LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Florian Alt

Master Thesis

An Approach for Spontaneous Pursuits Calibration in the Real World

Lukas Mecke
mecke@cip.ifi.lmu.de

Bearbeitungszeitraum:	1. 5. 2017 bis 31. 10. 2017
Betreuer:	Mohamed Khamis
Externer Betreuer:	Dr. Andreas Bulling
Verantw. Hochschullehrer:	Prof. Dr. Florian Alt

Zusammenfassung

Blickbasierte Interaktion ist eine vielversprechende Modalität für Anwendungen im öffentlichen sowie privaten Umfeld. Allerdings ist es aktuell notwendig einen Eye-Tracker zuvor zu kalibrieren um akkurate Schätzungen der Blickrichtung zu erhalten. Eine Möglichkeit diese Limitation zu umgehen ist ein Ansatz namens Pursuits. Dabei werden Bewegungen des Auges mit den Bewegungen eines auf einem Bildschirm dargestellten Objekts korreliert um Selektionen vorzunehmen oder einen Tracker zu kalibrieren.

Im Rahmen dieser Arbeit wird diese Idee weiterentwickelt. Aktuell ist es notwendig Objekte auf einem Bildschirm händisch zu erzeugen wohingegen mit unserem Ansatz beliebige Objekte aus der realen Welt als mögliche Interaktionsziele verwendet werden sollen. Diese werden aus den Videodaten der Welt-Kamera eines Eye-Trackers extrahiert. Diese Arbeit enthält einen neuartigen Ansatz, den wir dense Pursuits nennen. Dabei werden durch die Anwendung umgekehrten optischen Flusses Bewegungspfade für jeden Pixel eines Bildes erzeugt. Diese werden alsdann mit Blickpunkten korreliert, sodass Objekte denen ein Benutzer mit den Augen folgt daraufhin einer erzeugten Wahrscheinlichkeitsverteilung entnommen werden können.

Um diesen Ansatz echtzeitfähig zu machen wird ein alternativer Ansatz namens sparse Pursuits vorgeschlagen. Dieser verwendet anstatt eines dichten einen dünnbesetzten Algorithmus für optischen Fluss auf einer heuristischschen Auswahl an Bildpunkten. Die daraus entstehenden Bewegungspfade können dann wieder mit Blickpunkten korreliert werden.

Zum Abschluss wird eine Evaluation der Genauigkeit der erzeugten Blickabschätzung durchgeführt. Zudem werden mögliche Anknüpfungspunkte für zukünftige Arbeiten vorgestellt.

Abstract

The use of eye gaze holds a lot of promises for interactive applications in private but also public settings. Commercial applications are, as of now, limited due to the necessity of calibration in order to achieve accurate results. One approach to overcome this limitation is Pursuits. It is based on a correlation of eye movement and moving objects on a screen to detect selections or calibrate an eye tracker in the background.

The goal of this thesis is to extend the idea of using Pursuits for calibration. The current approach requires predefined on-screen objects that a user has to follow with her eyes whereas the aim of this thesis is to perform the calibration with arbitrary objects extracted from a video stream provided by the scene camera of an eye tracker. For this we introduce a novel approach that we call dense Pursuits. It is based on generating trajectories for each pixel in an image by applying reverse dense optical flow and correlating them to the current gaze position. Objects that a user follows with her eyes can then be extracted from the created probability map.

Additionally we propose an altered version of this approach that we call sparse Pursuits. This is dedicated to making dense Pursuits real-time viable. To achieve this we use a sparse optical flow algorithm on a set of heuristically selected feature points. So generated trajectories are then correlated with gaze.

We finally evaluate the proposed approach with respect to calibration accuracy and propose possible future work to further investigate the potential of our method.

Task description

The use of smooth pursuit for gaze-based interaction (Pursuits) was shown to be natural and promising for human-computer interaction. Since its introduction in 2013, Pursuits has been employed in different contexts, ranging from interaction with smartwatches to interaction in smarthomes and with public displays. It has been used for both: interaction and eye tracker calibration.

In order to interact via Pursuits, a stimulus must be moving in the user's view. To date, existing systems have artificially imposed a moving target to stimulate smooth pursuit eye movements. However, objects move around us all the time in our daily lives, and the detection of smooth pursuit eye movements for interaction or calibration using random in-the-wild targets was never investigated before.

In this thesis, the aim is to investigate how moving targets in the real world can be detected and compared to eye movements of the user, in order to detect if the user is following one of them. After a smooth pursuit is detected, the collected data can allow either for interaction, or for calibration.

Tasks:

- Review of related work in Pursuits
- Experimenting with different methods and implementing the detection of moving objects
- Evaluating the final method and comparing it to previous work

Contents

1 Introduction

The given master thesis aims to improve the applicability of gaze interaction in public settings. In the following chapter we will cover the reason why it is necessary to improve on this in the first place and give a short overview over this thesis afterwards.

1.1 Motivation

Eye-tracking has come a long way and was reported as a means to analyse reading behaviour [29] as early as 1908. Over the years it has developed into an important input modality and observational tool widely used in psychology but also other fields.

In the context of Human-Computer-Interaction (HCI) eye tracking is mostly of interest as an input modality for novel interaction techniques. Till now this requires a technique called calibration (further elaborated in *chapter* 2) which is the procedure of mapping eye movements to a gaze prediction and has to be performed before being able to interact. This leads to more accurate results but it can also be a tedious and tiresome process. Although it might only take a few seconds it greatly hinders the potential for eye-tracking in any non static environment, such like public space. Also spontaneous interaction in a private space is limited to users known to the system. Especially for scenarios like interaction with public information displays a previous calibration is not feasible as passers-by are often in a hurry and not willing to invest additional time.

To overcome this limitation there are several approaches that don't rely on calibration (e.g. [54, 76]). For the course of this work we will focus on a technique by Vidal et al. [66] called Pursuits which is based on smooth pursuit eye movements [48]. The basic principle is to correlate eye movements with the trajectories of a set of scripted moving targets and - given a high correlation - use this for selection. Due to the nature of the correlation used (Pearson correlation), no absolute coordinates are needed as input and thus the use of uncalibrated gaze points is sufficient. This approach is well suited for a range of applications like authentication and information display for a followed object.

This technique can - by itself - again be used to calibrate an eye tracker [45]. To do so a user is presented with a moving target on a display and an alignment of gaze coordinates and screen coordinates is derived based on the user's eye movement. This presents an easy and fast alternative to classical calibration. Nonetheless it is still limited, as a stimulus for calibration has to be explicitly generated and a connection between the eye tracker and the display is needed. An elegant solution to this would be to alter Pursuits calibration in a way that it can be performed implicitly (i.e. without the need to manually set-up a target) which is the very purpose of this thesis.

In the real world we are constantly surrounded by all sorts of moving objects that could be used for this purpose, be it cars passing by, people moving around or even television content when remaining at home. We envision all those objects to be possible targets for tracker calibration but also for a variety of new possible interaction techniques in the real world; rather than only with displays.

In contrast to current approaches this means, that we need to find possible moving targets that a user might follow instead of scripting them. Finding objects in an image is a highly non-trivial task, especially if the nature of this object is not known beforehand. To do so we propose to use the only known property of any potential interaction target: motion.

Optical Flow algorithms [9, 25, 30, 41] are a classical approach for tracking motion in an image sequence. We build our work around this and use it to generate trajectories for potential objects that can then be correlated with input from an eye tracker as proposed by Pursuits.

Another challenge of this work is the extraction of actual objects. While the approach described above will produce image regions or single feature points of an image that are likely to be pursued by a user's gaze, they are not able to determine the underlying semantic object. This task involves two steps: extracting an objects actual contours and determining what object it is. We omit the second step, as the nature of a tracked object is irrelevant for calibration. There are a wide range of specialized algorithms for extraction of known image content like humans [21, 69], foreground [24, 50] or pre-trained arbitrary objects [20, 68]. As in our case the object is not known beforehand though image segmentation techniques [1, 3, 4, 11, 26] are needed. We use this to segment the image into possible objects and choose one of them as our object proposal based on the underlying correlation.

As conclusion: With this thesis we explore the possibilities of using real world objects as interaction targets for Pursuits and contribute an algorithm to perform Pursuits selections with real-world objects based on video input. We evaluate the accuracy of a calibration based on our approach and suggest possible applications beyond calibration.

1.2 Overview

First, we introduce the background knowledge needed (*chapter* 2). This will be concepts and technical terms in the fields of eye tracking and tracker calibration, optical flow and object extraction. As there is - to our knowledge - no real comparable work as of yet, we mainly focus on finding relevant pieces of each approach and derive how we can utilize them.

The collection of our findings from related work and the conclusion of what is suited for our purposes is the main focus of the next chapter (*chapter* 3). We then derive our general approach. We also hypothesize about what application areas we see for this approach.

The following chapter will be dedicated to introducing our approach to realize Pursuits tracking with real world objects (*chapter* 4.1). We start by presenting our general pipeline, followed by a detailed description of the steps needed.

The previously described method aims to extract a whole object that a user is currently looking at. We realize that this is very computationally intensive and therefore introduce a second sparse version of our algorithm in *chapter* 4.2.

After that we introduce and discuss our apparatus to evaluate our approach (*chapter* 5). To do so we use pre-captured video material with hand coded ground truth and compare our calibration results to it. We also compare both our dense and sparse implementation of real world Pursuits to find out what situations both are suited for.

In the next chapter we discuss the results of our previous evaluation (*chapter* 6). We also try to find object properties that influence the quality of the generated output of our approach in order to derive both application areas and open questions with possible solutions.

We conclude our work with a short overview over possible improvements of our work as well as a list of suggestions towards possible applications that we envision to be realisable with our work (*chapter* 7).

2 Background and Related Work

In this chapter we will present a variety of background knowledge needed for the full comprehension of this thesis. This will include the basic principles of eye tracking as well as topics like optical flow and object extraction that are relevant for our approach of solving real world Pursuits.

2.1 Eye-tracking and Tracker Calibration

The basis of every eye tracking approach is a video source. Video-based eye tracking can be either used in combination with infrared based approaches (Infrared-pupil corneal reflection (IR-PCR)) or as a tracking technique on its own (videooculography (VOG)) [42].

The general approach (*compare figure 2.1*) is to find a face and - within that - the eye regions first. This can be done with a classifier introduced by Viola and Jones [68] or any derived technique. In the case of head mounted trackers, the step of finding a face can obviously be omitted. Within this region of interest the centres of the pupils are searched. There are several methods to do this, e.g. based on gradients [62] or isophote curvature [63].

The so found pupil position can now be used for calibration. This term describes a procedure where a user is presented with a number of targets (usually either 5 or 9) to look at. Then a mapping from pupil to target location is created, e.g. by finding a homography (transformation matrix).

This is already enough to achieve basic eye tracking. However, to make the approach more robust finding additional reference points in the face is beneficial. This can e.g. be surrounding edges and eye corners [11]. Due to that it is possible to track users head movements (to some extend). Otherwise the user would have to keep completely still in order to not disrupt the calibration.

2.1.1 Eye-tracking with Infrared Illumination

To enhance the results from video-based eye-tracking it can be beneficial to provide additional lighting. Traditionally infrared (IR) light is used as it is invisible to the human eye. Depending on the positioning of the IR light source (on or off axis) the lighting creates either a very bright pupil (comparable to the red eye effect in photos) or a dark pupil image [42]. In both cases it is easier to find the pupil in the image. The light source also creates a corneal reflection (glint) on the eye. The glint is used as a reference point to make tracking more robust. Furthermore in combination with the pupil centre the glint can be used as base for a calibration or to calculate gaze vectors.

2.1.2 Calibration-free Techniques

As calibration is a major obstacle for many scenarios (e.g. interaction in a public space) there are approaches to enable interaction without calibration.

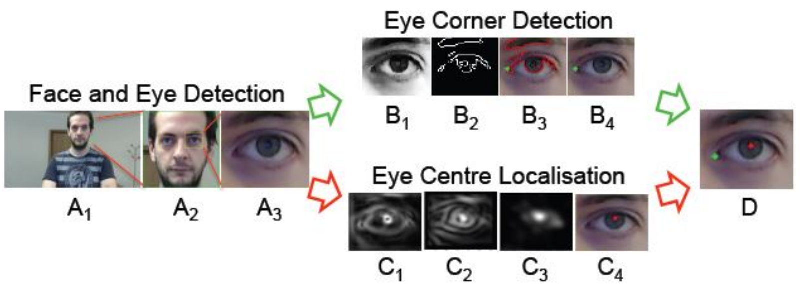

Figure 2.1: Video-based eye-tracking: Find a user's eyes (A_1-A_3), the eye corners (B_1-B_4) and the centres of the pupils (C_1-C_4). Finally use the extracted features (D) for gaze estimation. (Picture taken from [75])

Zhang et al. [76] introduce an approach called Pupil-Canthi-Ratio (PCR). It is based on a ratio between the pupil centres and the inner eye corners for both eyes. If a system is trained with this data it is possible to estimate horizontal gaze relatively accurate (3.9° of visual angle). This is enough to distinguish at least three directions and enable basic interaction [75, 77].

For use cases where only the information, whether an object is looked at, is sufficient, there are further approaches. This can for example be achieved by pure detection of pupils [54]. Smith et al. [56] propose another solution. They use basic image processing to find the eyes of users. Those eye patches are then classified by a support vector machine that was previously trained with direct gaze samples.

2.1.3 Pursuits

Based on the smooth pursuit eye movements [48] Vidal et al. [66] introduce a method for gaze interaction without previous calibration. It is based on moving interaction targets on a screen that the user has to follow with their eyes. Based on the correlation (traditionally Pearson Product-Moment Correlation Coefficient, short PCC is used) between eye movement and the trajectory of the target a selection can be made (*compare figure* 2.2). This approach is thus limited to selection and cannot produce a gaze prediction in cases where no object is followed.

Pfeuffer et al. [45] solved that by introducing Pursuits calibration. In this approach a stimulus is moved over the whole screen - in particular covering edges and corners - and a matching of screen points and pupil positions is generated by finding a homography. Pfeuffer et al. were able to achieve reliable accuracy of <1° of visual angle. Another finding was that they could calibrate users implicitly i.e. without them intending to calibrate or even notice.

Khamis et al. [35] built on the Pursuits approach and introduced TextPursuits. They use moving text instead of objects and make selections based on users reading this text achieving similar results to other Pursuits approaches. This work is particularly interesting for our approach, as it shows a first step beyond classical artificial approaches using more natural targets.

Sugano et al. [59] use Pursuits among other techniques to estimate attention maps for a display based on uncalibrated input of multiple users. They embed Pursuits stimuli in videos and show that they perform as well as classical 9-point calibration.

TraceMatch [17] and PathSync [13] are two approaches that use the concept of rhythmic path mimicry which is closely related to Pursuits. Here a selection is made by mimicry of an on-screen stimulus motion with arbitrary objects or body parts. Clarke and Gellersen show in a recent work [18] that it can be used to couple arbitrary objects to a screen and use them as pointers. AmbiGaze [65] uses the same concept to control an environment but applies it to scripted moving real-world stimuli instead of on-screen content. In contrast to our approach these works rely on a scripted stimulus for interaction and extract the users input from a video instead of extracting the moving objects and using gaze.

Santini et al. [52] propose a system for calibration of an eye tracker that uses a marker attached to a wall as a fixation point. Users have to move their heads to move the marker relative to their viewpoint. Then eye- and marker positions are matched to create the calibration. In our work we follow a similar idea but do not use static markers. Also this approach resembles explicit calibration whereas we aim for implicit calibration.

While the use of Pursuits, Pursuits calibration and similar approaches allow for a wide range of applications and accurate calibration result, they are bound to artificially created stimuli to function. With our work we aim to remove the need of pre-scripted targets, markers or even displays for the Pursuits technique to enable its application in a wider spectrum of settings.

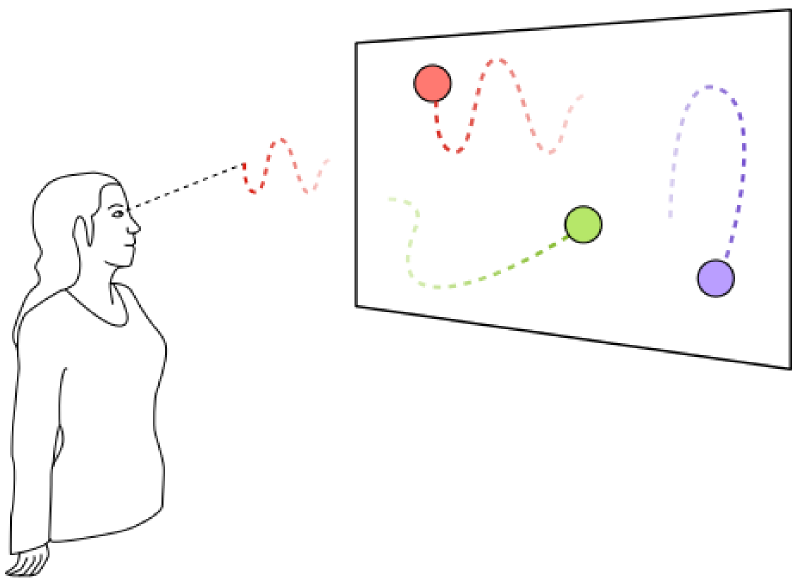

Figure 2.2: Pursuits: A selection is made based on the correlation of user's gaze movement and object trajectory. (Picture taken from [66])

2.2 Finding and Tracking Objects

The application of Pursuits to real-world targets requires being able to extract them from images or videos in some way. In this section we discuss different approaches how this can be done dependent on the objects properties, the input available and whether or not the object is known beforehand.

2.2.1 Object Recognition and Classification

Depending on the degree of abstraction that one aims for, object recognition can be as simple as finding an object by one of its properties, e.g. its colour or brightness (by thresholding appropriately).

Another option is to find predefined objects (e.g. faces) by training a model to recognize them. Possible approaches for this are the previously mentioned classifier by Viola and Jones [68], Histograms of oriented Gradients (HoG) [21] or more recently Support-Vector-Networks [20]. There are a range of other approaches, that build on this (e.g. [69, 74]) or a similar principle (e.g. [7, 40]). Further the use of depth images can yield strong results, as e.g. done for detecting humans with the Kinect sensor [71].

Lately also deep learning is used (e.g. [16, 36]). Many modern approaches go beyond mere recognition of objects and towards image classification. The difference is, that in recognition, images are searched for one predefined object, whereas in classification, one tries to name the content of an arbitrary image (i.e. recognize a wide range of different possible objects). This is also interesting for our work, as recognizing the semantic meaning of a detected object can enable further interactions, e.g. by triggering different actions when different object classes are gazed at.

2.2.2 Optical Flow

All previously mentioned approaches depend on knowing the object beforehand, be it for knowing its specific properties that are needed for recognition or for being able to train a model. In more general cases other methods are needed to determine the presence of an object (as it may be unknown what kind of object it is).

One property that is specifically well suited for the use in our approach is motion. As Pursuits is build upon matching object movement with eye movement, we can simplify the task of finding arbitrary objects to finding arbitrary *moving* objects.

