Discriminative Correlation Filters in Robot Vision

Andreas Robinson
POPULÄRVETENSKAPLIG SAMMANFATTNING

På mindre än tio år har djupa neurala nätverk utvecklats till heltäckande verktyg inom flera vetenskapliga och tekniska områden på grund av deras nästan orimliga effektivitet när det gäller att modellera komplexa verkliga förhållanden. I synnerhet inom datorseende har de tagit uppgifter som objektikenkning, som tidigare ansågs vara mycket svåra, och förvandlat dem till praktiska vardagliga verktyg. Neurala nätverk måste dock tränas med superdatorer på massiva datamängder i timmar eller dagar, och detta begränsar deras förmåga att anpassa sig till förändrade förhållanden.

Denna avhandling undersöker diskriminerande korrelationsfilter, ursprungligen avsedda för spårning av stora objekt i video, så kallad visual object tracking. Till skillnad från neurala nätverk är dessa filter små och kan snabbt anpassas till förändringar, med lite data och minimal datorkraft. Samtidigt kan de dra nytta av den infrastruktur som utvecklats för neurala nätverk och arbeta inom den.

De viktigaste bidragen i denna avhandling visar mångsidigheten och anpassningsförmågan hos korrelationsfilter för olika problem, samtidigt som de kompletterar kapaciteten hos djupa neurala nätverk. I det första problemet visas det att när de appliceras på att spåra små regioner och punkter, överträffar de den ofta använda Lucas-Kanade-metoden, både när det gäller robusthet och precision.

I det andra problemet appliceras korrelationsfiltren på en helt ny uppgift. Här används de för att skilja mellan olika platsen i en 16 x 16 kvadratkilometer stor havsregion nära land, givet endast en horisontprofil - kustlinjens silhuett av öar och holmar sett från ett fartyg.

I det tredje problemet visas hur korrelationsfiltren kan användas för segmentering av objekt i video. Detta är uppgiften att klassifiera enskilda pixlar som tillhörande antingen ett målobjekt eller bakgrunden, givet en segmenteringsmask försedd med den första bildrutan som enda vägledning. Det visas också att diskriminerande korrelationsfilter och djupa neurala nätverk kompletterar varandra; där det neurala nätverket behandlar videon på ett innehålls-agnostiskt sätt, anpassar filtren sig till specifika målobjekt. Den sammansatta funktionen är en realtidsmetod för segmentering.

Slutligen utvidgas segmenteringsmetoden bortom binär mål- / bakgrundsklassificering till att dessutom beakta distraherande objekt. Detta adresserar den grundläggande svårigheten att hantera objekt som liknar varandra.
ABSTRACT

In less than ten years, deep neural networks have evolved into all-encompassing tools in multiple areas of science and engineering, due to their almost unreasonable effectiveness in modeling complex real-world relationships. In computer vision in particular, they have taken tasks such as object recognition, that were previously considered very difficult, and transformed them into everyday practical tools. However, neural networks have to be trained with supercomputers on massive datasets for hours or days, and this limits their ability adjust to changing conditions.

This thesis explores discriminative correlation filters, originally intended for tracking large objects in video, so-called visual object tracking. Unlike neural networks, these filters are small and can be quickly adapted to changes, with minimal data and computing power. At the same time, they can take advantage of the computing infrastructure developed for neural networks and operate within them.

The main contributions in this thesis demonstrate the versatility and adaptability of correlation filters for various problems, while complementing the capabilities of deep neural networks. In the first problem, it is shown that when adopted to track small regions and points, they outperform the widely used Lucas-Kanade method, both in terms of robustness and precision.

In the second problem, the correlation filters take on a completely new task. Here, they are used to tell different places apart, in a 16 by 16 square kilometer region of ocean near land. Given only a horizon profile - the coast line silhouette of islands and islets as seen from an ocean vessel - it is demonstrated that discriminative correlation filters can effectively distinguish between locations.

In the third problem, it is shown how correlation filters can be applied to video object segmentation. This is the task of classifying individual pixels as belonging either to a target or the background, given a segmentation mask provided with the first video frame as the only guidance. It is also shown that discriminative correlation filters and deep neural networks complement each other; where the neural network processes the input video in a content-agnostic way, the filters adapt to specific target objects. The joint function is a real-time video object segmentation method.

Finally, the segmentation method is extended beyond binary target/background classification to additionally consider distracting objects. This addresses the fundamental difficulty of coping with objects of similar appearance.
Acknowledgments

This was not the plan. But here I am, writing the last paragraphs of my dissertation. These years have been a journey, not just for the places I had a chance to visit but for both the challenges, rewards and personal growth.

First and foremost, I am grateful to my supervisor Michael Felsberg for the trust he placed in me, and for being so patient and positive throughout this process, especially when things did not go according to plan. I would also like to thank my co-supervisors, Per-Erik Forssén for many interesting discussions and Fahad Khan for introducing me to the art of publishing scientific papers.

