
5. Ask user to register the object with the system (Provide help! / Explain what's going on and why we do this)
6. Place object somewhere in the test-area in an angle that the system can recognize. (After that: give the system a little time to recognize/meanwhile do first interview)
7. First interview
   - Fill general questionnaire (demographics)
   - Start recording (QuickTime Audio recording)
   - How often do you use a search engine? (Like Google or Bing)
   - Are you often losing objects or experiencing difficulties in finding objects? If yes, what objects are those?
   - What kind of strategies do you currently use to find lost objects? (Ask friends/re-visit places? What else?)
   - Which objects do you take along with you every day?
   - Do you always store those objects at the same places, when you are at home? Where do you usually store them?
   - Stop recording
8. Let user search for his object using Antonius (provide help and explain!)

9. Play around a little bit - move object to another location – let user search again – show both map and search-results. Goal: user should get a feeling for the system and understand what the system does.
10. Exit interview
    - Start recording: Say something like: This is a totally exploratory session, and the idea is to have fun and imagine together. We encourage you to be as honest as possible in your responses and not be afraid of saying things you don't like or don't understand, or talking about your dreams or ideal situations. There are no constraints here!
    - What did you expect when hearing about a real-world search engine – where your expectations met?
    - Would you use a real-world search engine in your daily life? Why?/ Why not?/ Could you imagine use cases different from the given office example?
    - Where would you set up this system? (In which rooms in your home? / other places?)
    - Was the location of the found objects accurate enough for you to find them?
    - Is the UI is intuitive and easy to use? Why? Why not? What would you expect? How would you like to do the search? How and in what way would you like to have the results presented? (Maybe voice... ?) How would you access the system? (Smartphone / voice etc.?)
    - What do you think about the interaction? (General principle of progressive implicit interaction and explicit search – maybe explain something about other interaction types)
    - Do you see problems with this form of interaction?
    - Would you feel comfortable using this system in your home? / In which rooms would you set it up?
    - We know that a system like this is a privacy issue. You would need to install web cameras in your home/office in order to use the system. Would you still use a system like this for the benefit of never losing objects again? Would it make a difference to you if the detection is done on a server or a local machine? What about anonymization techniques? Would you use it if the images couldn't be tracked back to you?
    - Any suggestions/ Comments/ Thoughts / Ideas?
    - Stop recording
11. Save all generated data for analyzing and transcribing

A.2. Focus Group Key Questions

Those are the key questions, which were used in the focus group session conducted November 21th, 2012.

1. Tell participants about the use of the data and that they are being recorded. Let them agree to that.
2. Thoughts about the system
   - Where would you use the system
   - Additional use cases/scenarios?
   - Would you find an object based on the provided information?
3. Comparing it to the stationary case
   - Which one feels better?
   - Which one would you use more likely?
4. Comparing the 3D representation to the 2D map
   - Which one is better? Why?
   - Which one would you use more likely?
   - Which one is easier to understand?
5. Comparing the 3D representation to the textual representation of location
   - Better/ Worse?
   - What about spoken textual representation (voice representation)?
6. Suggestions (How can I improve the system?)
7. Problems (Do you see any problems?)
8. Privacy issues (Would you use a system like this in your daily life?)
9. Integrating a project like this into Google glasses for the daily use?
   - Would this be ok?
   - Would you use a system like this?

Bibliography

[AHS07] K. Aberer, M. Hauswirth, A. Salehi. Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. In *Proceedings of the 2007 International Conference on Mobile Data Management*. 2007.

[BSS04] A. Butz, M. Schneider, M. Spassova. SearchLight - A Lightweight Search Function for Pervasive Environments. In *Second International Conference on Pervasive Computing*. 2004.

[BTVG06] H. Bay, T. Tuytelaars, L. Van Gool. SURF: Speeded Up Robust Features. In *Computer Vision ,ECCV 2006*. Springer Berlin / Heidelberg, 2006.

[CGKM07] R. O. Castle, D. J. Gawley, G. Klein, D. W. Murray. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proc. International Conference on Robotics and Automation, Rome, Italy, April 10-14, 2007*. 2007.

[CLSF10] M. Calonder, V. Lepetit, C. Strecha, P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Computer Vision – ECCV 2010*, volume 6314, pp. 778–792. 2010.

[CYH+03] D. J. Cook, M. Youngblood, E. O. Heierman, III, K. Gopalratnam, S. Rao, A. Litvin, F. Khawaja. MavHome: An Agent-Based Smart Home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*. 2003.

[Dav03] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. 2003.

[Far12] C. Farr. Indoor location is ready for its second act. `venturebeat.com/2012/08/30/indoor-location-is-ready-for-its-second-act-exclusive/`, 2012.

[FB81] M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, pp. 381–395, 1981.

[FBMK08] C. Frank, P. Bolliger, F. Mattern, W. Kellerer. The sensor internet at work: Locating everyday items using mobile phones. *Pervasive Mob. Comput.*, 2008.

[FBRK07] C. Frank, P. Bolliger, C. Roduner, W. Kellerer. Objects calling home: locating objects using mobile phones. In *Proceedings of the 5th international conference on Pervasive computing*. 2007.

[FH03] A. Field, G. J. Hole. *How to Design and Report Experiments*. Sage Publications Ltd, 2003.

[FPR05] K. P. Fishkin, M. Philipose, A. Rea. Hands-on RFID: Wireless wearables for detecting use of objects. In *In ISWC 2005*. 2005.

[Gib97] A. Gibbs. Focus Groups. *Social Research Update*, 1997.

[HHS+02] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster. The anatomy of a context-aware application. *Wirel. Netw.*, 2002.