A widely spread approach for detecting motion in an image sequence is called optical flow. The goal of optical flow is to approximate the movements in two consecutive images as a 2-D vector field. For each pixel in frame$_t$ a vector pointing to its location in frame$_{t+1}$ is given (with t being a time stamp).

Optical flow algorithms can be classified into several different categories based on their approach for flow estimation. Those include differential techniques, region-based matching, energy-based methods and phase-based techniques [6]. For our purposes however, we will mainly distinguish between dense and sparse algorithms, where dense algorithms are those that generate a vector field for the whole image whereas sparse algorithms only take in a set of features (e.g. points or edges) and calculate optical flow for those (*compare figure* 2.3).

The most prominent representative for sparse optical flow was suggested by Lucas and Kanade [41]. This might be partially due to its open source implementation as part of the OpenCV [31] framework, which also features the dense algorithm by Farnebäk [25]. More recent approaches are e.g. by Brox et al. [9] or the FlowNet (and FlowNet 2.0) frameworks [23, 30] which are based on neural networks and perform comparably to state-of-the-art approaches while being significantly faster. Another noteworthy approach is by Sevilla-Lara et al. [53] who model flow according to the type of object that it is applied to in order to better reflect the objects properties.

For clarification it should be mentioned, that optical flow by itself is of course not sufficient to find objects in an image or across several frames. It is merely a useful tool to find image regions that move in the same or a comparable way and can thus lead to deriving the underlying objects. Beyond that, optical flow can also be used to track previously detected objects.

2.3 Background Subtraction/ Foreground Extraction

Once an object has been found it is necessary to somehow process it. The output of most classifiers and recognizers is a bounding rectangle. Optical flow results in a vector field and the resulting extracted image region of interest is most likely fuzzy and noisy. In either case an exact approximation of object boundaries would be desirable.

One option to separate a foreground object from its surroundings is background subtraction. The task is to find out which areas belong to which of the two categories. The main assumptions are, that the camera is static and there are moving objects. One then takes a reference frame, also called "background image" or "background model" [46] and compares this to the frame in question. Changes are assumed to be foreground.

Finding the background model is the difficult part. It can change over time (e.g. lighting conditions) and is not trivially available as there might not be an only-background frame in an image

[1]Optical Flow Tutorial: `http://docs.opencv.org/3.2.0/d7/d8b/tutorial_py_lucas_kanade.html`

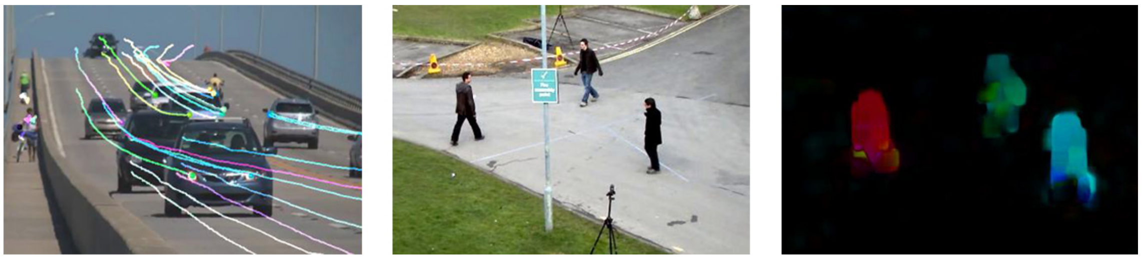

Figure 2.3: Optical flow methods[1]. Left: sparse flow by Lucas and Kanade [41], middle and right: dense flow by Farnebäk [25]

sequence. One solution is to model the background based on a probability model (e.g. Gaussian), accounting for minor changes but still detecting large changes that can then be attributed to the presence of a foreground object (e.g. [24]).

There are a range of further basic approaches of which we only want to address the Mixture of Gaussian (MoG) [57] method. It is widely popular and also has an implementation in the OpenCV framework. Here the background is not modelled by a single Gaussian distribution but rather a mixture of distributions (as the name suggests). It has proven to adapt reliably to long term changes and handle small background movements well.

The previously named techniques require a moving foreground object. This is something that e.g. Sun et al. [60] aim to overcome. Their approach FlashCut requires two images, one taken with a flash and one without. Due to the stronger illumination change for foreground objects they are able to extract those from the background.

A less restrictive and widely popular approach is GrabCut by Rother et al. [50]. They use user input to determine confirmed background (and foreground if provided) and build a model from that. The actual segmentation is then performed by minimizing an energy function optimizing both coherent grey level histograms and alpha distributions. This is then solved with a modified version of iterative graph cuts as suggested by Boykov and Jolly [8].

Though the results of the above approaches are very promising, all of them have major limitations. May it be that the camera has to be static, foreground objects need to be moving (not an issue in our case) or that user interaction is required.

2.4 Image Segmentation

A related, yet more general topic is image segmentation. Here we do not focus on separating foreground from background but rather segmenting the whole image into objects (much like with object recognition versus image classification). Both tasks are closely related and thus also share common approaches. An overview proposed by Pat et al. [43] can be seen in *figure* 2.4.

The two main approaches are to either base segmentation on discontinuities, i.e. changes in an otherwise homogeneous area or on subdividing the image in regions of high similarity with respect to some property e.g. colour or value.

2.4.1 Discontinuity-based Segmentation

The main approach for dissimilarity based methods is edge detection. One of the most frequently used techniques for that is based on a proposal by John Canny [11]. Canny describes his approach very formally and focusses on mathematically showing why it works, but in practice it comes down to the following steps: First the input image (which needs to be in grey value) is smoothed, e.g. by a Gaussian kernel. Then an intensity gradient is created by applying the first derivative to the image or approximated, e.g. with a Soebel kernel along both axis. Potential edges are reduced to single pixel width (called non-maximum suppression). Finally edge pixels potentially created by noise are filtered out and the actual edges are determined by applying an hysteresis. This is an approach using two thresholds T_1 and T_2. An edge is investigated when it contains a value above T_1 and is pursued as long as connected values are above the lower threshold T_2. This ensures that edges are preserved even if they contain single weaker pixels.

This approach is highly dependent on the choice of appropriate thresholds in order to create minimal edges with maximal expressiveness. If chosen too low the image will be noisy, if chosen too high, fine detail will be lost. One approach to tackle this is proposed by Lindeberg [38], based on the Scale-Space filtering approach introduced by Witkin [70].

Also, the Canny approach is limited to gradients in the grey scale image. However there are many possible edges that can not be derived like that, e.g. based on changes in texture. One recent approach to handle this was suggested by Dollár et al. [22]. They built a learning framework

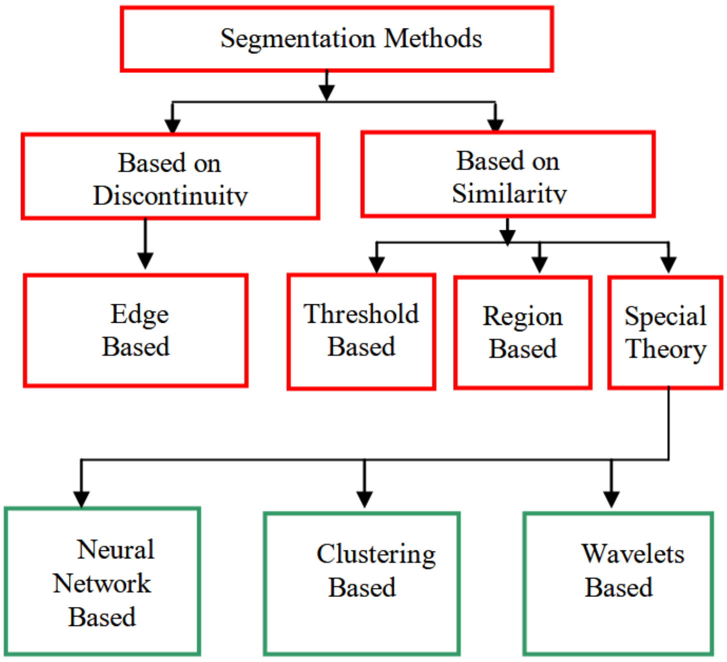

Figure 2.4: Overview over segmentation approaches as proposed by Pal et al. [43]

based on random decision trees which performs comparable to state-of-the-art while achieving higher speed.

Based on the determined edges, the image can now be segmented into regions by finding contours. This can for example be achieved by following the binary borders created, as suggested by Suzuki et al. [61]. One approach that omits the step of edge detection and is able to directly extract contours was suggested by Chan et al. [14] and is based on an initial curve. This is then refined until a stopping term is met. It is not based on colour gradients as e.g. in Canny's approach but rather related to a particular segmentation. A similar method, called Snakes, was proposed by Kass et al. [33]. It also uses an initial curve that is refined by applying forces that push it towards edges. Cohen [19] improved this method by adding an additional force blowing the selection up like a balloon.

2.4.2 Similarity-based Segmentation

In contrast to the techniques mentioned above, similarity based segmentation aims to combine image regions that share similar properties.

In the easiest case this can be done by applying a threshold to the image separating dark and light areas. Of course dark and light does not necessarily need to correspond to luminance, but can be a mapping onto an arbitrary property that can be expressed gradually.

Another option is choosing a seed pixel and growing a region around it based on a similarity criterion. The widely popular watershed algorithm [67] is based on this principle. Here the image gradient is interpreted as a topological relief and flood-filled in order to get segments. This can happen over the whole relief, starting at indicated seed points [2], or hierarchical. When the growing regions of two seeds collide a separating border is created, finally leading to a segmentation of the whole image. Zhu et al. [78] propose an approach for segmentation unifying region growing and Snakes.

Clustering is another widely used option for the segmentation of images (or any other data). In cases where the number of desired segments is known beforehand k-means can cluster the data based on any given similarity measure. This can also be used to generate SuperPixels [1] which are small clusters of similar pixels(basically an over-segmentation of the image). Merging those appropriately can again lead to solid segmentations as shown by Haris et al. [28]. Superpixels are otherwise generated by building a grid like structure and adapting it to the underlying edges [64].

As for background extraction, graphs can also be used for image segmentation. Carreira et al. [12, 37] propose an approach using iterative graph cuts to generate object proposals and assign them a likelihood to be objects. Similar to that Alexe et al. [3] suggest a measure, called Objectness, to determine the likelihood of a segmentation to be an object. Finally Felzenszwalb et al. [26] create segmentations based on a graph representation using a comparison predicate to decide if there is evidence for a border between two neighbouring segments.

Lately also neural networks and deep learning have been used to solve the task of image segmentation [5, 15]. For this networks are trained with hand-segmented images. Those are enriched with semantic labelling, allowing the algorithm to also output proposals for the content of the created segments (e.g. road or person).

3 Approach

In the previous section we introduced a wide range of techniques from the fields of eye tracking, object recognition and image segmentation. Many of the introduced techniques go beyond the scope of this thesis or have limitations that make it impossible to apply them to our work. In this chapter we will discuss which techniques are suitable for this thesis and derive a concept for our algorithm. Finally we hypothesize about possible applications.

3.1 Conclusions from Related Work

When using eye trackers, there are two options: either calibrate them or use techniques that work with uncalibrated gaze. Classical calibration approaches are not an option for Pursuits with real-world objects as they require a lot of time and a prepared calibration target. Both is unsuitable for spontaneous interaction. The use of uncalibrated gaze on the other hand comes with several drawbacks and, depending on the technique used, only allows for certain interactions (compare chapter 2.1).

We decided to use Pursuits calibration. This has two main advantages: 1) It can be used instantly for interaction with moving targets, as this does not require calibrated gaze. 2) Over the course of interaction, we can collect data in the background and use this for calibrating the tracker, enabling further interaction and gaze prediction in cases where no moving object is looked at.

When it comes to moving objects we have to distinguish four possible combinations of movement states. 1) A stationary camera and a stationary object, 2) a stationary camera and a moving object, 3) a moving camera and a stationary object and 4) a moving camera and a moving object. Of those four options, three can be used for Pursuits; given, that in 4) the movements are not identical and thus resulting in an identical frame sequence neutralising the movement in the video. Only if both camera and object are still, Pursuits is not applicable. We hypothesize this will rarely happen though, as we use a head mounted eye-tracker and thus assume the presence of slight head movements almost all the time.

Due to the choice of Pursuits we need to find moving objects that can be used for Pursuits. Recognizing objects as described in chapter 2.2.1 would be the most reliable method. Unfortunately this is not possible in our case as we attempt to build our system as general as possible. This means that we know nothing about the nature of the objects that we are searching for (except that they are moving). Thus there are no properties that we could use for recognition.

The only remaining property that we can use is the object movement. To detect this we decide to use optical flow. Here again we have two options: either use a dense algorithm or a sparse one. For our general approach this does not matter, however we can already determine possible advantages and drawbacks.

When using a sparse optical flow algorithm we will most certainly gain a lot of speed in comparison to a dense approach. However we will be highly dependent on finding the right features to track.

When using a dense optical flow algorithm on the other hand, we cover every possible feature point available and thus can guarantee that we find good matches if there are any. We buy this at the cost of high computational effort though.

We think that both ways have their advantages but decide to put our focus on using a dense algorithm. We do so, because this is the more fundamental approach and we see it more fit for showing that our idea works in the first place. We also believe that the computational effort will become less of a problem in the future. Approaches like FlowNet [23, 30] are a promising step in this direction.

We first aim to establish that our approach works. After that we propose the use of a sparse algorithm to make the system real-time viable. In the course of this work we will investigate both methods. To make them easily distinguishable we call them henceforth either *dense Pursuits* or *sparse Pursuits* by the optical flow approach they use.

As stated before (chapter 2.3), optical flow by itself is not enough to determine actual objects. In conjunction with Pursuits this only gives us either probabilities for single points (sparse Pursuits) or a probability map for the whole frame (dense Pursuits).

Background Subtraction is also built on moving objects and would thus be the obvious choice for extracting objects at locations with high probability. However in its basic form background subtraction requires a static camera which we cannot guarantee. We use footage from the world camera of a mobile eye-tracker and thus have to expect the camera to move constantly. In use cases with a stationary eye-tracker however this might be a viable option.

Approaches that do not rely on a moving object and static background like FlashCut [60] or GrabCut [50] would be interesting. However, the former requires flash images that we can not obtain and the latter requires user input. As this is limited to a bounding rectangle though, we keep this in mind as we should be able to generate that programmatically.

Another option is to segment the image and then match our probability distribution to a segment. There are a range of algorithms that might be fitting for this purpose so we decide to try several of them and make a final decision based on the results we get.

3.2 Realization

In this section we propose a possible pipeline based on the findings above. We keep this on a conceptual level, so that single building blocks could be exchanged based on specific use cases and needs.
The process consists of three main steps (*compare figure* 3.1):

- *Optical Flow*: First of all optical flow is applied to a given video source. The interchangeable options here are either a dense or sparse optical flow algorithm. When using a sparse approach (compare *chapter* 4.2), the trajectories for tracked features are a natural output of the algorithm. In case a dense approach (compare *chapter* 4.1) is used, an additional processing step is needed.

- *Correlation*: With given trajectories and gaze data we can calculate a correlation. For Pursuits the classical choice is the Pearson correlation (PCC) but in theory this block is completely interchangeable with an arbitrary correlation function that can generate probabilities.

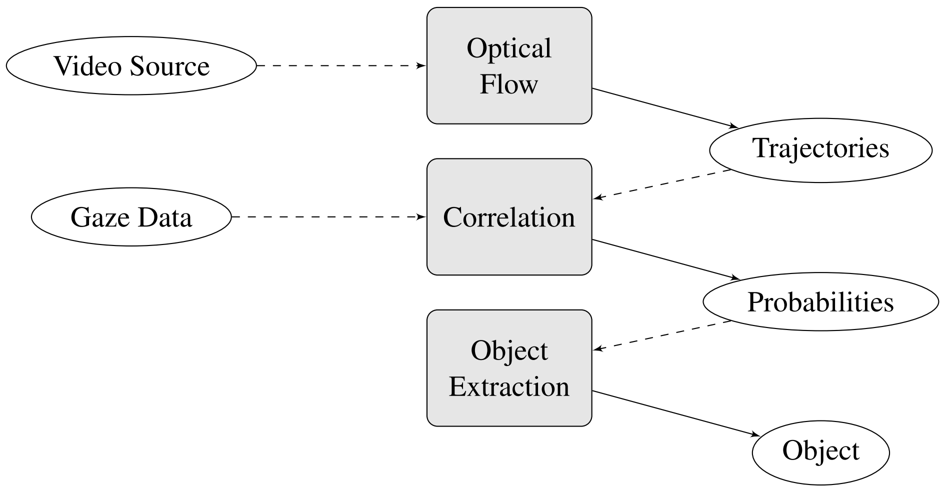

Figure 3.1: Generic pipeline. Left: input data, middle: processing blocks, right: output data

- *Object Extraction*: The final step is to extract the object that a user is looking at from the previously calculated probabilities. We test GrabCut [50] and a combination of image segmentation and nearest neighbour matching for this. In theory this block can again be replaced by any other object extraction method though.

Based on our previous considerations it would also be possible to move object detection to the beginning of the pipeline. In this case we would first detect all possible objects in an image. Then we would track their locations based on optical flow and correlate the results with gaze data. This approach only makes sense, if the objects are abstracted in the first step, as otherwise the results will be the same as for our proposed pipeline. In this case however, we would lose information (e.g. when the object is reduced to its centre). For this approach we want to use all information available (although we are aware that in practice simplifications would be appropriate and needed) and thus decide not to use this modified order.

3.3 Application

The main output of our algorithm is an object representation within a given frame. This can be used for a multitude of things. Our main focus for this work is to use it for calibration of an eye-tracker. The reason for this is threefold: 1) Calibration enables a range of additional interaction techniques that rely on gaze estimation. 2) When used for calibration, it is not necessary to know what an object is. Knowing its location is sufficient. Recognizing the objects found by our algorithm would introduce another level of complexity. 3) Calibration is a task that is very well suited for analysis. We can easily compare our results to a given ground truth in order to evaluate the accuracy of our method.

In our work we focus on calibration for mobile eye trackers. However this approach is also applicable for stationary eye-trackers with an attached display. In this case one would use the display content (e.g. screen capture it) as input for our algorithm. This sounds very similar to the set-up proposed in the initial Pursuits papers [45, 66]. The difference is that, with our algorithm, it is no longer necessary to script the targets on the screen. They will automatically be detected.

Another option would be the post-hoc annotation of captured eye-tracker footage. With our algorithm it is no longer necessary to calibrate the tracker beforehand and a user could start to instantly record footage. This could replace many unnecessary calibration procedures in settings where the created gaze data is not directly used for interaction. One limitation however is, that it only works with moving targets. The application areas would thus need to be chosen carefully.

Even without knowing what an object actually is, our algorithm can enable basic interaction. Given a defined set of gestures that are unlikely to naturally occur in the wild, a user would be able to create a stimulus by himself (e.g. moving their hand) and follow it to trigger an action (e.g. control volume or tracks of a music player). This could also be used for manual calibration, e.g. to recalibrate a tracker in fixed intervals in order to ensure high levels of accuracy.