Science is not a solitary pursuit. I am very grateful for all the collaborations within the lab, that actually made this thing possible in the first place: To Mikael Persson, both for our working together in various endeavors, both at CVL and outside, and helping out during stressful times. To Martin Danelljan with whom I shared an office, both for the collaborations on two papers, the discussions we had, and him taking the time to explain math much better than any textbook. To Felix Järemo Lawin, for all the discussions and the sometimes stressful but ultimately fun work together. To Bertil Grelsson, for our work on a rather unusual and interesting project. To Abdelrahman Eldesokey, for diving into a new topic and joining me in that mad dash to get the last paper finished.

I would like to thank everyone at CVL, past and present, both labmates and seniors, for the great and friendly work environment, interesting discussions and board games. Also many thanks to Mikael, Felix, Abdo, Gustav Häger and Emil Brissman, for proofreading and providing feedback on the manuscript.

Finally, thank you to my family; to my brother for saying I could do this, and to mom and dad for their endless support, and just being there.

Andreas Robinson
Linköping, May 2021
Contents

Abstract iii
Acknowledgments vi
Contents vii

I Background and overview 1

1 Introduction 3
  1.1 Outline ........................................... 4
  1.2 Included publications ........................................... 5
  1.3 Other publications ........................................... 9

2 Triangulation 11
  2.1 Introduction ........................................... 11
  2.2 Easy two-view triangulation ........................................... 11
  2.3 Optimal two-view triangulation ........................................... 12
  2.4 Optimal three-view triangulation ........................................... 12
  2.5 Fast three-view triangulation ........................................... 13

3 Tracking 15
  3.1 Introduction ........................................... 15
  3.2 The tracking process ........................................... 15
  3.3 Discriminative correlation filters ........................................... 16
  3.4 Point tracking ........................................... 18

4 Localization 21
  4.1 Introduction ........................................... 21
  4.2 Horizon regression with classical methods ........................................... 22
  4.3 Horizon regression by semantic segmentation ........................................... 24
4.4 Location estimation ........................................... 27
5 Video object segmentation ........................................ 31
5.1 Introduction .................................................. 31
5.2 VOS approaches .............................................. 31
5.3 Fast and robust target models ................................. 33
5.4 Performance evaluation ...................................... 36
6 Segmentation with distractors .................................... 39
6.1 Introduction .................................................. 39
6.2 Incorporating distractors ...................................... 41
6.3 Performance evaluation ...................................... 43
7 Concluding remarks ............................................... 45
7.1 Results ....................................................... 45
7.2 Future work ................................................ 46
Bibliography ....................................................... 47

II Publications ..................................................... 53
Paper A Robust Three-view Triangulation Done Fast ........... 55
Paper B Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking .......................... 65
Paper C GPS-level accurate camera localization with HorizonNet .................................................. 85
Paper D Learning Fast and Robust Target Models for Video Object Segmentation ........................................ 109
Paper E Distractor-aware video object segmentation ............ 123
Part I

Background and overview
Introduction

Within the broader field of computer vision, much theoretical insight can be gained from computationally expensive methods with excellent performance. However in the sub-field of robot vision, it is crucial that methods can adapt to fast changes in dynamic environments.

The goal of this work was to develop efficient estimation- and learning-based algorithms for robotic vision, striking a balance between the quality of results and computation speed. The main research is focused on discriminative correlation filters (DCFs). These have proven very successful in object tracking, but are here applied as adaptive classifiers in novel problems.

Compared to neural networks, DCFs are small and can be quickly adapted to changes, with minimal data and computing power. At the same time, they can take advantage of the computing infrastructure developed for neural networks and operate within them. The main contributions in this thesis demonstrate their versatility and adaptability in various problems, while complementing the capabilities of deep networks.

In the first problem, it is shown that when adopted to track small regions and points, they outperform the widely used Lucas-Kanade method, both in terms of robustness and precision.

In the second problem, the correlation filters take on a completely new task. Here, they are used to tell places apart, in a 16 by 16 square kilometer region of ocean near land. Given only a horizon profile - the coast line silhouette of islands and islets as seen from an ocean vessel - it is demonstrated that discriminative correlation filters can effectively distinguish between locations.

In the third problem, it is shown how correlation filters can be applied to video object segmentation. This is the task of classifying individual pixels as belonging either to a target or the background, given a segmentation mask provided with the first video frame as the only guidance. It is also shown that discriminative correlation filters and deep neural networks complement
1. Introduction

each other; where the neural network processes the input video in a content-agnostic way, the filters adapt to specific target objects. The joint function is a real-time video object segmentation method.

Finally, the segmentation method is extended beyond binary target-and-background classification to additionally consider distracting objects. This addresses the fundamental difficulty of coping with objects of similar appearance.