[Hol07] P. Holleis. Programming Interactive Physical Prototypes. In *Workshop on Design and Integration Principles for Smart Objects (DIPSO2007)*. 2007.

[Hol12] L. E. Holmquist. *Grounded Innovation: Strategies for Creating Digital Products*. Morgan Kaufman Publ Inc, 2012.

[KBM+02] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, M. Spasojevic. People, places, things: web presence for the real world. *Mob. Netw. Appl.*, October 2002.

[Kir95] D. Kirsh. The intelligent use of space. *Artif. Intell.*, 1995.

[KNLZ07] A. Kansal, S. Nath, J. Liu, F. Zhao. SenseWeb: An Infrastructure for Shared Sensing. *IEEE MultiMedia*, 2007.

[Lag11] R. Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook*. PACKT Publishing, 2011.

[LCS11] S. Leutenegger, M. Chli, R. Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV), 2011*, pp. 2548 – 2555. 2011.

[Low04] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 2004.

[LRH00] P. Ljungstrand, J. Redström, L. E. Holmquist. WebStickers: using physical tokens to access, manage and share bookmarks to the Web. In *Proceedings of DARE 2000 on Designing augmented reality environments*. 2000.

[LWJH06] S. Ljungblad, K. Walter, M. Jacobsson, L. Holmquist. Designing Personal Embodied Agents with Personas. In *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication*. ROMAN, 2006.

[MF10] F. Mattern, C. Floerkemeier. From the Internet of Computers to the Internet of Things. In *From Active Data Management to Event-Based Systems and More*, pp. 242–259. Springer Berlin / Heidelberg, 2010.

[ORM+10] B. Ostermaier, K. Römer, F. Mattern, M. Fahrmair, W. Kellerer. A Real-Time Search Engine for the Web of Things. In *Proceedings of Internet of Things 2010 International Conference (IoT 2010)*. 2010.

[PCB00] N. B. Priyantha, A. Chakraborty, H. Balakrishnan. The Cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*. 2000.

[Ped01] T. Pederson. Magic Touch: A Simple Object Location Tracking System Enabling the Development of Physical-Virtual Artefacts in Office Environments. *Personal Ubiquitous Comput.*, 2001.

[PMB11] S. Parlak, I. Marsic, R. S. Burd. Activity recognition for emergency care using RFID. In *Proceedings of the 6th International Conference on Body Area Networks*. 2011.

[RD06] E. Rosten, T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2006.

[ROM+10] K. Römer, B. Ostermaier, F. Mattern, M. Fahrmair, W. Kellerer. Real-Time Search for Real-World Entities: A Survey. *Proceedings of the IEEE*, 2010.

[RS03] G. Reitmayr, D. Schmalstieg. Location based applications for mobile augmented reality. In *Proceedings of the Fourth Australasian user interface conference on User interfaces 2003 - Volume 18*. 2003.

[SAH08] C. Silpa-Anan, R. Hartley. Optimised KD-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008.

[SAW94] B. Schilit, N. Adams, R. Want. Context-Aware Computing Applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. 1994.

[Sch00] A. Schmidt. Implicit Human Computer Interaction Through Context. *Personal Technologies*, 2000.

[SGM00] A. Schmidt, H.-W. Gellersen, C. Merz. Enabling Implicit Human Computer Interaction: A Wearable RFID-Tag Reader. In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*. 2000.

[Smi07] R. Smith. An Overview of the Tesseract OCR Engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*. 2007.

[TCC+10] S. S. Tsai, D. Chen, V. Chandrasekhar, G. Takacs, N.-M. Cheung, R. Vedantham, R. Grzeszczuk, B. Girod. Mobile product recognition. In *Proceedings of the international conference on Multimedia*. 2010.

[TMFR03] A. Torralba, K. P. Murphy, W. T. Freeman, M. A. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. 2003.

[Wei91] M. Weiser. The Computer for the 21st Century. *Scientific American*, 1991.

[WHFaG92] R. Want, A. Hopper, V. Falcão, J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 1992.

[WTL10] H. Wang, C. C. Tan, Q. Li. Snoogle: A Search Engine for Pervasive Environments. *IEEE Trans. Parallel Distrib. Syst.*, 2010.

[YGM08] T. Yan, D. Ganesan, R. Manmatha. Distributed image search in camera sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*. 2008.

[YGTD05] T. Yeh, K. Grauman, K. Tollmar, T. Darrell. A picture is worth a thousand keywords: image-based object search on a mobile platform. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*. 2005.

[YSM05] K.-K. Yap, V. Srinivasan, M. Motani. MAX: human-centric search of the physical world. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*. 2005.

[ZMG08] T. Zesch, C. Müller, I. Gurevych. Using wiktionary for computing semantic relatedness. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*. 2008.

[ZSA+10] R. van Zwol, B. Sigurbjornsson, R. Adapala, L. Garcia Pueyo, A. Katiyar, K. Kurapati, M. Muralidharan, S. Muthu, V. Murdock, P. Ng, A. Ramani, A. Sahai, S. T. Sathish, H. Vasudev, U. Vuyyuru. Faceted exploration of image search results. In *Proceedings of the 19th international conference on World wide web*. 2010.

All links were last followed on December 19, 2012.

Declaration

I declare herewith that I am the author of this diploma thesis. I did not use any other sources than those named and all those passages quoted from other works either literally or in the general sense are marked as such.
Neither the complete thesis nor essential parts of it have been used up to now in any other examination procedure.
I have not published this thesis up to now either in part or as a whole.
The electronically submitted version is identical to all the other versions which have been submitted.

(Markus Funk)