Finally, when object recognition is applied to the output of our algorithm this allows for a wide range of additional interaction scenarios, e.g. (taken from [51]):

- Authentication: The selection of real world objects might authenticate a user. This could for example happen by looking at predefined objects in a PIN-like manner.

- Information: When the system can recognize the object a user is looking at it can easily provide additional context sensitive information.

- Establishing contact: In certain situations it might be desirable to establish contact to another person. If this is not physically possible, following the person in question with ones eyes provides an easy and natural way of establishing contact.

- Attention indication: There are situations where it is essential to know that a user has seen certain content. The systems of a car could for example warn the driver when they detect that she did not see a traffic light or important sign.

4 Algorithm

4.1 Dense Pursuits

In this chapter we introduce an implementation of our concept proposed in chapter 3.2. We start by a short overview over the algorithm and the alterations that have to be made to the general pipeline (*compare figure* 3.1). After that the single steps are explained in detail.

4.1.1 Overview

For dense Pursuits we alter the previously suggested pipeline (*compare figure* 3.1) at one point. As dense optical flow creates a vector field we need to extract trajectories from this manually. Also we add an additional step at the end for calibration. Thus the main steps are as follows (compare figure 4.1):

- *Optical flow:* We investigate different dense optical flow algorithms and discuss how they are to be used.
- *Trajectory map generation:* We introduce our method of generating trajectories by the reverse application of optical flow to the (in step one) generated vector field.
- *Correlation map generation:* We present the classical approach using Pearson as well as a possible alternative. Additionally we introduce a vectorized version to speed up this step.
- *Object extraction:* We present different approaches to extract an object from the generated probability map. We use thresholding, GrabCut and several other image segmentation methods. We then decide which one to use and justify our choice.
- *Calibration:* We discuss how to use the extracted object for calibration. In particular we compare a simple offset calibration and calibration by generating a homography. Also we talk about using the whole object instead of reducing it to a single point.

For our implementation, we decided to use the Python[2] programming language (version 3.5) as it is a quick and easy scripting language. Furthermore it is suited for complex matrix operations due to the NumPy[3] library. With SciPy[4] and OpenCV[5] it also supports two strong libraries for scientific computation and computer vision which is very useful for our purposes.

For this work we use a mobile eye-tracker[34] by Pupil Labs[6]. This is a tracker developed in Berlin and supported with open source software. In theory any other eye tracker could be used as well, if the communication interfaces are exchanged.

4.1.2 Optical Flow

The first step of our algorithm is the application of optical flow. As a preparation we retrieve a new frame from our video source, convert it to grey scale and resize it. The last step is necessary as we have high resolution videos as input (1280x720px). Dense optical flow and the following processing steps are computationally very heavy, so a lower resolution is needed to achieve reasonable computation times.

[2]Python: `www.python.org`
[3]NumPy: `www.numpy.org`
[4]SciPy: `www.scipy.org`
[5]OpenCV: `www.opencv.org`
[6]Pupil Labs: `www.pupil-labs.com`

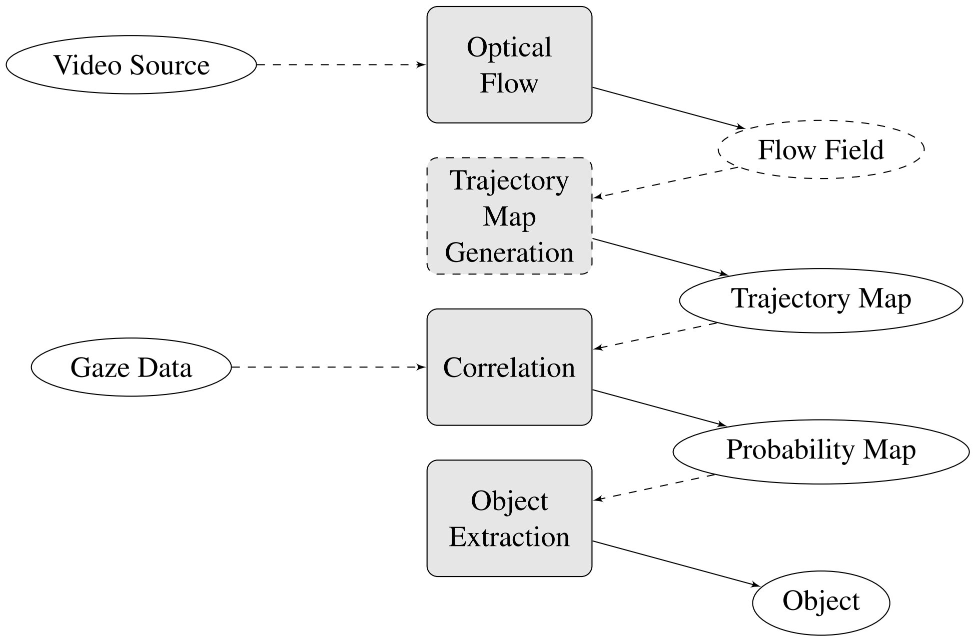

Figure 4.1: Pipeline for dense Pursuits. Left: input data, middle: processing blocks, right: output data. Added blocks are indicated by dashed borders.

Dense optical flow takes in an image pair I_1 and I_2 (both as matrices) and estimates the location of a pixel from $I_{1(i,j)}$ in the frame I_2 as $I_{2(i',j')}$. The output is a new matrix F as follows

$$F_{(i,j)} = I_{2(i',j')} - I_{1(i,j)} \tag{1}$$

In other words, each entry of the output matrix F holds a vector pointing from its associated pixel position in I_1 to the estimated position in the next frame I_2.

There are a variety of available algorithms for dense optical flow. For our work we first use the OpenCV implementation of Farnebäk's [25] algorithm. After some testing we decided that this was not ideally suited though. The reason is, that Farnebäk's algorithm seems to work well when tracking edge areas whereas it fails to properly depict movements inside object borders.

We decided to use a python wrapper provided by Pathk et al. [44] for an implementation that was developed by Liu [39] in the course of his doctoral thesis. It is based on the algorithm for dense flow estimation by Brox et al. [9] and the work of Bruhn et al. [10].

Changing the technique resolved the issue of uncovered object surfaces encountered with Farnebäk's algorithm and proved to provide more stable results (*compare figure* 4.2). This was bought at the cost of computational time, but we decided that for this proof-of-concept, seeking better results requiring higher computational times was acceptable. We suggest for future work to

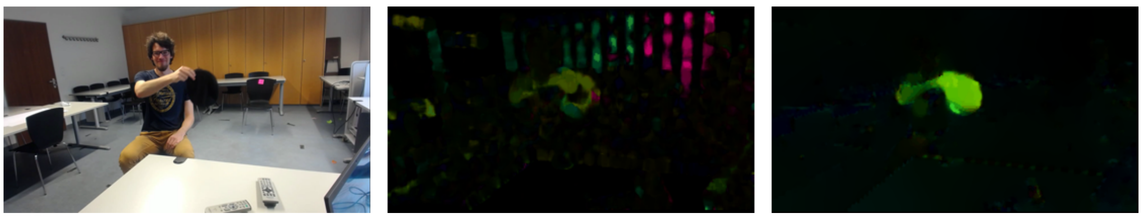

Figure 4.2: Comparison of Farnebäk [43] (middle) and Brox [9] (right) dense flow algorithms applied to an example frame (left). Colour represents movement angle and value magnitude. In comparison Brox produces a smoother and more stable result.

exchange this algorithm e.g. by a deep learning approach like FlowNet 2.0 [30] to increase the performance.

An option to ensure a high quality flow field is to apply back-propagation by using optical flow with swapped inputs. We call this *reverse optical flow*. The idea is to generate two flow matrices:

$$\begin{aligned} forward &= optFlow(I_1, I_2) \\ backward &= optFlow(I_2, I_1) \end{aligned} \tag{2}$$

We can now derive a quality measure q for each pixel $p_{(i,j)}$ by calculating

$$q_{(i,j)} = I_{1(i,j)} - backward_{forward_{(i,j)}} \tag{3}$$

In other words, we first calculate the position p' of a pixel p in the next frame I_2. We then calculate the estimation of p' for I_1 by applying reverse optical flow and compare the resulting location p" to the initial pixel p. Ideally the difference should be zero, so we can use the resulting value q as a measure of how far off the flow estimation is.

For our algorithm we tested this approach but decided to use a single pass optical flow without this optimisation. Although it improved results slightly, in our opinion it was not worth the effort of calculating optical flow for a second time and lead to unwanted artefacts. Also the finding of an appropriate threshold proved to be non-trivial and might have required further experimentation.

4.1.3 Trajectory Map Generation

Based on the created vector field we can build a trajectory map T_t for a given timestamp t holding a trajectory for each pixel in the image. The procedure is recursive and has the following base case T_0 for the first timestamp:

$$T_{0,(i,j)} = [(i,j)] \tag{4}$$

For any other timestamp T_t is constructed as follows:

$$\begin{aligned} reverse &= optFlow(I_t, I_{t-1}) \\ T_{t,(i,j)} &= T_{t-1,reverse(i,j)} \oplus (i,j) \end{aligned} \tag{5}$$

In other words, we retrieve the previous location (i',j') of a given pixel at location (i,j) by applying reverse optical flow. The current trajectory is then created as the trajectory at position (i',j') in T_{t-1} concatenated with the pixels own location (i,j). An exemplary step for a single pixel can be seen in *figure* 4.3.

To avoid the recursion we simplify the procedure by iterating the video file in temporal order and storing the last map T_{t-1}. Additionally we introduce a Window *w* that indicates the length of trajectory chain. This is done to account for the locality of motion. We are not able to match one

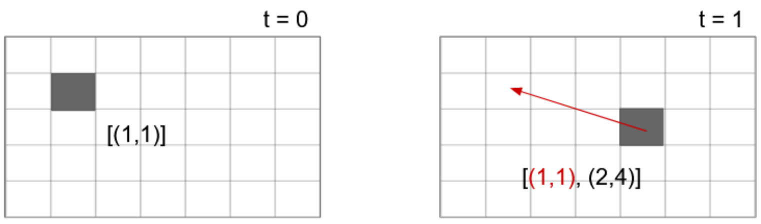

Figure 4.3: Example for the trajectory map generation: The trajectory for a pixel p is constructed recursively as the trajectory at its previous location appended by its own location.

object trajectory to gaze for the whole video but rather for short intervals (as gaze will also rest or follow different objects). By testing, we determined *w=20 frames* to be a good value.

4.1.4 Correlation Map Generation

The next step for Pursuits is to match gaze trajectories to object trajectories. As we have no objects given, we apply this on a pixel-base to the whole frame in order to generate a correlation map C. The suggested correlation function in the original Pursuits paper [66] is Pearson product-moment correlation coefficient, short PCC. For two trajectories X and Y of length n it is calculated as follows:

$$PCC = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}} \tag{6}$$

In the formula X and Y are two 1-D trajectories and x_i and y_i are their respective elements. The symbols $\bar{x}$ and $\bar{y}$ are the means of the two trajectories, defined as:

$$\bar{x} = \frac{1}{n}\Sigma(x_i) \tag{7}$$

The formula only allows for 1-D trajectories which means, that we have to calculate it twice for the two image axis. To combine the results we take the average of both axis. If the movement was only along one axis, it can happen that the above formula leads to a division by zero. In this case only the other axis is used. If there is no variance in both of the trajectories, we output zero.

The result is a number in the range [-1, 1] where 1 indicates strong positive correlation, 0 indicates no correlation and -1 depicts a strong negative correlation. For our purposes negative correlation is equally useless as no correlation at all, so we cut all negative values to zero.

During testing it turned out, that applying PCC pixel-wise produced the main share of our computation time, so we decided to vectorize this. The code for a vectorized version can be seen below:

```python
import numpy as np

def vectorized_pearson(T, gaze, w, h):
        n = T.shape[2] # Trajectory length
        # flatten to Vector of trajectories
        T = T.reshape(w*h, n)
        # pre-calculate sums
        sT = np.sum(T, axis=1)
        sGaze = np.sum(gaze)
        # denominator
        d = n*np.dot(gaze, T.T)-(sT*sGaze)
        # nominator
        n1 = n*(np.sum((gaze**2), axis=0))-(sGaze**2)
        n2 = n*(np.sum((T**2), axis=1))-(sT**2)
# correlation
corr = d / np.sqrt(n1*n2)
return corr.reshape(w, h)
```

We flatten the trajectory matrix T to an array of vectors of the same shape as the given gaze trajectory. We then apply the formula above (*equation* 7) and finally shape the result back to a matrix. Additional precautions are needed to ensure we do not divide by zero or apply the formula to trajectories of insufficient length, but we leave these out here. Using this approach we were able to speed up computation by a factor of about 100.

During testing, we realized that moving objects were creating a tail of high correlation. We attribute this to the fact, that a pixel behind a moving object shares parts of its trajectory, followed by not moving any more as the object has passed by. We concluded this effect could be softened by punishing trajectories (i.e. assigning lower correlation values) with multiple identical entries (no movement). The same applies to gaze, as it is an indicator for non-valid gaze data. By using this technique we could improve the quality of our estimation (compare figure 4.4).

It should however be noted, that with this approach we punish subtle movements as well. Those can contain identical values naturally and would thus get lower correlation values. We accept this drawback, because the technique improved our results a lot and we could not find evidence for subtle movements to be ignored over the course of our testing.

Theoretically PCC is not the only correlation function applicable here. We decided to try out a different correlation as well. This is based on the element-wise difference of the two trajectories in question. For a strong correlation the difference should be always identical and thus the difference points should form a dense cluster. We use the density of the resulting cluster as a means to determine the strength of the correlation (*compare figure* 4.5, right). When testing this, it turned out to be less robust than using PCC so we decided not to use it in our final approach. Nonetheless we would like to encourage anyone to try out different techniques for this step to adjust the results to their individual needs.

Possible correlation maps for a real scenario are shown in figure 4.5. The object followed by the user is clearly distinguishable. However we also get a high correlation for the arm of the experimenter. This is an expected outcome, as both arm and object move on a very similar trajectory.

4.1.5 Object Extraction

With the previously generated correlation map we can start to extract objects. The easiest method to do so is using a threshold on the map. The resulting shape is then treated with morphological dilation and erosion to fill possible gaps and remove small grained noise. An object prediction is then generated by finding contours on the created mask.

While this has the desired effect it comes with two downsides: 1) This selection method will often lead to multiple possible objects. This might be desired for other approaches, but for calibration we need one object to follow. 2) The object representation does not actually match the underlying object (*compare figure* 4.6).

One option to solve this is the use of a background subtraction technique. We decide to use GrabCut [50] as it does not require a static camera. We use a dual pass approach, first generate the required user input rectangle by calculating the bounding box around the thresholded correlation and secondly applying GrabCut. For this there are two options. GrabCut can be either applied to the source image or the optical flow field. While the first produces the desired object borders, the second works more reliably while creating weaker contours.

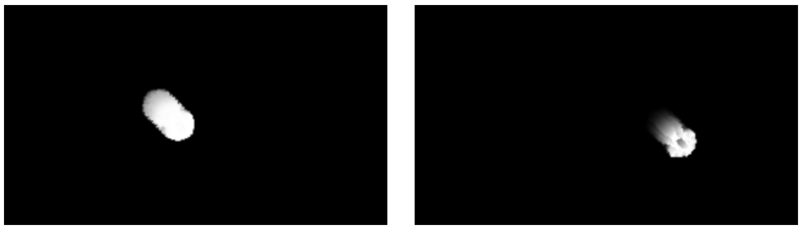
Figure 4.4: Results of correlation map generation on an artificial example. Left: previous version, right: after application of punishment for identical entries.

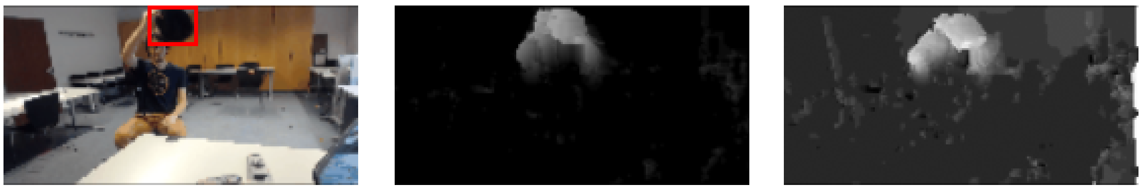
Figure 4.5: Correlation map applied to a real example. The users gaze follows the moving object indicated left. Middle: results using PCC, right: results using density method.

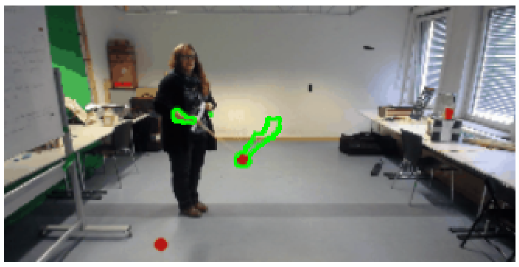
Figure 4.6: Object extraction based on threshold. The extracted object is indicated in green (hand and ball). The contours clearly do not match the underlying objects.

We decide to use the source image, as the goal is to create stronger object contours. It turns out that this approach can yield strong results and a significant improvement over pure thresholding when successful, but most of the time no detection is possible (compare figure 4.7).

The main problem seems to be that GrabCut is built for foreground extraction but has no means of properly incorporating our correlation map. Thus, the created selections are indeed foreground objects, but not necessarily the ones we wanted to select.

One possibility to circumvent this limitation is the creation of a segmentation and mapping of correlation to the respective segments. We decide to use Superpixel segmentation based on the SLIC algorithm [1] as it creates an over-segmentation and should thus be well suited for mapping the correlation.

We assign each segment the average correlation of the pixels covered. After that there is the option to threshold again or pick the strongest match. The latter comes with the problem of limiting object size, as superpixels are by nature of limited size. An exemplary output can be seen in *figure* 4.8.

It turns out that this approach works as well as the underlying segmentation does. For the low resolutions that we used this was not very accurate though, so we decided not to use it.

As mentioned before, having multiple objects as a result is problematic for our purposes. To solve this we decide to find the maximum correlation in the image instead of mere thresholding. As it turns out this works surprisingly well, although it introduces the new task of finding an object based on a single point.

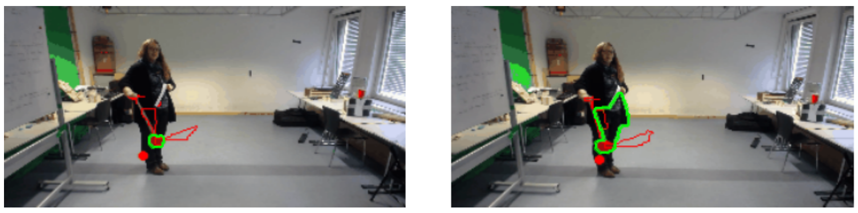
Figure 4.7: Successful (left) and unsuccessful improvement (right) of the object contours by application of GrabCut[50]. The threshold is marked red, the result green.