A major paradigm shift occurred in computer vision during the time of the work presented here was carried out. This is reflected in this thesis by a transition from classical methods and optimization to neural networks and learning. Specifically, the first two papers of this thesis show the author’s initial work on geometry and 3D reconstruction. From paper B (chapter 3) and onward, the work pivots to focus on classification with DCFs, and progressively moves towards approaches involving deep neural networks and learning.

1.1 Outline

Part I of this thesis presents both additional background and overviews of each of the published works, organized into separate chapters for the individual applications. The intent is to outline the core ideas of each paper and provide additional context that is generally left out of the original publication.

Chapter 2 covers traditional concepts of geometric vision, providing background to optimal two- and three-view triangulation, and introduces the fast three-view triangulation approach of paper A.

Chapter 3 describes discriminative correlation filters for visual object tracking, and outlines how they can be applied to point tracking as elaborated on in paper B. This chapter marks the beginning of the transition from classical models to data-driven approaches.

Related to paper C, chapter 4 outlines a localization method whereby an observed horizon is matched to a map location with the use of discriminative correlation filters as feature matching operators. The chapter also details the work to transform a classical vision approach to horizon detection and segmentation, into an equivalent solution based on neural networks.

Chapter 5 covers paper D, where discriminative correlation filters for visual object tracking are adapted into a fast method for video object segmentation.

Chapter 6 introduces the ideas behind paper E, where correlation filters for video object segmentation are generalized to simultaneously recognize distractors in addition to the intended target.
1.2 Included publications

Part II consists of the published editions of the five papers. The abstracts and details of the author’s contributions to each one are provided below.

**Paper A: Robust Three-view Triangulation Done Fast**

Johan Hedborg, Andreas Robinson, and Michael Felsberg. “Robust three-view triangulation done fast.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014, pp. 152–157

This paper won the best paper award in the fourth IEEE international Workshop on Mobile Vision held in conjunction with CVPR.

**Abstract**

Estimating the position of a 3-dimensional world point given its 2-dimensional projections in a set of images is a key component in numerous computer vision systems. There are several methods dealing with this problem, ranging from sub-optimal, linear least square triangulation in two views, to finding the world point that minimizes the $L_2$-reprojection error in three views. This leads to the statistically optimal estimate under the assumption of Gaussian noise. In this paper we present a solution to the optimal triangulation in three views.

The standard approach for solving the three-view triangulation problem is to find a closed-form solution. In contrast to this, we propose a new method based on an iterative scheme. The method is rigorously tested on both synthetic and real image data with corresponding ground truth, on a mid-range desktop PC and a Raspberry Pi, a low-end mobile platform.

We are able to improve the precision achieved by the closed-form solvers and reach a speed-up of two orders of magnitude compared to the current state-of-the-art solver. In numbers, this amounts to around 300K triangulations per second on the PC and 30K triangulations per second on Raspberry Pi.

**Contributions**

This work develops a method for three-view triangulation by optimization, that avoids considering the problem structure. It is found to be much faster and more stable than methods formulating and solving systems of equations. The simplicity makes it very fast on hardware for mobile applications.

The author contributed the software implementation, experiment design and execution and the majority of the writing.
1. Introduction

**Paper B: Beyond Correlation Filters: Learning Continuous Convolution Operators for Visual Tracking**

Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. “Beyond correlation filters: Learning continuous convolution operators for visual tracking.” In: *European conference on computer vision*. Springer. 2016, pp. 472–488

**Abstract**

Discriminative Correlation Filters (DCF) have demonstrated excellent performance for visual object tracking. The key to their success is the ability to efficiently exploit available negative data by including all shifted versions of a training sample. However, the underlying DCF formulation is restricted to single-resolution feature maps, significantly limiting its potential. In this paper, we go beyond the conventional DCF framework and introduce a novel formulation for training continuous convolution filters. We employ an implicit interpolation model to pose the learning problem in the continuous spatial domain. Our proposed formulation enables efficient integration of multi-resolution deep feature maps, leading to superior results on three object tracking benchmarks: OTB-2015 (+5.1% in mean OP), Temple-Color (+4.6% in mean OP), and VOT2015 (20% relative reduction in failure rate). Additionally, our approach is capable of sub-pixel localization, crucial for the task of accurate feature point tracking. We also demonstrate the effectiveness of our learning formulation in extensive feature point tracking experiments.

**Contributions**

In this work it is shown that discriminative correlation filters designed for visual object tracking outperform Lucas-Kanade point tracking in terms of robustness and precision. The author contributed the implementation, experiment design and analysis, as well as the writing related to the point tracking.