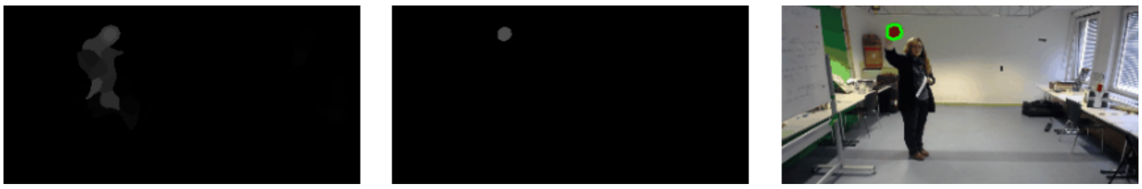

Figure 4.8: Superpixel segmentation. Mapping of correlation to superpixels (left), extraction of the strongest match (middle) and resulting object (right).

We solve this by segmenting the image with Canny edge detection [11]. Finding a good threshold for this algorithm is a very challenging task (as described in chapter 2.4.1). We use a solution proposed by Adrian Rosebrock in his blog[7]. It is based on the simple snippet of code below.

```python
import numpy as np
import cv2

def auto_canny(image, sigma=0.33):
    # compute the median
    v = np.median(image)
    # apply automatic Canny edge detection
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    edged = cv2.Canny(image, lower, upper)
    return edged
```

The method relies on using the median value of the image to generate thresholds. Those are not perfect, but achieve good results while being computationally inexpensive. This step could certainly be improved, but we do not find it necessary.

We use the generated edged image and find all contours. We then match the previously determined maximum value to the closest contour to generate an object proposal.

Contours are not necessarily closed so we continue with calculating the convex hull. This can lead to weak results due to the forced convexity, but during testing this turned out to be no problem. If the extracted object is to be used for more than calibration it might be worth the effort to try to improve this approximation, e.g. by approximating the shape with a polygon as suggested by Urs Ramer [47]. For our purposes however this is sufficient.

With this approach it can happen that a wrong contour is picked accidentally. To avoid this we introduce means of assuring the quality of our output objects. A simple option to do so is to compare the area covered by the approximation to the area covered by the thresholded correlation map. This excludes any major outliers. A better option is to calculate the average correlation in the area covered by the object proposal and accept it when a certain threshold is exceeded. Another option (that we did not investigate further) would be to iterate through the closest matching contours and test each of them to find a best match. This promises to be more reliable but also the computationally heaviest option hence we decided to use average correlation only. A successful object detection can be seen in *figure* 4.9.

Our main goal in the quality assurance step is to avoid false positives, as they can easily disrupt the calibration process. *Figure* 4.10 shows an example of a correctly rejected object proposal based on insufficient correlation strength. False positives can still happen, but they are most of the time

[7]pyimagesearch: `www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/`

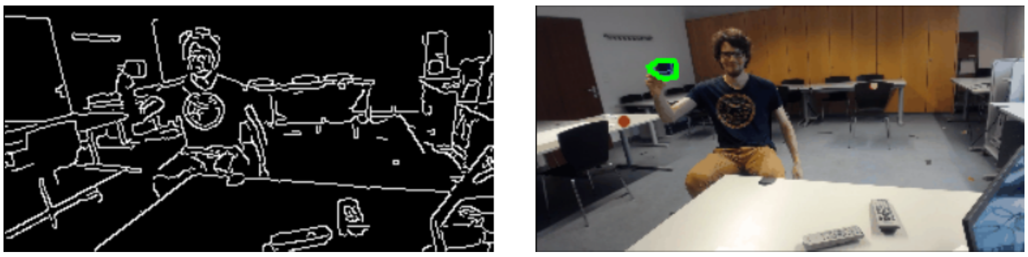
Figure 4.9: Successful object extraction with matching to "Canny-Segments". Left: edge detection with Canny, right: correct object proposal.

very close to the actual object (*figure* 4.10, right). For calibration this is sufficient. If the actual object is needed, we propose to use the iterative quality assurance measure mentioned above.

Overall we found the matching of maximum correlation to canny segments to be the best performing of the options introduced above, so we decided to use it for our algorithm. For other goals than calibration we highly recommend to investigate some of the above approaches further. Most of them are well suited for specific tasks and can thus be a better option in other scenarios.

4.1.6 Calibration

For calibration we reduce the detected objects to a single point. On first glance it might seem, that generating an object proposal from the maximum correlation is thus rendered superfluous, but it is not. While we can expect the maximum correlation to occur within the object bounds of the actual gaze target it is not necessarily the point a user is actually looking at.

One approach to find regions on an object that a user is most likely looking at is finding salient regions (e.g. [32, 72, 73]). Those are regions in an object that stand out, e.g. by contrast, colour or comparable properties, and are thus likely to be looked at.

During testing we found the results not suited for making reliable predictions. The approach that we used [73] didn't generate fine grained enough saliency to base an object-internal selection on it. For finding objects in the first place though (compare chapter 4.1.5), this might be another promising approach.

We decided to reduce the object to its centre based on the minimum enclosing circle instead. This produces good results for small objects and a rough approximation for large ones. For future work one might try to filter by object size to ensure strong calibration points. Also other methods could be investigated.

One approach that we find very promising would be the use of whole extracted objects. When keeping track of those we believe it to be possible to continuously refine a prediction to where a user is actually looking at.

Using the extracted prediction point we calibrate the eye tracker in one of two possible ways:

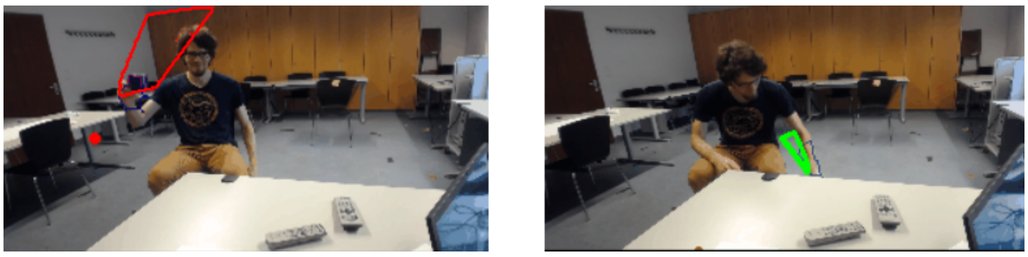
Figure 4.10: Quality assurance with matching to "Canny-Segments". Left: correctly rejected object proposal, right: wrongly accepted object proposal.

- *Global offset*: One very simple yet efficient method is the use of a simple offset vector. This describes a linear error measure between eye tracker output and the actual gaze location.

- *Homography*: A more complex model. A homography is a transformation matrix mapping one image plane to another. It is often times used for matching images, e.g. for panorama stitching. We can utilize it here for matching gaze points to detected object locations and thus create a transformation from one to the other. In comparison to global offset this can also handle scaling and rotation.

Our hypothesis is, that global offset will be well suited for a low amount of input samples whereas it will be outperformed by homography calibration once a certain number of calibration samples is reached. We evaluate this hypothesis in *chapter* 6.

Whenever an object is successfully detected we add its trajectory as well as a gaze trajectory of matching length to our calibration set. For the actual calibration process we use the OpenCV function *findHomography*[8] with RANSAC[27] for refinement. For global offset the averaged difference vector between object and gaze points is calculated.

When using this for longer sequences (beyond the length of our evaluation data) we propose to introduce forgetting of calibration samples. The purpose of this is to compensate for temporal changes, e.g. the eye tracker position changing due to the users movement. Thus it is possible to maintain a self adjusting calibration.

4.1.7 Summary

In this chapter we discussed our implementation of dense Pursuits. We create a map of trajectories for each pixel in an image by reverse application of an implementation of Brox [9] optical flow. Each trajectory is then correlated to the current gaze path using PCC. To avoid an unwanted behaviour where objects would create tails of high correlation in their movement path we punish correlation for trajectories with multiple identical entries.

Based on the created correlation map we extract objects from the image frame. We first segment the image using canny edge detection [11]. Then we find the maximum correlation (given that it is above a given threshold) and match it to the closest canny segment in the frame. To assure the quality of our selection we compute the average correlation of the chosen segment and accept the selection given it exceeds a certain threshold. To create a closed contour a convex hull algorithm is applied.

For calibration we reduce the extracted object to its centre and add the whole underlying trajectory combined with the actual gaze to our calibration set. Calibration is then performed by either computing a homography or a difference vector of the two point clouds.

4.2 Sparse Pursuits

As mentioned previously, the application of dense Pursuits is very computationally heavy and, as of now, not real-time viable. This is why we propose sparse Pursuits to compensate for this. As the approach is built to run in real-time some of the positive properties of dense Pursuits can no longer be achieved. The following chapter will explain which changes to the basic pipeline are needed and how we implemented sparse Pursuits.

[8]findHomography: `www.docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`

4.2.1 Overview

The main difference between dense and sparse Pursuits is the choice of optical flow algorithm. For sparse Pursuits we choose a sparse algorithm, i.e. an algorithm that calculates flow for single feature points instead of the whole image.

As a consequence we need an additional step of finding appropriate feature points and manage them properly. This covers in particular how feature points should be generated, when to re-sample and when to discard old features. To handle all of these tasks we introduce a structure that we call *Hypothesis (h)* and that represents a possible matching point.

Another changed point is the handling of successful matches. As we do not have a correlation map there is no good way to assure a correct object is selected. We thus renounce the attempt and generate calibration points differently.

A coarse pipeline can be seen in figure 4.11. We are inspired by a classical particle system. The Hypothesis object is our central storing unit for any data related to a certain trajectory (i.e. a single particle). New Hypothesis are generated whenever new features are detected and updated using optical flow in every iteration. The updated trajectories are then used to compute PCC and find possible matches based on high correlation. Finally we use heuristics to determine if we have a matching Hypothesis to use for calibration and to discard old particles which scored low correlations over a longer amount of time.

4.2.2 Candidate Generation

The first step of our approach is the generation of candidates. At this point we basically take a non-deterministic guess at where a user might be looking and then try to verify it by following that location in the image and correlating the trajectory with gaze. As we have no actual clue where the user might be looking at (we assume uncalibrated gaze) we have to guess as well as possible.

The most straight-forward heuristic would be to try and cover as much of the image as possible. This leads to a trade-off between having enough coverage to safely assume we also cover the correct features and oversampling at the cost of performance. The extreme case would basically be equivalent to a dense optical flow approach.

To avoid oversampling another option is to mask areas already covered by feature points. In conjunction with a minimal distance between feature points this can lead to a well covered screen with a sparse set of samples.

An alternative tailored to our approach would be to sample from areas covering motion. This does however imply that we would need to generate a dense flow field which is the very thing we try to avoid.

When we take the correlation from previous iterations into account there is another option. As we expect the real gaze point to be situated in an area of high correlation it can be beneficial to add additional feature points in such areas. This allows for more precise localisation of possible matches.

We decide to sample from the whole screen using a mask to avoid previously generated feature points and a minimum distance between points. Additionally we generate feature points in areas of high correlation (circular area around highly correlated feature points).

To find features we use the OpenCV function *goodFeaturesToTrack*[9] which is based on the algorithm suggested by Shi et al. [55]. Testing showed that we achieve the same results using the FAST [49] algorithm (Function in OpenCV: *FastFeatureDetector_create*[10]) while being significantly faster, so we use that instead for our final version.

[9]goodFeaturesToTrack: `www.docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html`

[10]FastFeatureDetector: `www.docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html`

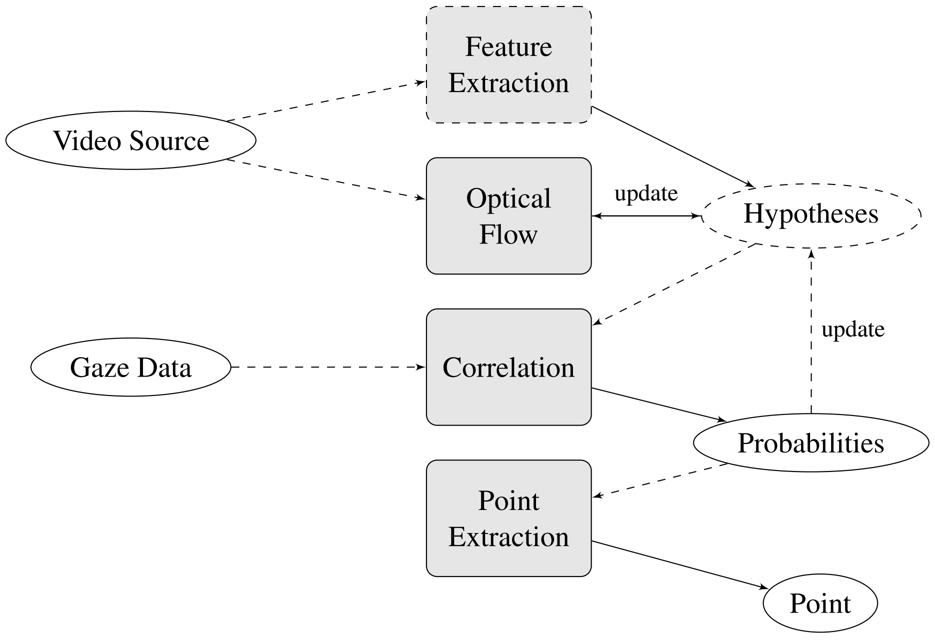

Figure 4.11: Sparse pursuits pipeline. Left: input data, middle: processing blocks, right: output data. Added or altered blocks are indicated by dashed borders.

4.2.3 Hypothesis Management

As mentioned before we store information related to trajectories in an dedicated particle-like object that we call Hypothesis.

In an Hypothesis object we store the following data:

- *Track:* Here we store the trajectory of the associated feature point.
- *Gaze:* This field contains gaze points for each feature location that is stored in Track. Together they are used to calculate correlation.
- *Score:* We store previous correlation values and determine a score as average over those. This ensures that a Hypothesis achieved constant good correlation over a longer time period of time before it is considered a possible match.
- *Life:* We use life as a measure of how useful the respective particle is. When the given Hypothesis scores weak results (low score values) we decrease its lifetime and 'kill' the particle once it reaches zero.

In addition to that we provide functionality to update the particle with new gaze and position data and to compute the correlation.

We introduce the lifetime mechanic to ensure that we do not waste computation power on particles that are not correlated to gaze. One problem that can arise though is 'mass dying'. As the first batch of feature points is generated simultaneously (when the script execution starts) they also 'die' simultaneously given that they do not correlate. This cycle then repeats as new feature points are generated to replace the old ones. We solve this problem by introducing randomized additional lifetime.

A special case occurs when the tracking for a feature point is lost. For this case we provide an additional function that allows to directly remove the Hypothesis without having to decrease its lifetime first.

4.2.4 Matching Feature Extraction

For each new frame an update cycle for the Hypotheses is started. The trajectories for each particle are updated using the sparse optical flow approach by Lucas and Kanade [41] (Function in OpenCV: *calcOpticalFlowPyrLK*[11]). We apply back-propagation (compare *equation* (3) in *chapter* 4.1.2) to determine when the tracking was lost. In contrast to dense flow a second pass with our sparse algorithm is not expensive so that we can effectively use back-propagation.

Using the new positions we calculate PCC for each Hypothesis. Based on that we adjust the internal scores and, if necessary, decrease lifetime. Possible matches are stored for later processing. After that we re-sample particles that have no remaining lifetime.

We apply several further steps to ensure the quality of possible matches and derive a good approximation of the point followed by the user. Those steps are as follows:

- *Filter for outliers*: We eliminate outliers as a first step. We determine the average over all possible points and remove everything farther than 1.5 times the distance covered by the middle 50% of the points in question.
- *Exclude border cases*: Testing has shown that tracking with our sparse algorithm leads to unstable results in the border regions of the frame. We exclude such points.
- *Pick best matches*: From the cleaned dataset we pick the best matches. Through testing we found five matches to be a good amount. If there are fewer matches at this point we ignore this step.
- *Hierarchical clustering*: To ensure that all possible matches belong to the same real-world object we apply hierarchical clustering (SciPy function: *fcluster*[12]). We only consider the points for calibration if there is only a single cluster. Otherwise we assume there are two possible objects and we cannot decide which one to pick.

This gives us a set of up to five points that should belong to a single object and are strong matches. We use those to estimate a final point for calibration. Again there are several options:

- *Pick best*: The easiest method is picking the best match out of the selection of up to five possibilities. This has the advantage of relying on an actual feature point in contrast to the following options.
- *Pick average*: Under the assumption that the detected features spread across the borders of an object, taking the average location gives a good estimation for the object centre. This requires that for the calibration trajectory past positions have to be interpolated as well which can lead to weak results.
- *Pick weighted average*: The average approach can be combined with the best match. Taking the weighted average using the single features individual scores as weights results in a centre point that tends towards strong matches.
- *Pick all*: Instead of making a decision at all we can also take all of the possible features for calibration. Due to the previous optimizations we know there are only few of them and they are reasonably close together.

[11] Optical flow by LK: `www.docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html`

[12] Hierarchical clustering: `www.docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.cluster.hierarchy.fcluster.html`

After testing all of the options named above we decide to pick all samples. Generating more samples generates more stable results when using global offset for calibration. Unfortunately we could not achieve a well working version using homography for calibration. We assume that we made a mistake in the implementation but were unable to find it.

A visualization of the generated results can be seen in *figure* 4.12. Features points are coloured according to their score where red corresponds to low scores and green to high correlation. A successful selection of the hand is made using three matching feature points. Similar to dense Pursuits we see strong correlations in other than the intended areas due to them moving similarly.

4.2.5 Summary

In contrast to dense Pursuits, sparse Pursuits is a more lightweight approach to perform Pursuits with real world objects. Instead of applying dense optical flow to create the trajectories we use the sparse algorithm of Lucas and Kanade [41] on a set of feature points determined using FAST [22].

For feature point selection we aim for a good coverage of the whole screen without overly dense crowding of feature points. We use a minimal distance between feature points to achieve this and constant re-sampling in order to not miss features that recently entered the frame.

To manage feature points easier we use a particle-like approach and store relevant information for each particle in an object that we call Hypothesis named for its potential to be the feature we are looking for. All Hypotheses are correlated to eye gaze using PCC. We determine an overall score for each respective particle as a mean of its correlation over time to ensure constant high values and choose a set of highly scoring features as potential object representatives.

From this set we remove outliers, pick the five best matches and apply hierarchical clustering to ensure all points belong to the same cluster i.e. the same object. Additional Hypotheses are created in the respective area to allow for more precise localisation of the object. All determined points are then used for calibration analogue to dense Pursuits. Calibration using homography turned out to be potentially error prone but we will evaluate it together with the other options in the following chapter anyway.

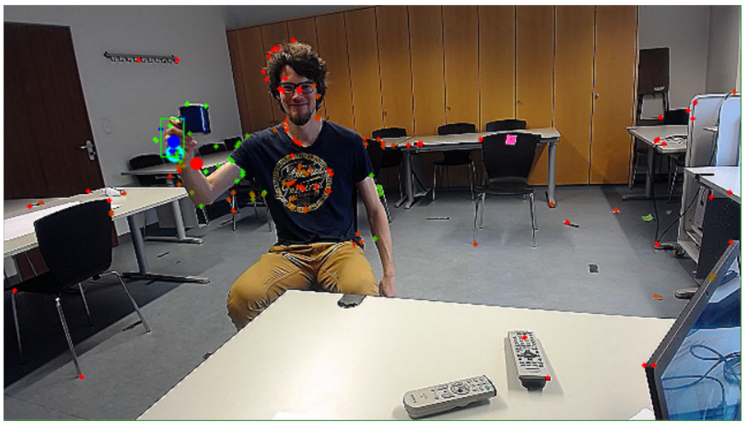

Figure 4.12: Selection with sparse Pursuits. Feature points with high scores are indicated in green, low scoring features in red. The blue dot indicates a successful selection.

5 Evaluation

In the previous chapters we introduced our implementations of dense and sparse Pursuits to enable Pursuit calibration with real world objects. This chapter is dedicated to evaluating our success in doing so as well as answering our research questions beyond calibration. We first introduce the data that we use, then talk about the research questions that we try to answer and finally elaborate our apparatus to do so. The results are presented in *chapter* 6.

5.1 Data

For our study we use recordings of eye tracker footage as dense Pursuits is not real-time viable. Our main data source are the videos generated by Katharina Sachmann [51] in the course of her master thesis. Those consist of two datasets (compare *figure* 5.1, top):

- *Pre-Study*: This dataset consists of gaze recordings by 24 participants. They were presented with a red ball as a target and asked to follow it with their eyes *(D1)*. Sachmann used this to investigate applicability of real-world recordings for Pursuits interactions. In contrast to our work she coded this manually though. We use her coding as a ground truth for the object location in our evaluation.

- *Use-Case Study*: This dataset consists of gaze recordings by 12 participants *(D2)*. They were presented with 3 scenarios that were each iterated two times:

  Stationary waiter scenario: The user was asked to follow a moving researcher with their eyes while remaining stationary. The setting of this scenario was a restaurant where the waiter would be notified when a guest would follow them with their eyes.

  Moving waiter scenario: This scenario is very similar to the one above. However this time the participants were asked to move while following the moving waiter with their eyes.

  ATM scenario: In this scenario the participants were asked to fixate on stationary targets while moving themselves. The story was a novel technique to enter a pin by looking at real-world objects while moving towards an ATM.

We use the Pre-Study dataset as our main means to validate our approach. It is well suited, as it uses an easily distinguishable target object which is moved according to pre-defined trajectories.

The second dataset is interesting as well, as it covers all the ways user and object can move. The target is however harder to detect as participants were asked to look at the researchers head. This moves similarly to a persons body which has the potential to make selections with our approach hard.

Beyond those two datasets we use two further datasets for that we do not have a ground truth available. They have more versatile content and will thus be used for qualitative analysis instead (compare *figure* 5.1, bottom):

- *Arbitrary objects*: For this study we recorded the gaze data of four participants. We presented them with arbitrary every-day objects that were moved in circular and rectangular trajectories *(D3)*. The goal was to determine object properties that have an effect on the quality of the created calibration as well as investigate how well we are able to determine contours of arbitrary objects.

- *Every-day situations*: This dataset consists of several every day activities like driving and moving indoors and outdoors generated by Steil et al. [58] *(D4)*. In this case we neither have a ground truth nor clear objects that the participants followed. Using the calibrated gaze data however we can make assumptions as to where the participants were looking and qualitatively evaluate if our method could work in such a completely non-restricted scenario.

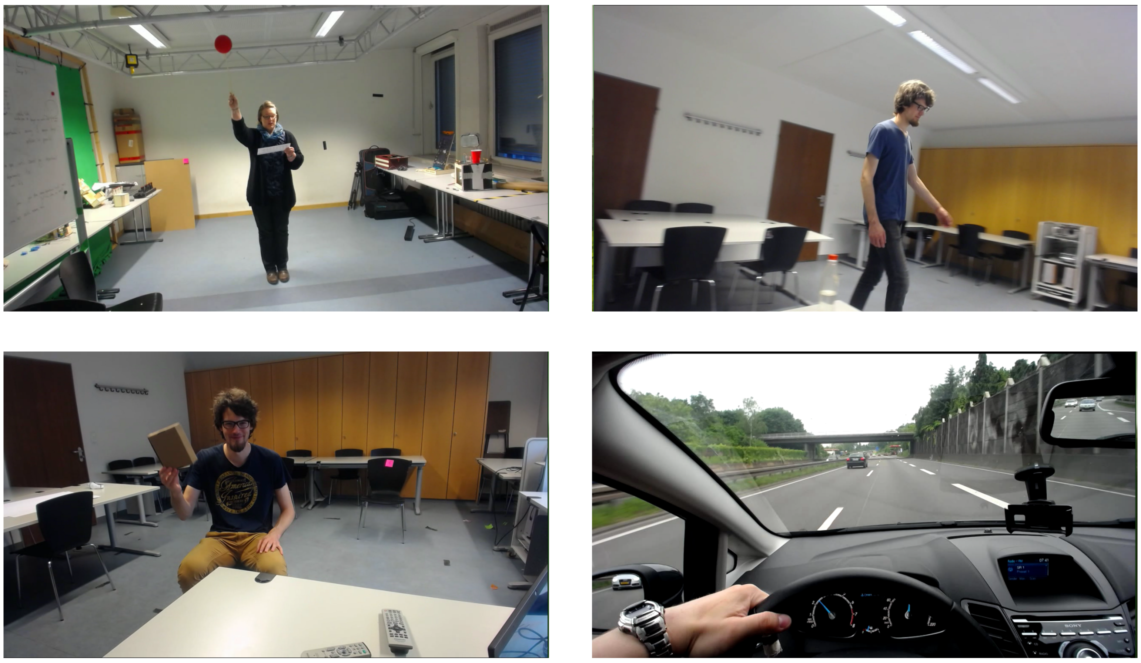

Figure 5.1: Examples from our data. Top left: Pre-Study. Top right: Use-case study with stationary user and moving waiter. Bottom left: Arbitrary objects. Bottom right: driving participant from the every-day situations dataset.

5.2 Research Questions

With our evaluation we aim to answer several research questions:

- *Q1:* Are we able to successfully calibrate an eye tracker using dense Pursuits and how accurate is our prediction? We investigate this question with respect to different resolutions and the calibration method as we hypothesize those might have a large impact on the results.
- *Q2:* How do results for sparse Pursuits compare to dense Pursuits? Our aim with this question is not to discard one of the two approaches but rather find out in which situations they perform best and what the advantages and disadvantages of each technique are.
- *Q3:* How does our approach perform in a more natural setting and are we able to calibrate with actual unscripted real-world objects? We test whether our approach can handle different relations of object and user. In case that it does not we investigate the reasons and ways to resolve them.
- *Q4:* Is our approach for real-world Pursuits applicable beyond calibration? To answer this we explore our results using arbitrary objects and try to determine object properties that hinder or support our ability to correctly determine object selections.

5.3 Apparatus

To answer our research questions we apply a range of evaluations on the given data. The main method to determine the quality of a calibration process is the mean accuracy in degree of visual angle. We determine this for the dataset *D1* to answer question *Q1*. To gain a better insight in the factors affecting accuracy we investigate different resolutions for both of our calibration methods. Furthermore we explore the impact of calibration samples used and area covered by calibration samples as well as the quality of the samples taken.

To answer question *Q2* we apply our sparse Pursuits algorithm to the same dataset *D1* and compare the results to our previous findings. Calibration with homography might not be possible so we focus mainly on global offset. We nevertheless evaluate homography as well to back this.

Our research question *Q3* can be answered by applying both of our algorithms to dataset *Q2*. This dataset covers all possible relations of object and user movement and presents a more natural setting. We will need to be careful though, as we will have to find out which of the changed factors potential stronger or weaker results can be attributed to.

Finally we use the datasets *D3* and *D4* to answer our last research question *Q4*. As we do not have ground truth data available for these datasets the nature of our analysis will be qualitative. We mainly aim to find effects of colour, contrast, detail, setting and gaze target on our approach. We investigate those for both dense and sparse Pursuits as we hypothesize that they will perform differently with respect to different properties.

6 Evaluation Results

In this chapter we execute our evaluation apparatus presented in chapter 5.3. One should reference chapter 5.2 for the research questions and data used. We discuss the outcomes and possible implications and derive further tasks for future work that is discussed in chapter 7.

6.1 Data Preprocessing

For all cases with ground truth data available we limit our sampling of data points based on three conditions:

- After a first pass on dataset *D1* we determined the mean accuracy of calibrated gaze to be 7.53° of visual angle (SD 5.54°). We determine the border for outliers as 2.5 times the standard deviation which is 21.40° of visual angle (compare *figure* 6.1).

  Based on that we choose a threshold of 21° of visual angle and assume for gaze points farther away, that the participant was not actually looking at the specified object.

- From the eye-tracker we use, we get a confidence value indicating the certainty that a given gaze estimation is accurate. The estimations are given in the interval [0, 1] where zero indicates cases where the eye-tracker can tell nothing about actual gaze, e.g. due to blinking. In their documentation the manufacturer proposes a threshold of ~0.6 to get good results[13]. We follow this recommendation and exclude samples with a confidence below their proposed value.

- As a final criterion we exclude cases where our model predicts a gaze point outside the frame borders. In contrast to the actual accuracy of our prediction this can be determined at runtime and we assume that any real application would exclude such predictions.

The participants for datasets *D1* and *D2* had breaks between tasks. This means that the data is only partially annotated with baseline predictions in the sections where the users had to actually follow a moving target with their eyes. Due to that we limit our sampling of data points to sequences with baseline data present if not specifically stated otherwise.

After some initial runs, we notice that two of the files from our dataset produce significantly (t-test: p<.001) more samples than the others (compare *figure* 6.2).

Inspection shows, that those two video files were recorded with 30 frames per second (fps), whereas the rest is recorded with 15 fps. We decide to exclude those two files from our analysis, as they might produce misleading results for high sample counts.

Our generated data is very jittery, as it is based on recorded gaze data which is never perfectly stable. *Figure* 6.3 (left) depicts an example from our data. It shows gaze and prediction accuracy over time. To allow for visual analysis we process this raw data and display a running mean instead (figure 6.3,right). This means for every point in time we display the mean over all previous samples.

If not specified otherwise graphs in our analysis can be assumed to be based on the running means over the accuracy data.

Also it should be noted that we label the x-axes in *figure* 6.3 with 'time' although we only plot the samples in temporal sequence. We would like to clarify that we display only samples that we have a ground truth for. There may actually be additional samples between those that we do not include in our analysis as we can not measure their accuracy.

[13]Pupil Labs documentation: `https://github.com/pupil-labs/pupil-docs/blob/master/user-docs/data-format.md`

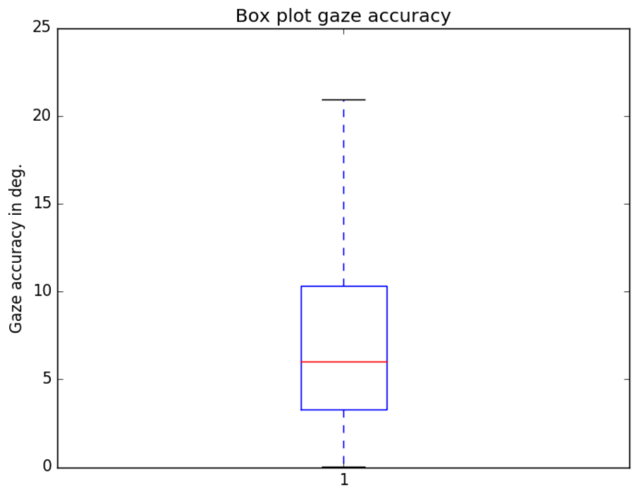

Figure 6.1: Accuracy distribution for calibrated gaze on dataset *D1*.

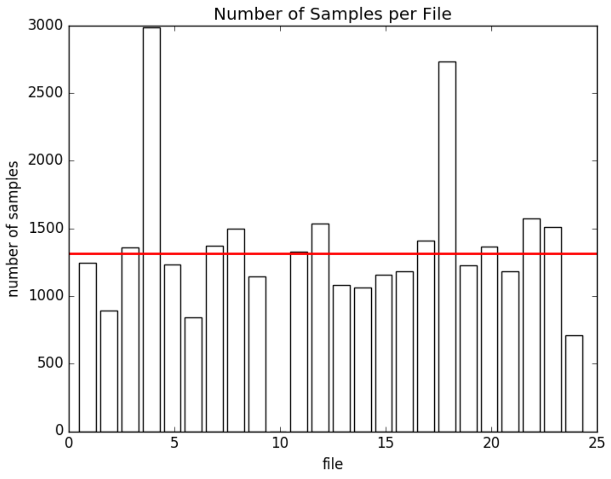

Figure 6.2: Number of samples generated during analysis of files from Dataset *D1*. Mean is given in red. Two files produce significantly more samples.

6.2 Dense Pursuits Evaluation

The first question that we want to answer is whether dense Pursuits generates reasonably accurate results and under what settings it performs best. To do so we compare both homography and global offset calibration for different input resolutions.

6.2.1 Prediction Accuracy

First we investigate the effect of resolution on the accuracy. *Figure* 6.4 shows the mean accuracy over multiple resolutions for calibration with homography.

For each instance from Dataset *D1* we plot the running mean over the accuracy i.e. the difference to the ground truth object location. The mean over all those curves is plotted bold. Gaze accuracy is indicated in red and remains stable over all resolutions as would be expected as this is an input rather than a computed value. It should be noted, that we combine time series of different length. To compensate for this we only plot mean curves as long as they are supported by at least

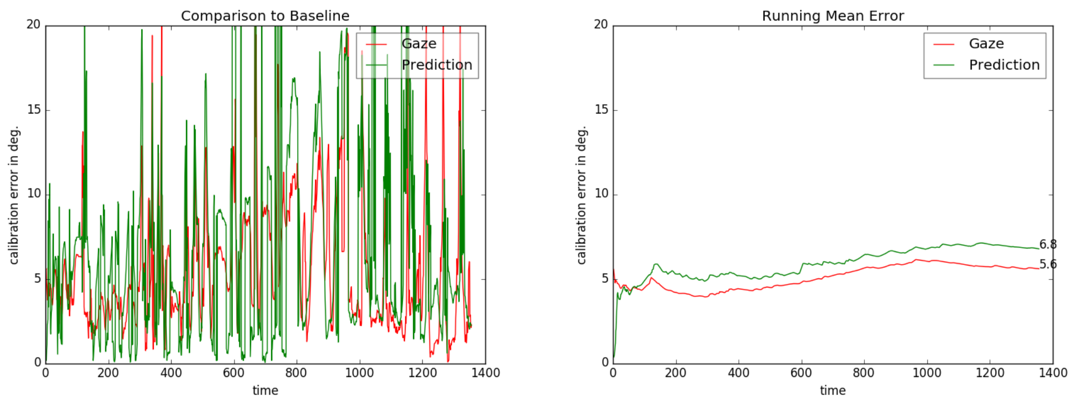

Figure 6.3: Difference between baseline and both gaze(red) and our prediction(green) over time. Left: raw data. Right: running mean of the raw data.

three samples. Nonetheless it has to be expected, that values for later points in time have less support and thus trends in the last section should be handled with caution.

The blue curves resemble calibration using the baseline. That means whenever there was a ground truth object position available we add this to our calibration set and base our prediction on that. In theory this gives an upper bound for the accuracy that we could expect under perfect conditions. From the plotted curves it quickly becomes clear, that this measure is not useful here. The single instances create as it seems random patterns and also the overall mean (bold) performs weaker than the actual accuracy that we are able to achieve. We attribute this to a lack of calibration samples. Using only baseline points generates way less samples than adding a whole trajectory whenever an object was detected.

The green lines show our prediction accuracy using dense Pursuits. For both the single instances as well as the overall mean accuracy we see a slight tendency to better results both over time and for larger resolutions.

The graphs for 100x50px and 200x100px both show a dent at about t=750. Further investigation shows that this can be attributed to a single outlier with an average gaze prediction error of 18.3° of visual angle. While in general we do not rely on a calibration it is indeed problematic when the gaze estimation is completely off and does not follow the overall movement trend of the participants eyes. We think this is no general weakness of our approach but something that should be kept in mind.

Analogue to *figure* 6.4, *figure* 6.5 shows the mean accuracy over multiple resolutions for calibration with global offset instead of homography.

In this case the baseline calibration behaves as expected and indicates that in theory we would be able to achieve better results than calibrated gaze. In our opinion this result should be handled cautiously though, as offset is closely coupled to the quality of the gaze estimation and cannot compensate for effects like warping and scaling over the screen space. Nonetheless the graphs show that this approach can produce strong results. We encourage further testing on real world applications to show if our doubts are valid.

While the expected baseline accuracy remains constant over different resolutions (as should be expected) our prediction (indicated in green) approximates this value for higher resolutions. For 50x25px the prediction error remains almost constant over the whole calibration period but for higher resolutions we observe an improvement of prediction accuracy over time.