**Paper C: GPS-level accurate camera localization with HorizonNet**

Bertil Grelsson, Andreas Robinson, Michael Felsberg, and Fahad Shahbaz Khan. “GPS-level accurate camera localization with HorizonNet.” In: *Journal of Field Robotics* 37.6 (2020), pp. 951–971

**Abstract**

This paper investigates the problem of position estimation of unmanned surface vessels (USVs) operating in coastal areas or in the archipelago. We propose a position estimation method where the horizon line is extracted in a
1.2. Included publications

360° panoramic image around the USV. We design a convolutional neural network (CNN) architecture to determine an approximate horizon line in the image and implicitly determine the camera orientation (the pitch and roll angles). The panoramic image is warped to compensate for the camera orientation and to generate an image from an approximately level camera. A second CNN architecture is designed to extract the pixelwise horizon line in the warped image. The extracted horizon line is correlated with digital elevation model data in the Fourier domain using a minimum output sum of squared error correlation filter. Finally, we determine the location of the maximum correlation score over the search area to estimate the position of the USV. Comprehensive experiments are performed in field trials conducted over 3 days in the archipelago. Our approach provides excellent results by achieving robust position estimates with global positioning system (GPS)-level accuracy in previously unvisited test areas.

Contributions

The paper included in the thesis is the journal extension of [17]. It demonstrates the applicability of DCFs to navigation, exploiting their periodicity to match horizons. The author contributed significantly to the idea development, the software implementation, experiment design and analysis. Both main authors contributed equally to the project.

Paper D: Learning Fast and Robust Target Models for Video Object Segmentation

Andreas Robinson, Felix Järemo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. “Learning Fast and Robust Target Models for Video Object Segmentation.” In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 7406–7415

Abstract

Video object segmentation (VOS) is a highly challenging problem since the initial mask, defining the target object, is only given at test-time. The main difficulty is to effectively handle appearance changes and similar background objects, while maintaining accurate segmentation. Most previous approaches fine-tune segmentation networks on the first frame, resulting in impractical frame-rates and risk of overfitting. More recent methods integrate generative target appearance models, but either achieve limited robustness or require large amounts of training data.

We propose a novel VOS architecture consisting of two network components. The target appearance model consists of a light-weight module, which is learned during the inference stage using fast optimization techniques to
1. Introduction

predict a coarse but robust target segmentation. The segmentation model is exclusively trained offline, designed to process the coarse scores into high quality segmentation masks. Our method is fast, easily trainable and remains highly effective in cases of limited training data. We perform extensive experiments on the challenging YouTube-VOS and DAVIS datasets. Our network achieves favorable performance, while operating at higher frame-rates compared to state-of-the-art.

Contributions

This paper generalizes visual object tracking with DCFs as used in for example paper B, to the video object segmentation task and at video-rate speeds. The author initiated the project and shared equally the idea development, implementation, experiment design and execution and writing, with the co-author.

Paper E: Distractor-aware video object segmentation

Andreas Robinson, Abdelrahman Eldesokey, and Michael Felsberg. “Distractor-aware video object segmentation.” In: Submitted. 2021

This paper is submitted for review.

Abstract

Semi-supervised video object segmentation is a challenging task that aims to segment a target throughout a video sequence given an initial mask at the first frame. Discriminative approaches have demonstrated competitive performance on this task at a sensible complexity. These approaches typically formulate the problem as a one-versus-one classification between the target and the background. However, in reality, a video sequence usually encompasses a target, background, and possibly other distracting objects. Those objects increase the risk of introducing false positives, especially if they share visual similarities with the target. Therefore, it is more effective to separate distractors from the background, and handle them independently.

We propose a one-versus-many scheme to address this situation by separating distractors into their own class. This separation allows imposing special attention to challenging regions that are most likely to degrade the performance. We demonstrate the prominence of this formulation by modifying the learning-what-to-learn [2] method to be distractor-aware. Our proposed approach sets a new state-of-the-art on the DAVIS val dataset, and improves over the baseline on the DAVIS test-dev benchmark by 4.8 percentage points.
Contributions

This extends paper D (and paper [2]) to improve the robustness of the correlation-filter model, by additionally modeling distracting objects. The author initiated the project, contributed to the idea development, implementation, experiment design and execution and writing. Both main authors contributed equally to this project.

1.3 Other publications

Related to papers D and E, the author additionally contributed to
Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. “Learning What to Learn for Video Object Segmentation.” In: Computer Vision - ECCV 2020. 2020

Additional geometry-related work on joint calibration of multiple cameras with minimal shared field of view, was published in
Andreas Robinson, Mikael Persson, and Michael Felsberg. “Robust accurate extrinsic calibration of static non-overlapping cameras.” In: International Conference on Computer Analysis of Images and Patterns. Springer. 2017, pp. 342–353

The author contributed a baseline application where a neural network, AlexNet with new output layers, performed steering and throttle control for autonomous road following, in
Kristoffer Öfjäll, Michael Felsberg, and Andreas Robinson. “Visual autonomous road following by symbiotic online learning.” In: 2016 IEEE Intelligent Vehicles Symposium (IV). IEEE. 2016, pp. 136–143
2

Triangulation

2.1 Introduction

Triangulation is a fundamental computer vision task, with many applications such as 3D-reconstruction, mapping and localization. It takes its name from the triangle formed by the point to be triangulated and two locations from which the point is observed. Multiple methods exist, each designed with different goals in mind, such as ease of computation or correctness. This chapter will overview some of them, starting with a somewhat inaccurate but intuitive method, continuing with precise but slow approaches with closed-form solutions, and ending with that of paper A which trades algebraic correctness for speed.