As stated before the final values of our plotted graphs might be misleading as they have weak support so we decide to measure overall accuracy differently. All participants recorded in *D1* were presented with the same moving stimuli. Although calibration with dense Pursuits results in different amounts of samples due to participant behaviour this enables a comparison based on the

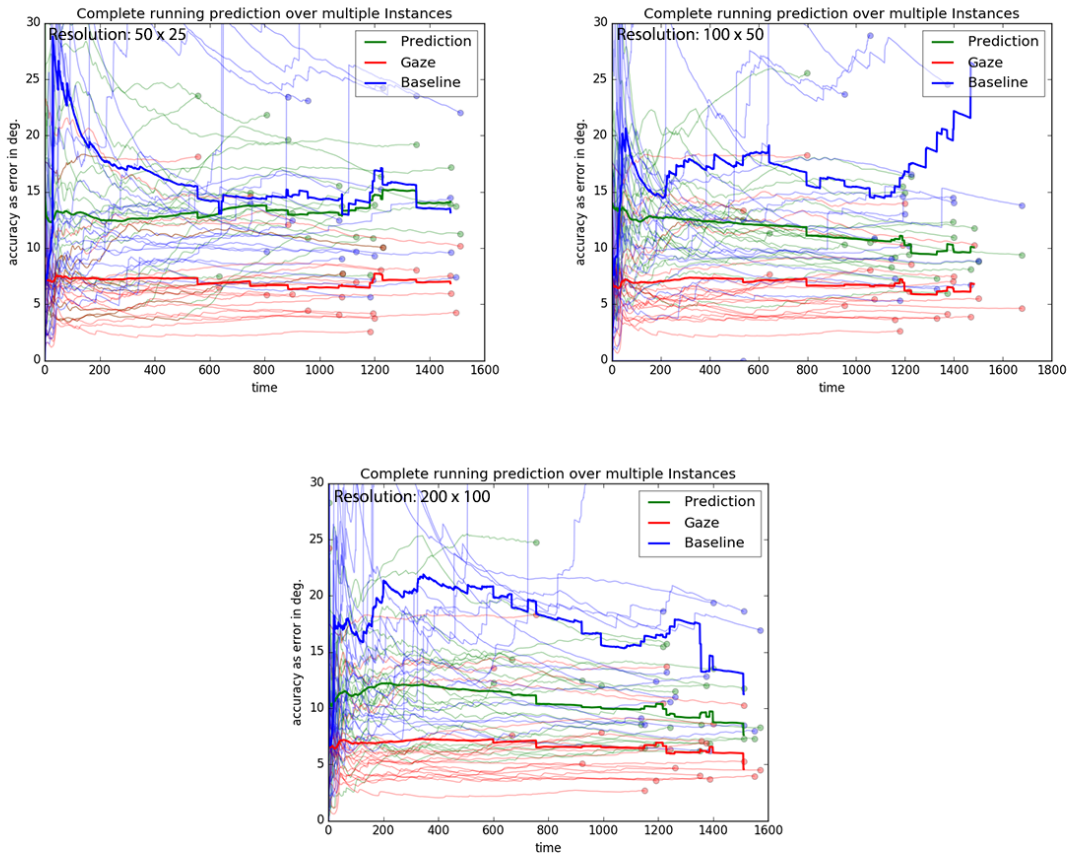

Figure 6.4: Running mean prediction accuracy over all instances of dataset *D1* using homography calibration; given as error in degree of visual angle. Comparison of different resolutions. Top left: 50x25px, top right: 100x50px and bottom: 200x100px.

mean accuracy achieved during each run. We calculate overall mean accuracy as weighted mean over the results of each instance with the number of samples as weight. The results with respect to calibration method and resolution can be seen in table 6.1:

All pairwise differences of prediction accuracy are significant (p<.001). Both calibration methods show a trend towards better results for higher resolutions while calibration using global offset is slightly superior in every case. We think this trend will continue for higher resolutions but are unable to test it due to time constraints.

For completeness we include the numbers for gaze and baseline as well. The figures support the visual impression. Gaze accuracy is stable over all resolutions and should thus have no impact on the results. The baseline accuracy for calibration using homography produces as far as we can tell random results while the baseline for offset remains stable over all resolutions. Our results using global offset approximate the baseline with increasing resolution but following the given trend it would take very high resolutions to get reasonably close.

6.2.2 Calibration Quality

The accuracy of dense pursuit can be affected by a variety of factors. We believe that the quality of used calibration samples might have a large impact and decide to investigate this further.

We propose two measures to determine calibration quality:

- *Spacial mapping*: From our collected data we generate a heat-map where the colour at a pixel location indicates the difference between a calibration sample taken at this point and

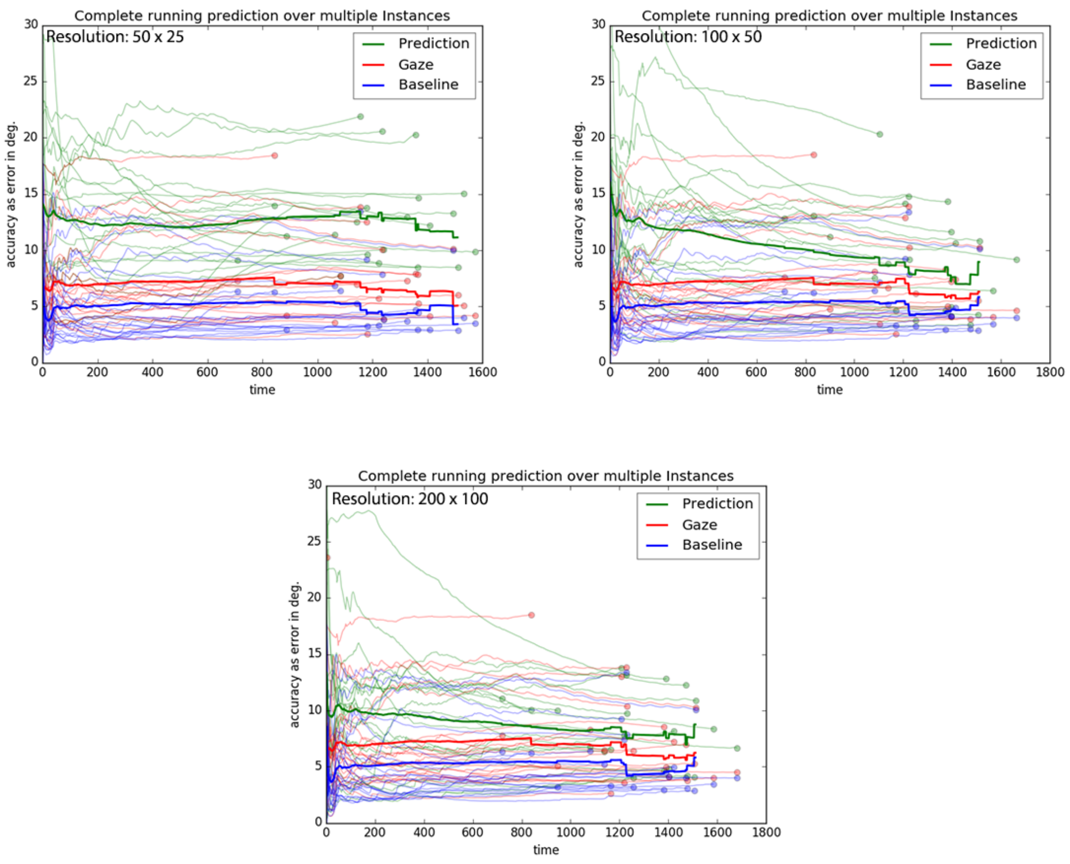

Figure 6.5: Running mean prediction accuracy over all instances of dataset *D1* using global offset calibration; given as error in degree of visual angle. Comparison of different resolutions. Top left: 50x25px, top right: 100x50px and bottom: 200x100px.

the actual location of the stimulus. We hope for insights to what areas of the screen are covered as well as the spacial distribution of object approximation quality.

- *Absolute error*: In addition to a heat-map we propose a mapping of the prediction with respect to the baseline. With this representation we hope to find out if our approximations distribute normally around the desired object and how many of our calibration samples are of good quality. This visualisation focuses on how our prediction differs from the baseline whereas spacial mapping visualises where in the frame different calibration quality was encountered.

We apply this to the previously used resolutions (50x25px, 100x50px and 200x100px). The collection of calibration samples does not depend on the actual approach used for calibration so we do not distinguish by calibration method. The results can be seen in *figure* 6.6.

As we expected the results for the lowest resolution tested are very inaccurate. There are also a lot less calibration samples collected which indicates that objects could not properly be detected. Most calibration points seem to be off along the vertical axes but there are not enough samples to claim an actual effect.

For the resolutions 100x50px and 200x100px we see stronger results. The centre area of the frame is mostly covered by calibration points of high quality. Calibration quality for the top of the frame seems to be generally better. The bottom area shows several outliers. We attribute this to cases where the moving experimenter was detected instead of the moving stimulus as intended.

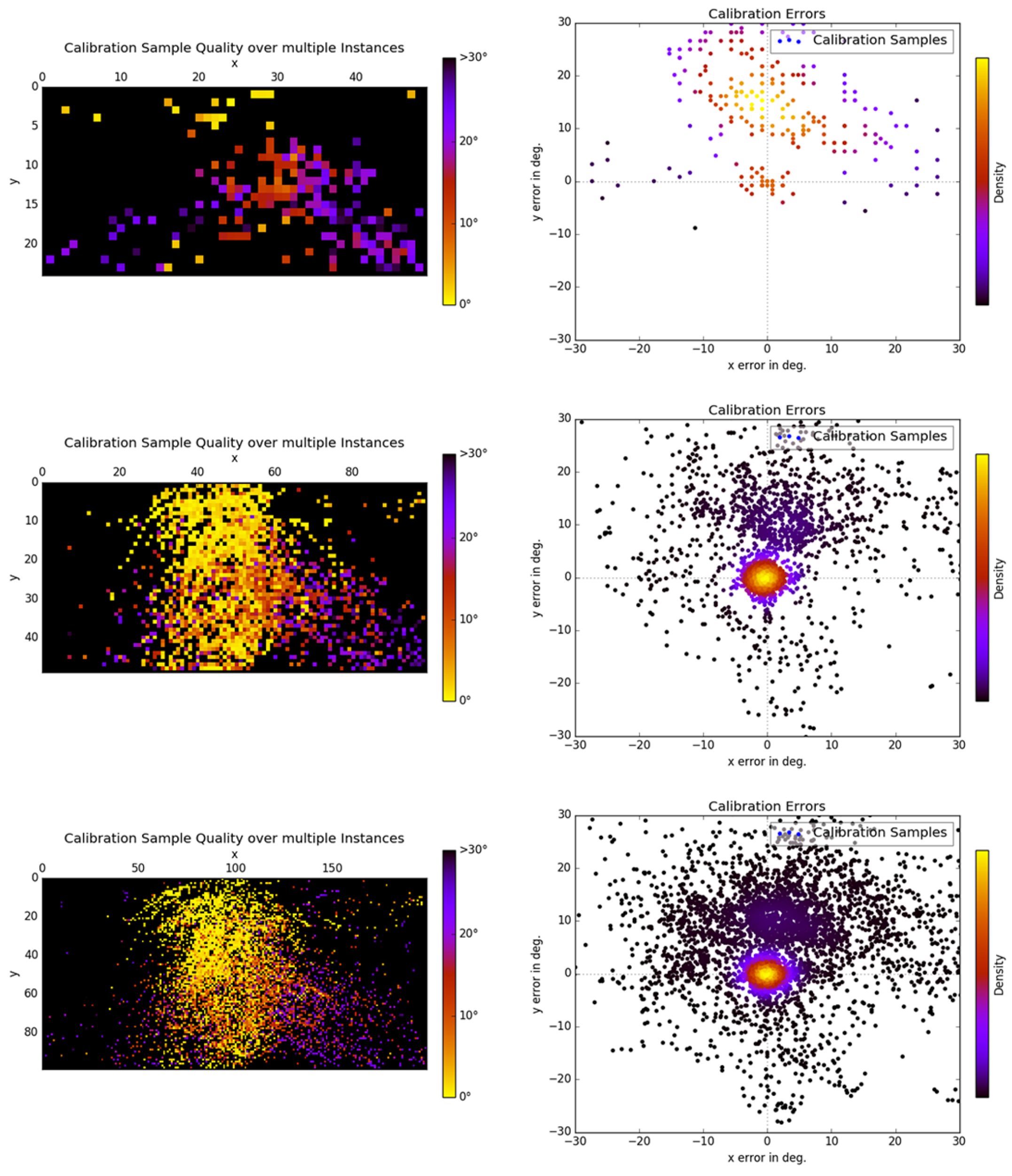

Figure 6.6: Quality of calibration samples over multiple resolutions (top: 50x25px, middle: 100x50px and bottom: 200x100px). Left: Heat-map of sample locations where colour indicates deviation from baseline. Right: Absolute deviation from baseline with colour coded point density.

	50x25	100x50	200x100
homography	13.74°	10.87°	10.45°
global offset	12.73°	9.22°	8.23°
gaze	7.06°	7.00°	7.60°
homography baseline	16.07°	17.0°5	27.11°
global offset baseline	5.32°	5.39°	5.89°

Table 6.1: Mean accuracy of *dense Pursuits* with respect to calibration method and resolution compared with gaze accuracy and the respective baselines.

While this is not generally wrong (the experimenter moved similar to the stimulus) it is not what we wanted and hints for a possible weakness in our approach.

The plot of absolute error supports those findings. We observe a high amount of calibration samples with very little deviation from the baseline. Larger deviations are less common and mostly lie in the top half of the plot that most likely corresponds to the previously mentioned type of error.

6.2.3 Temporal and Spacial Impacts

In *chapter* 6.2.1 we saw an impact of calibration time on the accuracy of dense Pursuits. We investigate this further by directly correlating the number of samples collected to the accuracy achieved.

Figure 6.7 shows this relation. To account for multiple data samples recorded with the same amount of calibration samples we apply binning and compute the mean running prediction as the weighted average over those bins. That means that a value of the mean prediction curve corresponds to the mean over all data points with as many or less calibration samples. The amount of samples in one bin is colour coded and a darker tone corresponds to more samples. We do not include a graphic for 50x25px as it contains only few samples and is of little value here.

The graphs show that we collected a large amount of data points with only few calibration points available. For both resolutions we see a tendency towards higher accuracy with increasing amount of collected calibration samples. Also with increasing resolution an overall larger amount of calibration samples is collected. As the durations of the input do not change this hints at more object detections.

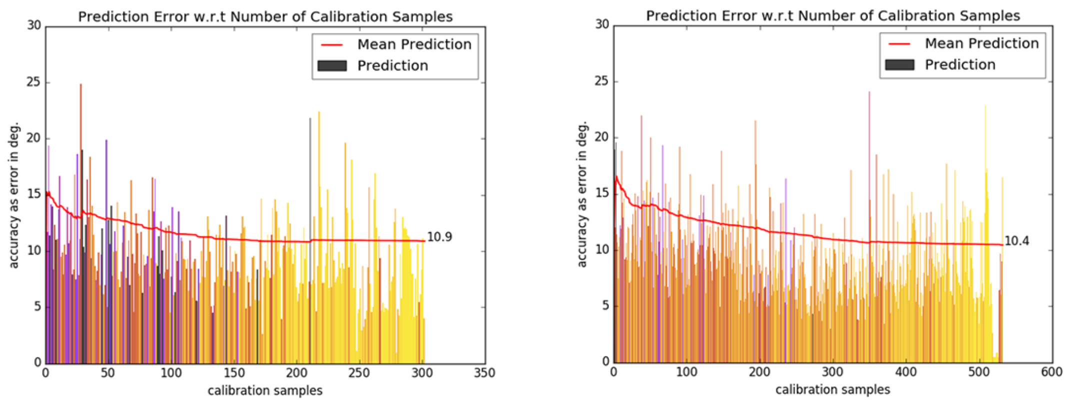

Figure 6.7: Prediction accuracy with respect to number of calibration samples for homography calibration. Left: 100x50px, right: 200x100px.

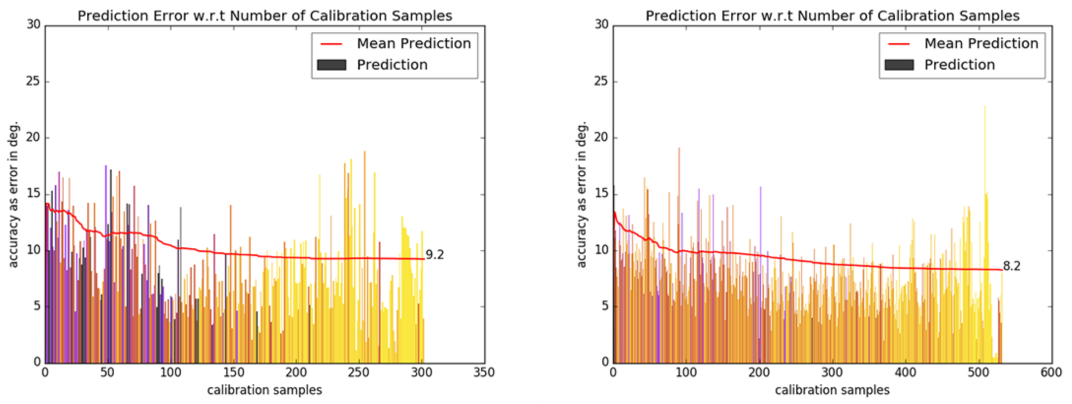

Figure 6.8: Prediction accuracy with respect to number of calibration samples for global offset calibration. Left: 100x50px, right: 200x100px.

The same applies when using global offset for calibration (compare *figure* 6.8), however the accuracy converges to its final state way quicker which means that a small amount of samples seems to be sufficient. Thus, global offset might be well suited for quick calibration whereas homography calibration can produce a theoretically stronger model but needs more calibration samples to achieve good results. Mixing the two approaches might thus lead to a promising synergy.

In addition to the number of calibration samples we also investigate the effect of area covered by calibration samples. We hypothesize that a larger area covered should be correlated to higher accuracy. However this is not actually the case as shown in *figure* 6.9. We do not elaborate on all possible combinations of calibration method and resolution as they all produce comparable results.

We do not know the exact reason for this behaviour but we assume that the quality of calibration points has a larger impact on the accuracy than the area covered. As shown in *chapter* 6.2.2 we observe a centre area with high quality calibration samples and a surrounding area with weaker samples. It might thus be the case that increases in area beyond a certain point are often coupled with adding weak calibration samples and thus weakening the overall calibration accuracy.

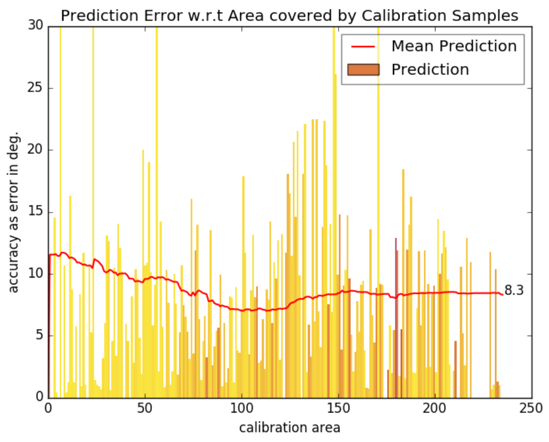

Figure 6.9: Prediction accuracy with respect to area covered (in px^2) by calibration samples.

6.3 Sparse Pursuits Evaluation

Following our research questions we next evaluate the performance of our sparse Pursuits approach (*Q2*). Our goal is to compare the results to dense Pursuits and find weaknesses and strengths of both approaches as well as recommendations when to use which if possible.

6.3.1 Prediction Accuracy

The first property that we investigate is prediction accuracy. As sparse Pursuits is significantly less computationally demanding we run this on higher resolutions though. In order to be able to compare it to dense Pursuits we use a resolution of 200x100px as well as the maximum resolution of 1280x720px. The maximum is given by the resolution of the video files from *D1*.

A comparison of both resolutions using homography calibration can be seen in *figure* 6.10. Process of generation and semantics are identical to the ones explained for *figure* 6.4.

Comparing the results for identical resolutions we see inferior accuracy of sparse Pursuits in the long run. It starts off almost identical to the results from dense Pursuits but does not improve over time. Also in contrast to the results from dense Pursuits we do not see any improvement from using a larger resolution. We see two possible explanations for this: 1) As we only use samples instead of the whole image information, resolution is not the factor scaling accuracy. In this case other factors like the number of hypothesis or the quality of tracking might influence the results instead. 2) In *chapter* 4.2.4 we expressed our doubts towards our implementation of the homography calibration for sparse Pursuits. The results we see could be caused by an error in the implementation.

An analysis of global offset calibration using sparse pursuit could help to answer this question. The results are shown in *figure* 6.11.

Again we do not see significant improvement when using higher resolutions. We conclude that our explanation (1) is the more likely option. When comparing the results to dense Pursuits we achieve comparable accuracy while performing slightly weaker. Taking into account that computation times are only a fraction of their dense Pursuits counterparts though this provides an acceptably good approximation of dense Pursuits results.

The result in numbers is shown in table 6.2.

We leave out figures for gaze and baseline accuracy as we established previously that both are independent of resolution and are not affected by the choice of dense or sparse Pursuits either. All pairwise differences are significant (p<.001) though as mentioned before not meaningful.

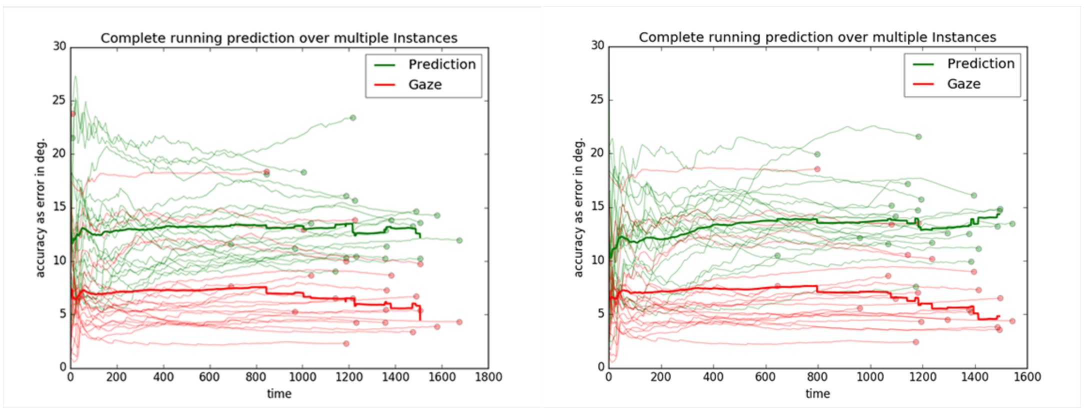

Figure 6.10: Running mean prediction accuracy over all instances of dataset *D1* using sparse Pursuits with homography calibration; given as error in degree of visual angle. Comparison of different resolutions. Left: 200x100px, right: 1280x720px.

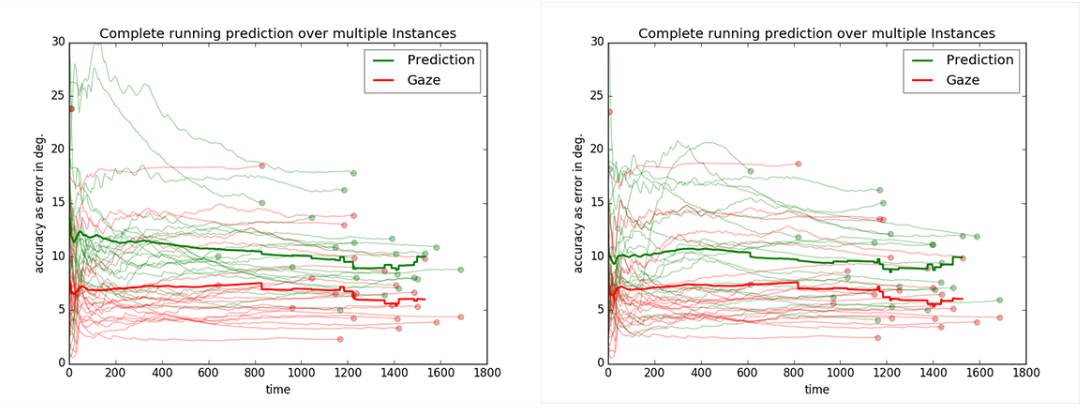

Figure 6.11: Running mean prediction accuracy over all instances of dataset *D1* using sparse Pursuits with global offset calibration; given as error in degree of visual angle. Comparison of different resolutions. Left: 200x100px, right: 1280x720px.

	200x100	1280x720
homography	13.42°	13.77°
global offset	9.97°	9.42°

Table 6.2: Mean accuracy of sparse Pursuits with respect to calibration method and resolution.

For comparison we include frame rates for all tested resolutions of both dense and sparse Pursuits in *table* 6.3. Computation times for dense Pursuits grow with the area of the frame which leads to unusable frame rates rather quickly. Sparse Pursuits performance on the other hand is not as tightly coupled to resolution and overall significantly faster.

6.3.2 Calibration Quality

As the previous chapter showed that resolution has no meaningful impact on the quality of sparse Pursuits results, we only investigate calibration quality for a resolution of 200x100px. The results based on the method introduced in *chapter* 6.2.2 are shown in *figure* 6.12.

While the general pattern in the heat-map is comparable to the results for dense Pursuits we observe a larger area covered but also an overall tendency to a worse prediction accuracy (indicated by darker colour).

In contrast to dense Pursuits false object detections can not be attributed to one single cause but seem to happen over the whole screen space. As before though, we observe a higher frequency of good approximations for the centre area whereas the accuracy declines towards the borders.

The absolute error seems to be distributed normally around perfect detection. This is what we would expect although we would have hoped for more samples in the centre and less density in the surrounding comparable to what we achieved for dense Pursuits.

	50x25	100x50	200x100	1280x720
dense Pursuits	8.5fps	2.3fps	0.4fps	-
sparse Pursuits	-	-	11.0fps	3.9fps

Table 6.3: Frame rates for dense and sparse pursuits dependent on resolution.

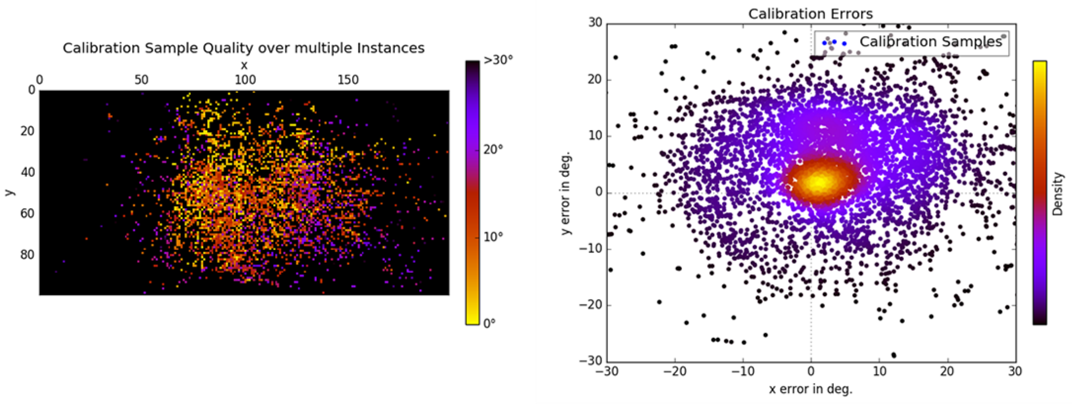

Figure 6.12: Quality of calibration samples using sparse Pursuits. Left: Heat-map of sample locations where colour indicates deviation from baseline. Right: Absolute deviation from baseline with colour coded point density.

6.3.3 Temporal and Spacial Impacts

Although the accuracy is not affected by resolution we find a strong difference in the number of calibration samples generated. Contrary to our expectation the use of a higher resolution resulted in less samples collected (compare *figure* 6.13).

While we - as of now - have no good explanation for this it is another pointer towards a hidden factor that might not be scaled appropriately to account for higher resolution. We suggest to investigate this further in future work.

Apart from that we see a compensating effect for higher resolution. While there are less samples available the accuracy for low sample counts is higher. This leads to an overall comparable accuracy while even being slightly better and converging earlier. For higher resolution we actually see the same behaviour that we observed for dense Pursuits where prediction accuracy would improve over the first few samples collected and then remain stable afterwards.

For the effect of area covered we see the same results - or rather lack of results - as discussed for dense Pursuits. There seems to be no quantifiable impact.

6.4 Real-world Pursuits with Humans

From our previous artificial example we move on to a more realistic setting. We use dataset *D2* where participants were asked to follow a moving experimenter with their eyes, both while stationary and while moving. Overall we distinguish three cases:

- *C1*: The participant remains stationary while the target is moving.
- *C2*: The participant moves while following a moving target with her eyes.
- *C3*: The participant moves and follows stationary objects with her eyes.

For scenarios *C1* and *C2* the target is the moving experimenters head whereas for *C3* it is post-its attached to objects along the path. The dataset consists of 100 samples where cases *C1* and *C2* are covered with 40 files each and *C3* is covered with 20 files. The files differ in length but have an approximate runtime between half a minute and a minute.

Following our previous findings we do not use homography calibration for sparse Pursuits and limit ourselves to one resolution per approach. For dense Pursuits we decide to use a resolution of 100x50px as a trade-off between accuracy and computation time and for sparse Pursuits we choose 640x360px as full resolution is not needed (compare *chapter* 6.3.1).

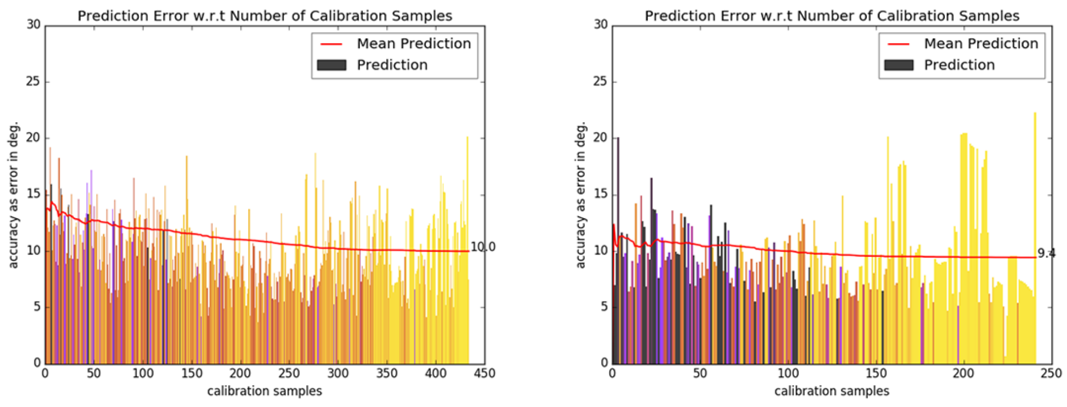

Figure 6.13: Prediction accuracy with respect to calibration samples collected using sparse Pursuits with global offset calibration. Comparison of different resolutions. Left: 200x100px, right: 1280x720px.

6.4.1 C1

For all three test cases we compare the overall accuracy as running mean over time as introduced previously. A comparison of the overall accuracy of all three approaches for test case *C1* is presented in *figure* 6.14.

As mentioned previously the number of data samples vary over different files. Thus the expressiveness of curves towards the end diminishes due to less support. In the case of *figure* 6.14 that means that the predictions of dense and sparse Pursuits using global offset can be seen as comparable. All pairwise differences are significant (p<.001) though.

While the two approaches using global offset perform comparably well the approach using homography for calibration does not generate reasonably good results. We assume that there were not enough samples due to the short videos to generate strong calibrations. The baseline shows that in theory a better calibration than the given gaze input could be achieved but none of the approaches does so over the long run. Better results are achieved occasionally though when an actual object is detected.

Figure 6.15 shows that dense Pursuits produces a significantly lower amount of calibration samples and only few of them are reasonably strong. Sparse Pursuits on the other hand produces many strong calibration points. The pattern indicates that often a point below the baseline was selected instead of the actual target. We attribute this to selections on other body parts of the experimenter as those move similar to the head (which was the actual target).

6.4.2 C2

In contrast to *C1* participants were moving instead of remaining seated while following a moving experimenter with their eyes in *C2*. A comparison of the overall accuracy of all three approaches for test case *C2* is presented in *figure* 6.16.

While dense and sparse Pursuits performed comparable when using global offset in *C1* we see a strong discrepancy for *C2*. Both dense approaches perform significantly inferior to sparse Pursuits which achieves an accuracy close to the given gaze prediction.

Based on the previous (comparably) good results achieved by dense Pursuits we exclude the task of tracking a human as the reason for the weaker performance. We thus see the results for *C2* as an indicator that dense pursuit in its current form is not well suited for handling moving users. Sparse Pursuits on the other hand is only slightly affected.

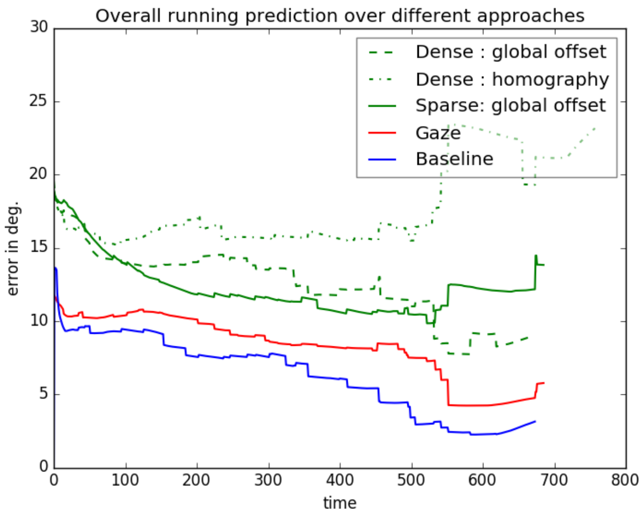

Figure 6.14: Prediction accuracy on test case *C1* using dense Pursuits with both homography and global offset as well as sparse Pursuits using global offset calibration.

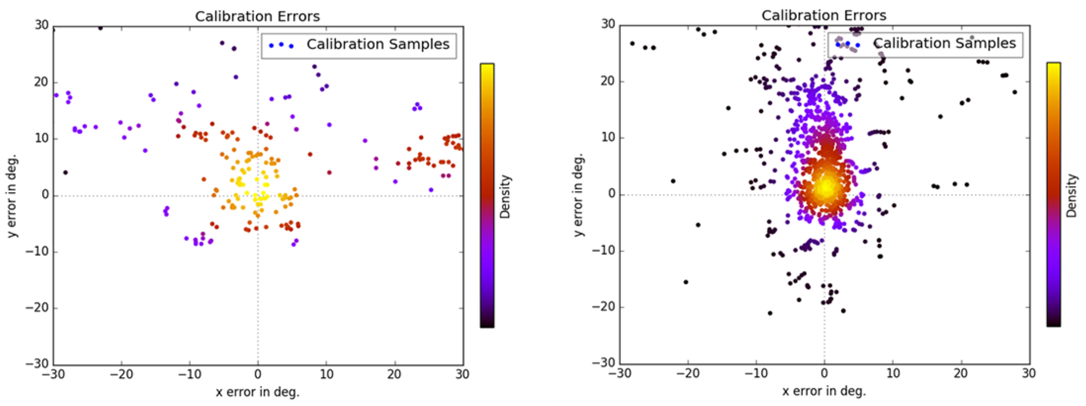

Figure 6.15: Calibration quality for *C1* using dense (left) and sparse Pursuits (right).

The above trends are also reflected in the calibration quality (compare *figure* 6.17). Calibration points created by dense Pursuits are generally not as accurate and we do not see a strong agglomeration around the centre. That means there are almost no accurate object estimations.

6.4.3 C3

The final test case *C3* covers cases where a participant is moving while the stimulus remains stationary. From the participants point of view this results in relative movement that we might use for Pursuits. The overall accuracy of all three approaches for test case *C3* can be seen in *figure* 6.18.

For this test case it turns out that none of our approaches generates good results. The sample size is only half of the sizes for test cases *C1* and *C2* so the results should be treated carefully. Nonetheless the achieved accuracy is very low over all approaches.

One possible reason for this can be found during closer inspection of the raw video and gaze data. This shows that for almost all participants the targets were located close to or below the lower frame border. Targets were mounted at a height of about one meter and would likely not be

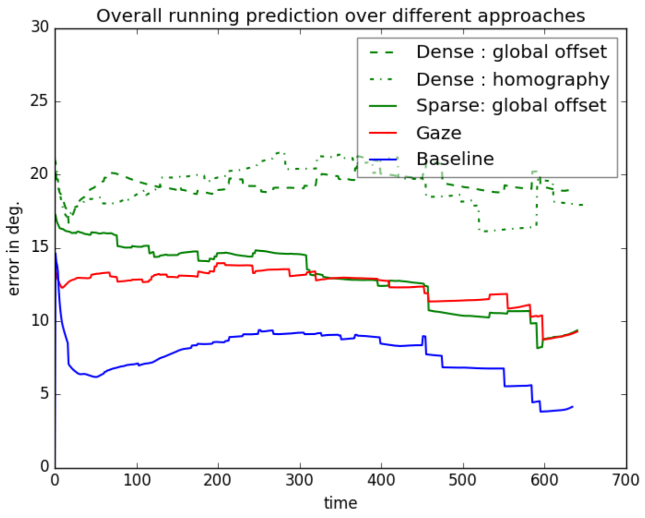

Figure 6.16: Prediction accuracy on test case *C2* using dense Pursuits with both homography and global offset as well as sparse Pursuits using global offset calibration.

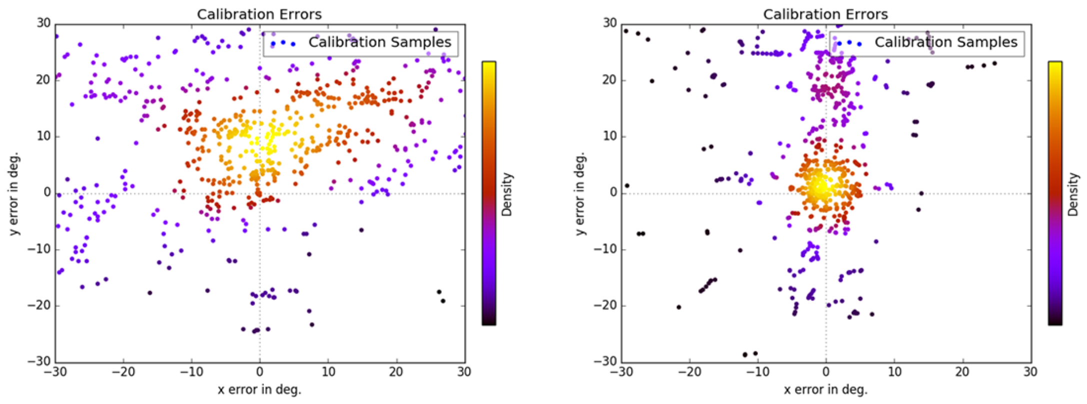

Figure 6.17: Calibration quality for *C2* using dense (left) and sparse Pursuits (right).

captured when users were looking straight ahead instead of deliberately downwards. That leads to targets appearing not at all or only shortly in the frame.

While our approach might indeed perform worse on this last test case we do not feel confident to claim such an effect based on the used data. We suggest further investigation of this question in some future work.

Overall we see a tendency of global offset performing better in all use cases (except *C3*, but we are not willing to trust this result). While we saw high accuracy for dense Pursuits in cases with a stationary participant the approach seems to struggle with cases where the participant is moving. Sparse Pursuits seems to be unaffected by this differentiation and performs equally accurate in both cases. *Table* 6.4 quantifies those results as numbers. The differences between all displayed prediction accuracies are pairwise significant (p<.001).