2.2 Easy two-view triangulation

The ideal view of triangulation might be constructed as in figure 2.1a. Here, there are two pinhole cameras observing the point we wish to triangulate.
2. Triangulation

Rays from their centers are cast through the observations in the image plane into 3D-space, where they intersect on the point. However, the actual view of triangulation is probably closer to figure 2.1b. Here there are multiple cameras looking at the point, and their observations are noisy so rays do not necessarily intersect.

A simple and intuitive approach to two-view triangulation that does account for non-intersecting rays, is the **midpoint method**. Outlined in [19], the idea is to cast two rays from the centers of each camera \( P_1 \) and \( P_2 \) through the respective 2D observations \( x_1 \) and \( x_2 \). The midpoint between the two rays at their closest distance to one another is assigned to the triangulated 3D point \( X \). This is easily solved as a least-squares problem with two unknowns in three equations.

2.3 Optimal two-view triangulation

Although straightforward, the midpoint method does not necessarily minimize the reprojection error, \( d(P_1X, x_1)^2 + d(P_2X, x_2)^2 \). Here, \( d(\cdot, \cdot) \) is the Euclidean distance in the image plane between the projection of \( P_iX \) onto it, and the observation \( x_i \) in it. In other words, it expresses the difference between what the cameras observed, and projections of \( X \) back into the cameras.

Hartley and Sturm offer an improvement over the midpoint method, with their **polynomial method** [19]. Unlike the midpoint method, it seeks to directly minimize the reprojection error, while requiring that the epipolar constraint is fulfilled.

To understand this constraint, first extend a ray from the camera center in one camera, through the observation \( x_1 \) of the point \( X \). The ray is visible in the other camera’s image plane as the **epipolar line** \( \lambda_2(t) \). If the observation \( x_2 \) lies on \( \lambda_2(t) \), i.e. \( \lambda_2(t) = x_2 \) for some \( t \), the epipolar constraint holds.

Reformulating the problem in terms of the epipolar lines, Hartley and Sturm eventually arrive at the expression

\[
s(t) = d(x_1, \lambda_1(t))^2 + d(x_2, \lambda_2(t))^2
\]

(2.1)

The optimal \( t \) is one of the three minima of the 6th order polynomial equation \( \partial s / \partial t = 0 \).

2.4 Optimal three-view triangulation

The computational complexity of two-view triangulation seems quite manageable, both for the mid-point and polynomial methods. However, three views affords an advantage over two, in that an extra observation can improve the stability of the solution. Moreover, a third camera is an inexpensive addi-
2.5. Fast three-view triangulation

Given three $3 \times 4$ camera matrices $P_i$, observed 2D points $x_i$ and the sought 3D point $X$, Stewenius et al. [37] define the objective function

$$C(X) = \sum_{i=1}^{3} d(P_i X_i, x_i)^2$$

(2.2)

which analogously to the polynomial method, have minimas at $\nabla_X C = 0$, and can be reorganized to a system of three 6th order equations with three unknowns.

This much larger system can be attacked with the Gröbner basis method [36], which rewrites it as a (usually) larger equation system, but with the same roots. However Byråd et al. [5] notes that this approach is poorly conditioned and requires 128-bit floating point numbers, which also makes it extremely slow. They develop modifications that simultaneously relax the constraints and reduces the size by dropping equations, at the expense of allowing some incorrect solutions. This drastically improves the speed and numerical stability and removes the need for high-precision floating point operations.

2.5 Fast three-view triangulation

The methods above are ordered by increasing triangulation accuracy, but also in order of increasing computational cost. Paper A details a three-view triangulation method intended to be much faster, and useful on mobile platforms like robots or mobile phones. Unlike the previous algorithms mentioned in this chapter, this is an iterative approach. To triangulate, it directly minimizes the cost function in equation 2.2 with a nonlinear least squares solver, with the starting solutions provided by the midpoint method. At the same time, the optimizer is not guaranteed to find a global optimum, but as will be demonstrated below, this may not be an issue.

The triangulation problem is closely related to bundle adjustment (BL), simultaneous camera optimization and triangulation, so a natural choice of solver would be the Levenberg-Marquart (LM) algorithm. However, [23] suggests that Powell’s dog-leg (DL) method is more suitable for BL as it converges faster, with almost identical residual error. For this reason, we selected DL over LM, and the experiments indicate that it works well. The DL solver proved to be significantly faster than Byråd et al.’s closed-form approach, at 3.1 µs per triangulated point in single-thread execution on a standard desktop CPU.

In the experiments in paper A, we test the iterative method on both synthetic and real point clouds and 2D observations, and contrast it to both the polynomial method and that of Byråd et al. Some of these results are shown below, represented here by one result each, on synthetic and real data.
2. Triangulation

The synthetic data includes a cube of random 3D points, projected into cameras placed on a circle observing it. Noise distributed as $\mathcal{N}(0, 1)$ [pixels] was added to the image 2D points. The real data is much harder. This is the Notre-dame dataset [35], a reconstruction of the famous cathedral from 715 tourist images. Ground truth 3D coordinates are obviously hard to come by here, so we compare to the points estimated from the complete 3D reconstruction, that optimized all observations and camera poses jointly.

Figure 2.2: Three-view triangulation experimental results.

The results of these experiments are shown in figure 2.2. In the synthetic case (top), all three approaches provide similar performance. In fact, ours and Byröd et al.’s are identical, as one plot covers the other perfectly, suggesting that the iterative approach is on-par with the closed-form solver.

The utility of three-view triangulation over two views is directly apparent on the real data (bottom). Here the polynomial two-view method exhibit an order of magnitude greater 3D placement error, and much greater variance in the reprojection error, compared to the other methods. However, looking closer at the error in 3D placement plot, it seems Byröd’s method have significant failure cases, visible as a distinct mode around $10^0$. It is not clear what is causing this, but one hypothesis is that these failures stem from the eliminated equations in the relaxed formulation. In summary, the experiments indicate that the additional accuracy allowed by three-view triangulation is worthwhile and can be performed both extremely fast and accurately with an iterative solver.
3

Tracking

3.1 Introduction

A triangulation method like that of paper A and chapter 2 is but one part of a 3D reconstruction or ego-motion (odometry) estimation system. It is also necessary to match visual landmarks, observed in different frames at different times. One way to do this is to follow them from frame to frame, perhaps with the Lucas-Kanade tracker [24]. The landmarks might be selected with the feature detector of Tomasi and Kanade [39]. Together these two methods form the Kanade-Lucas-Tomassi (KLT) tracker. This chapter will refer to this function, i.e. following many small regions all over an image, as point tracking.

However there exists a similar task, visual object tracking (VOT). The common use of VOT is to observe one or more larger objects (or targets) visible in video, estimate their evolving location and size and indicate this with a bounding box. Potential applications are varied, from video compression to traffic monitoring and of course object tracking in robotics. However, here and in paper B, VOT is adapted to point tracking, and its performance evaluated.

3.2 The tracking process

We begin by giving a more detailed overview of a typical visual object tracking framework.

In the beginning, the tracker is given one video frame and a bounding box around the target to track. In figure 3.1, this is illustrated by the top-left image. The tracker is now supposed to update the bounding box as as the target moves, even if the latter changes in appearance or size, or is partially occluded by something else. To do so, it begins by encoding the bounding box into an
3. Tracking

Figure 3.1: Generic tracking framework.

internal representation. For our purposes, this label function is a Gaussian centered on the target. In the figure, the label function is exemplified by white blobs, although their sizes are exaggerated for illustration purposes.

Next, the image and label function are stored together in a memory, which is subsequently read by the trainer. It uses this data to build a target model, which is what the tracker uses to recognize the target. It is trained to produce a new label function centered on the target’s new location as new frames arrive. In the figure, the input frames are illustrated as a stack of images in the bottom left.

As the tracking progresses, the view of the target and background will inevitably change in appearance. If the target model is not changed as well, it would eventually fail. To avoid this, new images and newly estimated label functions are periodically added to the memory, so that the trainer can update the target model.

The tracker output is again bounding boxes, that are recovered from the label functions, as shown in the top right corner of the figure.

3.3 Discriminative correlation filters

In the past several years numerous tracking methods have appeared, that implement their target models with discriminative correlation filters (DCFs). Rather than measuring similarity between a target and template as one typically does with correlation or in the KLT-tracker, these filters are optimized to separate some observation of interest from everything else nearby. For tracking, this proved to be a very effective strategy which quickly advanced the state-of-the-art.

MOSSE

The first such method was the Minimum Output Sum of Square Error adaptive filter (MOSSE) filter [3], and the following will outline how it is trained.
3.3. Discriminative correlation filters

To train the filter, assume there exists one or more tuples \((x_i, y_i)\) of training data. In this case, \(x_i\) are gray-scale images containing views of the target to track and \(y_i\) are label functions. As was mentioned earlier, \(y_i\) are chosen to be Gaussian functions with their peaks centered on their corresponding targets.

Training the filter amounts to solving the optimization problem

\[
\arg\min_f \sum_i |f * x_i - y_i|^2. \tag{3.1}
\]

In other words, when convolving the input signal \(x_i\) with the filter \(f\), the output signal should be as similar to \(y_i\) as possible, in the least-squares sense.

Applying \(f\) to an image \(x\) with a view of a target the filter was trained on, but in an unknown location, we get a new label function, which we will refer to as a score map.

\[
s = f * x. \tag{3.2}
\]

This is (ideally) also Gaussian-shaped and with its maximum is centered on the target. In the simplest case, the target is found by locating the maximum response in \(s\).