6.5 Properties affecting Object Detection

As a final step in this thesis we investigate the applicability of our work beyond calibration. For that we search for object properties and environments that affect the performance of our dense

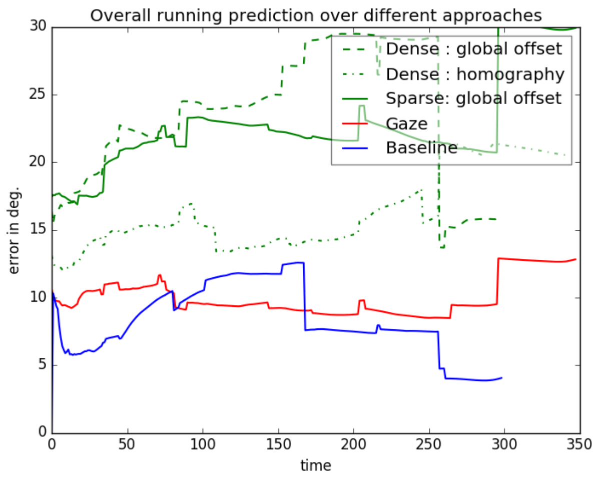

Figure 6.18: Prediction accuracy on test case *C3* using dense Pursuits with both homography and global offset as well as sparse Pursuits using global offset calibration.

	C1	C2	C3
dense homography	15.93°	19.84°	17.35°
dense global offset	13.62°	20.76°	23.90°
sparse global offset	10.36°	13.65°	19.97°
gaze	9.05°	13.30°	9.80°
global offset baseline	6.90°	8.60°	5.89°

Table 6.4: Mean accuracy of dense and sparse Pursuits on the three test cases *C1-3* with gaze accuracy and baseline.

and sparse Pursuits algorithms. We collect qualitative findings from our previous evaluations and using the datasets *D3* and *D4*.

During the analysis of dataset *D1* we see dense Pursuits generating strong results. In this dataset objects were clearly distinguishable from the background which we see as a property positively affecting detection performance of dense Pursuits. For sparse Pursuits this property seems to be not as important, but the approach struggles with small objects as they are likely to be not covered by feature points. Dense Pursuits does not have this limitation as it covers all possibilities due to its dense nature.

Another effect that we see with all our datasets is wrong detections based on similar movements. In the case of dataset *D1* this is oftentimes the experimenters hand while moving the stimulus while in dataset *D2* we see strong correlations to the whole body when participants were focussing on the head of the experimenter (compare *figure* 6.19).

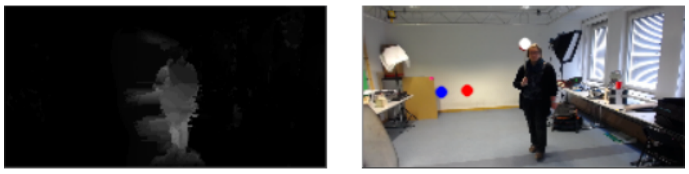

Figure 6.19: Dense Pursuits creates high correlation for the experimenter instead of only the stimulus in cases where both are moving similarly.

This is a situation that other approaches using classical Pursuits deliberately try to avoid. The original paper even reports on differences between trajectories to be able to successfully distinguish them. We do not have control over this variable though. As we saw this leading to wrong object detections multiple times it might be necessary to search for more than one matching object. This would enable more sophisticated decisions whether an object should be selected or if there are multiple equally likely possibilities. We also see the use of gaze calibrated with our approach as an option to decide between multiple possible objects.

Another critical situation occurs when neither gaze nor the world view move. This can happen when users focus on a non-moving object. In this case we get strong correlations for the whole screen. For dense Pursuits we excluded cases with absolutely no movement but subtle movements can still lead to this effect.

As our approach is based on smooth pursuits eye movements it is by definition not built to work for resting gaze. Nonetheless it would be necessary to reliably detect those situations (e.g. by recognizing the mentioned correlation of the whole screen) and exclude generated calibration points.

Faster moving objects seem to generally be better suited for detection for both dense and sparse Pursuits as they generate stronger vector fields that can be easily distinguished from the expected surrounding noise. This effect does only generalize to a certain degree though, as too fast movements can lead to a loss of tracking in the optical flow algorithm.

Dataset *D3* consists of several cases where participants were presented with a variety of different objects. Those were moved in circular trajectories. We noticed that level of detail is an important property for the success of sparse Pursuits. High levels of detail led to a dense crowding of feature points around the detailed objects leaving the rest of the screen less covered. Low levels of detail on the other hand lead to no feature points generated at all. One type of object very susceptible to this was clothing and even more so dark clothing. In these cases feature points are only generated at the object borders which leads to weak calibration samples that are no good representation of the underlying object.

While level of detail has no visible effect on the correlation map generation by dense Pursuits it affects outcomes in a different way. Object borders are poorly detected for low levels of detail and high levels of detail can lead to over-segmentation and thus to undesired results when trying to find the object contours. The same applies to colour differences, e.g. between a foreground object and the background. If the difference is not large enough no border will be detected.

For most cases we see both more calibration samples and better calibration quality towards the centre of the frame. We believe this happens due to humans moving their heads rather than only their eyes to look at objects in their periphery. We do not see this as a limitation of our approach but rather think it might be advised to come up with a model for calibration that is able to better account for different levels of confidence for different areas of the frame.

Dataset *D4* consists of several real world scenarios like driving and moving in- and outdoors. Contrary to our expectation driving seems not to be well suited for calibration with our approach. As the gaze of the driver is most of the time directed straight forwards, the above mentioned effect leads to correlations with the interior of the car. We would have expected correlations with passing-by cars or signs but that did not happen. We only have a limited sample of one participant though, so this might be worth some further investigation as we see high potential in this use case.

Another issue that we noticed for the car scenario was poor tracking of fast moving objects. Especially road surface markings could often not be tracked successfully so that feature points moved together with the car instead of following the ground movement. We attribute this to the high speed of the car that blurs detail that would be needed for tracking.

Changing lighting conditions caused by participants moving indoors and outdoors had no effect on the detections. However indoors we saw several cases of gaze correlating with whole walls, e.g. when the participant looked at some information pinned to the respective wall while passing by. Unfortunately we see no good way to handle this using Pursuits with optical flow. While walls were problematic in this regard, they are ideally suited for object extraction by dense Pursuits. The high contrast of any potential object to the wall behind allows for very precise contour estimation.

Also outdoor movement performed very well for our method. It seems that the high level of detail from the surrounding paired with slow movement speed while walking resembles ideal conditions for feature tracking. Open spaces can impose difficulties though, whenever a user gazes at something on the ground. While the flow field changes continuously over distance (at constant speed, close objects move quicker from the users point of view), those changes are subtle and it is often hard to find the actual location on the ground.

The dataset also features some instances of the participant communicating with other people. Those cases were highly unsuited for our approach as the participants gaze showed strong jumping behaviour during conversations. While this would be expected in a conversation it also appeared in other situations. For future work it might be interesting to explicitly find such situations and avoid calibration. On the other hand sometimes jitter in the gaze prediction is only imposed by poor tracking quality. In those cases some sort of interpolation for missing gaze points might still enable to follow the pursuits eye movements. In any case it would be needed to decide which of the two options is the cause of jumping gaze in order to react appropriately.

6.6 Summary of the Evaluation Results

During our evaluation we came across several interesting findings. We showed that dense Pursuits achieves an accuracy of up to 8.23° of visual angle with a trend towards better results with increasing resolution. Using homography for calibration performs constantly inferior to global offset calibration but improves for higher resolutions as well. While we were not able to achieve this we showed that in theory an accuracy of up to 5.39° of visual angle is possible with our approach.

When using homography we see a constant increase of accuracy with increasing number of calibration samples whereas global offset quickly converges to a constant accuracy level that remains almost unchanged by increasing the number of calibration samples.

In contrast to dense Pursuits our sparse Pursuits approach does not show major effects for increasing resolution. Global offset calibration achieves an accuracy of up to 9.42° of visual angle and thus approximates dense Pursuits results reasonably well while performing significantly faster. Homography calibration seems not to work with sparse Pursuits which might be caused either by a mistake during implementation or lower quality of calibration samples when using this approach.

We compare different relations of users and objects moving to each other and find our approaches performing best when users remain stationary and follow a moving object with their eyes. Both dense and sparse Pursuits can handle this case when using global offset calibration while homography calibration achieves considerably weaker results. When both user and object are moving only sparse Pursuits with global offset calibration achieves results comparable to the previous lab study. We attribute this to a poor calibration set for dense Pursuits that might be caused by a general weakness of the approach for moving users. In the case of moving users looking at still objects none of our approaches could achieve good results. This might again be caused by a general weakness of our approach towards this condition but could very likely also be attributed to faulty data.

Finally our qualitative analysis showed that object speed, level of detail and trajectory similarity are properties that can affect our approach in negative ways. We find that users fixating on an object while not moving is a case that needs to be detected and excluded for future work as it is not suited for Pursuits calibration and caused faulty selections in our analysis.

7 Conclusion and Future Work

The previous chapters addressed the design, implementation and evaluation of an algorithm to enable Pursuits interaction and calibration with real world objects. With this final chapter we recap our previous findings and lessons learnt. Furthermore it should be an inspiration for future work based on this approach.

7.1 Summary

Over the course of this thesis we introduced a technique to select real world objects and use them for calibration of a head mounted eye tracker. Our idea is based on the Pursuits approach [45, 66] to use smooth pursuits eye movements and correlate them with moving objects. We extend this idea beyond the current approaches and use real world objects instead of artificially generated targets for Pursuits.

We propose a pipeline using optical flow to track feature points. We correlate those features with gaze data provided by a head-mounted eye tracker and extract objects based on high correlations.

We provide our approach in two different versions with different goals each: 1) We propose dense Pursuits as a theoretical approach to cover the whole screen space and allow for the extraction of actual object representations based on a generated correlation map. 2) To make our approach real-world applicable we also propose sparse Pursuits which is a close to real-time version of our idea using sparse optical flow for tracking heuristic feature points instead of the whole screen space. We use the extracted objects for calibrating an eye-tracker by either computing a homography or a global offset.

We evaluate our algorithm on lab data to find out how well both dense and sparse Pursuits perform and on more realistic scenarios to find out what properties have an effect on the outcome. We find calibration with global offset to create superior results to homography calibration. Dense Pursuits creates the overall best result with an accuracy of $8.23°$ of visual angle. Sparse Pursuits performs slightly weaker with up to $9.42°$ of visual angle while being significantly faster. The theoretical best results possible with the method we use is an accuracy of $5.32°$ of visual angle when using perfect object representations. In a real-world setting only spars Pursuits can achieve comparable results to our lab study and calibrates a stationary user with moving objects at an accuracy of about $10.36°$ and a moving user on the same target with $13.65°$ of visual angle. Our approach is dependent on level of detail and colour contrast in the input and struggles with users looking at stationary objects while being stationary themselves.

7.2 Conclusion

While our approach can not (neither in theory nor in practical application) achieve accuracy comparable to other state-of-the-art methods (Pursuit Calibration achieved better than $1°$ of visual angle [45]) we are able to generate reasonable calibration without the use of artificial stimuli. Based on the results of our evaluation of dense Pursuits we believe that with higher resolution we can achieve considerably better accuracy, although we were not able to test this due to time constraints.

We also believe that sparse Pursuits has high potential for real-world applications as it is not dependent on resolution to achieve comparably strong results. The approach can (with slight modifications) already run in real-time and we believe that it could be a viable option for many applications if prediction accuracy can be further increased.

From our qualitative analysis we found one mayor challenge that we were not able to solve with our current approach. When a participant would focus on a non-moving object while also not moving herself this caused strong correlation values with all non-moving content in the image.

Those are situations that lead to many of the wrong object proposals that occurred during our analysis and thus need to be reliably detected and addressed in future work.

Our qualitative analysis showed that we are able to correctly find objects that a user is looking at in a variety of different contexts. We think going beyond calibration and using those objects for interaction should be the next step. For this it might be necessary to apply object recognition in order to allow for different commands to be triggered on following specific objects.

7.3 Future Work

As just mentioned we see a lot of potential for improvement of our approach. The trajectory map generation in our dense Pursuits algorithm could be vectorized similarly to the correlation computation to achieve an easy boost in speed. As most of the operations involve matrix operations we also believe that it should be possible to execute them on a GPU to possibly get it running in real-time for smaller resolutions. The algorithms for optical flow and image segmentation that we use are not always among the best currently used, so there are certainly a lot of options for improvements that could be explored.

In addition to increasing performance, accuracy could be improved. For dense Pursuits this might go hand in hand. Other options could include determining calibration over several frames instead of frame based or applying salience models to the extracted objects to find likely gaze points. It might also be promising to change our computation order and segment the image into objects first and track and correlate them afterwards.

In *chapter* 3.3 we propose a range of different possible applications for our approach. We believe and hope that our work on calibration is only a starting point.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994.

[3] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[6] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[7] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[8] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[9] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *Computer Vision-ECCV 2004*, pages 25–36, 2004.

[10] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[11] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[12] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3241–3248. IEEE, 2010.

[13] M. Carter, E. Velloso, J. Downs, A. Sellen, K. O'Hara, and F. Vetere. Pathsync: Multi-user gestural interaction with touchless rhythmic path mimicry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3415–3427. ACM, 2016.

[14] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001.

[15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[16] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[17] C. Clarke, A. Bellino, E. Velloso, H. Gellersen, et al. Tracematch: A computer vision technique for user input by tracing of animated controls. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 298–303. ACM, 2016.

[18] C. Clarke and H. Gellersen. Matchpoint: Spontaneous spatial coupling of body movement for touchless pointing. In *Proceedings of the 30th annual ACM symposium on User interface software and technology*, pages 179–192. ACM, 2017.

[19] L. D. Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[20] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[22] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570, 2015.

[23] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[24] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *Computer Vision–ECCV 2000*, pages 751–767, 2000.

[25] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. *Image analysis*, pages 363–370, 2003.

[26] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[27] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[28] K. Haris, S. N. Efstratiadis, N. Maglaveras, and A. K. Katsaggelos. Hybrid image segmentation using watersheds and fast region merging. *IEEE Transactions on image processing*, 7(12):1684–1699, 1998.

[29] E. Huey. The psychology and pedagogy of reading. 1908.

[30] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[31] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[32] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[33] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[34] M. Kassner, W. Patera, and A. Bulling. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*, pages 1151–1160. ACM, 2014.

[35] M. Khamis, O. Saltuk, A. Hang, K. Stolz, A. Bulling, and F. Alt. Textpursuits: Using text for pursuits-based interaction and calibration on public displays. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 274–285. ACM, 2016.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[37] F. Li, J. Carreira, and C. Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1712–1719. IEEE, 2010.

[38] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–156, 1998.

[39] C. Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, Massachusetts Institute of Technology, 2009.

[40] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[41] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[42] P. Majaranta and A. Bulling. Eye tracking and eye-based human–computer interaction. In *Advances in Physiological Computing*, pages 39–65. Springer, 2014.

[43] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.

[44] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017.

[45] K. Pfeuffer, M. Vidal, J. Turner, A. Bulling, and H. Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 261–270. ACM, 2013.

[46] M. Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[47] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972.

[48] D. A. Robinson. The mechanics of human smooth pursuit eye movement. *The Journal of Physiology*, 180(3):569–591, 1965.

[49] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[50] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[51] K. Sachmann. Pursuits-based Gaze Interaction using Real World Elements as Stimuli. Master thesis, LMU München, 2017.

[52] T. Santini, W. Fuhl, and E. Kasneci. Calibme: Fast and unsupervised eye tracker calibration for gaze-based pervasive human-computer interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2594–2605. ACM, 2017.

[53] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3889–3898, 2016.

[54] J. S. Shell, R. Vertegaal, and A. W. Skaburskis. Eyepliances: attention-seeking devices that respond to visual attention. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 770–771. ACM, 2003.

[55] J. Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[56] B. A. Smith, Q. Yin, S. K. Feiner, and S. K. Nayar. Gaze locking: Passive eye contact detection for human-object interaction. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 271–280. ACM, 2013.

[57] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 246–252. IEEE, 1999.

[58] J. Steil and A. Bulling. Discovery of everyday human activities from long-term visual behaviour using topic models. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 75–85. ACM, 2015.

[59] Y. Sugano, X. Zhang, and A. Bulling. Aggregaze: Collective estimation of audience attention on public displays. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 821–831. ACM, 2016.

[60] J. Sun, S. B. Kang, Z.-B. Xu, X. Tang, and H.-Y. Shum. Flash cut: Foreground extraction with flash and no-flash image pairs. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[61] S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[62] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. In *VISAPP*, pages 125–130, 2011.

[63] R. Valenti and T. Gevers. Accurate eye center location and tracking using isophote curvature. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[64] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*, pages 13–26. Springer, 2012.

[65] E. Velloso, M. Wirth, C. Weichel, A. Esteves, and H. Gellersen. Ambigaze: Direct control of ambient devices by gaze. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, pages 812–817. ACM, 2016.

[66] M. Vidal, A. Bulling, and H. Gellersen. Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 439–448. ACM, 2013.

[67] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):583–598, 1991.

[68] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[69] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[70] A. Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE, 1984.

[71] L. Xia, C.-C. Chen, and J. K. Aggarwal. Human detection using depth information by kinect. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 15–22. IEEE, 2011.

[72] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013.

[73] C. Yang, L. Zhang, R. X. Lu, Huchuan, and M.-H. Yang. Saliency detection via graph-based manifold ranking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3166–3173. IEEE, 2013.

[74] C. Zhang, J. C. Platt, and P. A. Viola. Multiple instance boosting for object detection. In *Advances in neural information processing systems*, pages 1417–1424, 2006.

[75] Y. Zhang, A. Bulling, and H. Gellersen. Sideways: A gaze interface for spontaneous interaction with situated displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 851–860. ACM, 2013.

[76] Y. Zhang, A. Bulling, and H. Gellersen. Pupil-canthi-ratio: a calibration-free method for tracking horizontal gaze direction. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, pages 129–132. ACM, 2014.

[77] Y. Zhang, M. K. Chong, J. Müller, A. Bulling, and H. Gellersen. Eye tracking for public displays in the wild. *Personal and Ubiquitous Computing*, 19(5-6):967–981, 2015.

[78] S. C. Zhu and A. Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18(9):884–900, 1996.

All web references named in the course of this thesis were last visited October 27, 2017.

Contents of the attached CD

The content of the attached CD is organized n the following sub-folders:

- *Thesis*: Contains the original document written in LaTeX as well as a version in PDF format
- *Related Work*: Contains all related work cited in this thesis
- *Source Code*: This folder includes the source code for this project. The most important files are "dense_pursuits.py" for dense Pursuits and "sparse_pursuits.py" for sparse Pursuits.
- *Raw Data*: This folder includes the raw data generated in the process of our evaluation.

We do not include the video files used, as they are both too large and also not our property. To get access we propose to ask the authors of the respective material personally.