A key benefit of the MOSSE filter is that it is very fast if formulated in the Fourier domain, as elementwise multiplications are equivalent to convolutions, and because there exists a closed-form optimal solution. There, the optimization problem is

\[
\arg\min_{F^*} \sum_i |F_i^* \odot X_i - Y_i|^2 \tag{3.3}
\]

with the solution

\[
F^* = \frac{\sum_i Y_i \odot X_i^*}{\sum_i X_i \odot X_i^*} \tag{3.4}
\]

where \(*\) and \(\odot\) denotes complex conjugate and elementwise multiplication, respectively.

The score \(s\) of the filter response to the input signal \(x\) given by

\[
s = \mathcal{F}^{-1}(F^* \odot X). \tag{3.5}
\]

In other words, the low-cost operation in equation 3.4 in the Fourier domain produces the filter response to \(x\). The inverse-Fourier transform on this response will produce a score that ideally is a Gaussian function with its peak at the target location.

**Continuous correlation operators**

Paper B describes another DCF-based approach to target tracking, C-COT, and its application to both object tracking and point tracking. This section
3. Tracking

outlines the idea of the C-COT target model. More details are found in paper B and in the dissertation of Martin Danelljan [9].

Paper B first defines an operator $J$ mapping gridded pixels $x$ onto a continuous domain, as

$$J\{x\}(t) = \sum_{n=0}^{N-1} x[n] b\left(t - \frac{T}{N}n\right)$$

(3.6)

where $n$ is the spatial location of the pixel $x[n]$ and $b(t - \frac{T}{N}n)$ is a shifted cubic-spline basis function.

The score function is now continuous,

$$s_f\{x\}(t) = (f * J\{x\})(t)$$

(3.7)

with the optimal filter

$$F = \frac{\sum_i \alpha_i (X_i \odot B)^* \odot Y_i}{\sum_i \alpha_i |X_i \odot B|^2 + \beta^2}$$

(3.8)

in the Fourier domain, where $B$ are the Fourier coefficients of the cubic spline basis function $b$.

Like before, the objective is to build a filter $f$ such that when applied to an image $x$, it produces an output label $s$ with its peak response on the center of $x$. However, the training set $\{(x_i, y_i)\}$ is now extended as the tracker is running, with the most recent $x$ and a new continuous Gaussian label $y$ centered on the target.

In contrast to MOSSE, C-COT has additional sample-importance weights $a_i$ and a regularization term $\beta$. As new samples are added, they are assigned a fixed importance weight. At the same time, weights of earlier samples are decayed exponentially at a constant rate. While simple, this learning and forgetting strategy has been proven to be very effective, and is also used in paper D.

Finally, estimating the peak location with sub-pixel precision is now a two step process. First the coarse integer location is found on the pixel grid, then an optimization problem is set up to find the sub-pixel location with the maximal response.

3.4 Point tracking

This section outlines how C-COT was adapted to point tracking, and tested. Although an object tracking framework could be directly applied to points, some modifications are in order.

C-COT for object tracking trains its target models on features from a deep neural network [34]. It is certainly possible to use them here as well, but these are actually less than ideal in this particular application. First of all, deep feature maps generally have much lower resolution than the image itself, and
3.4. Point tracking

Figure 3.2: Qualitative point tracking results. Ground-truth tracks are green, C-COT tracks are red.

This prevents small spatial details from being tracked, as they tend to blend into one another. Second, deep features are typically represented as high-dimensional vectors, and this is potentially computationally very expensive to work with when there are many points to track. Instead, we use gray pixel values. Not only is the resolution high, gray values are also minimally expensive to work with, and it makes fair and direct comparisons with MOSSE and the KLT tracker very easy, as both used gray values in their original formulations.

The other modification is the addition of a coarse-to-fine search pyramid in three levels, tracking the points at multiple scales, to allow for large frame-to-frame motion.

The resulting method is tested on the MPI-Sintel dataset [4], a collection of video sequences from the animated short film Sintel. Although it is synthetic, Sintel is rendered with realistic lighting, atmospheric effects, and both motion and focus blur to make it more challenging. This dataset is equipped with dense ground-truth optical flow, and is widely used for training and benchmarking dense optical flow methods, including the recent [38]. As our task is distinctly separate from dense optical flow, we initialize our ground-truth starting points from the flow by first detecting good features to track [33]. The flow in these points is then accumulated forward in time to create
3. Tracking

the tracks. Unstable points on object boundaries are removed by checking whether the flows in their neighborhoods are divergent.

Our tests are performed on two variants of the C-COT tracker. The first has a memory and evolving target model, as with the VOT version. The second variant (named Ours-FF in figure 3.3 below) is retrained from scratch every frame and is consequently more prone to drifting. We compare the performance to the classic KLT tracker and MOSSE, both also employing three-level coarse-to-fine tracking to allow for large motions. The methods’ accuracies are given by the endpoint-errors (EPEs) of the tracked points, measured independently for every track, in every frame. EPE is commonly used to benchmark optical flow methods, and is simply the Euclidean distance between estimated and ground-truth point coordinates, measured in pixels.

A qualitative example from Sintel, that compares inlier tracks (red) to the ground-truth (green) is shown in figure 3.2. Quantitative experimental results are shown in figure 3.3.

Figure 3.3: Quantitative point tracking results.

The plot on the left contains the distribution of EPEs of the four methods over the dataset. The plot on the right is the precision, which in this case is the fraction of EPEs below a threshold indicated on the x-axis. The values in parenthesis are the average inlier EPE (left plot) and precision at the inlier threshold (right plot). A point is considered an inlier, if the EPE is less than three pixels.

From the figure, it is apparent that C-COT with filter updates has its largest fraction of EPEs well below 0.1 pixels. This appears to be the case for the KLT-tracker as well, but the lower fraction and the precision plot reveal that it has a much longer tail of outlier tracks, suggesting that C-COT tracker is much more robust. The MOSSE tracker has worse error distribution, possibly because it does not have any sub-pixel refinement. Nevertheless, the precision plot reveals that it is still more robust than KLT, as its precision above one pixel is on par with C-COT.
4

Localization

Figure 4.1: The test platform.

4.1 Introduction

Historically people navigated long distances by looking at the stars and the sun. This has now mostly been replaced by GPS, but for shorter distances and familiar surroundings, we still look for known landmarks. When traveling on water near the coast, it makes sense to look towards the shore to determine your location. This is the topic of paper C, where we develop a method to look at the horizon and subsequently find the most likely location in a digital elevation model (DEM). A DEM is a specific kind of geographic map, where each pixel represents the altitude at the location covered by the pixel. However, the camera is on the ground, while the DEM provides a birds-eye view and a drastically different perspective. Somehow both views must be transformed
4. Localization

into representations that can be compared and this chapter outlines the steps taken and the considerations made to do so.

Starting from the ground view, it is clear that the first steps should involve both detecting and estimating the shape of the horizon. At this point in time, we may attack a vision problem with one of two categories of approaches; the classical and the one built on neural networks. The strength of the former category is that methods generally need little or no training. The strength of the latter is that methods can be made very robust to irrelevant image details. Developing the localization method of paper C we found it useful to combine both, and developed two methods for the same purpose. The first method, built on classical techniques, was designed to provide training data for the second, trained method. These are outlined below.

4.2 Horizon regression with classical methods

Figure 4.2: Top: Panorama from the Karlskrona archipelago, not corrected for camera pitch and roll. Bottom: Corrected panorama. Horizon and water profiles estimated by the classical approach shown in green and red, respectively.

In paper C, the experiments are based on data captured in a 16 by 16 kilometer region in the Karlskrona archipelago in Sweden, and a DEM of the same region. The data included panoramic videos, GPS location tracks and compass readings collected from a small remotely operated boat, shown in figure 4.1. With the captured video we extract horizon profiles from the videos with the method of [15] which is outlined next. A horizon profile is the curve across an image, where land and sea meets the sky.

In a panoramic video with a cylindrical projection model, the horizon will appear like an S-shaped curve across the frame, as shown in the top part of
4.2. Horizon regression with classical methods

This occurs when the world is projected onto a camera-aligned cylinder which is tilted relative to the surface of the ocean. If the observed horizons are reprojected onto an ocean-aligned cylinder, the S-shape will disappear.

It is reasonable to assume that the horizon profile has clearly defined edges between sky, land and sea, and a classic approach to finding edges is the Canny edge detector [8]. However, this detector will not only mark the horizon, but also any other edge-like structures. To filter out spurious edges, [15] applies another classical approach, Hough voting, for curve detection [13]. The detected curve is expressed as the horizon’s normal vector parameterized as the camera’s pitch and roll angles.

With the camera orientation estimated, the next step is to find the horizon profile. As the purpose of this method is to generate training data, it is acceptable to take a short-cut and exploit the camera’s location and heading, as indicated by the GPS and onboard compass\(^1\). With this information, a virtual camera is placed in the DEM. Rays are then traced out in every direction from the camera, as is illustrated in figure 4.3. The highest recorded elevations in the direction of each ray are subsequently organized into a virtual horizon.

![Figure 4.3: Horizon generation through DEM raytracing from a set position, sweeping through all angles \(\psi\), capturing the highest point in every direction as \(h(\psi) = \tan(\theta_i)\)](image)

Finally, the camera pitch and roll parameters \((\theta, \phi)\) as well as the heading \(\psi\) are found by minimizing

\[
\arg\min_{\theta, \phi, \psi} \sum_i D\left(\pi(\psi)\pi(\theta, \phi)\pi^{-1}(h_i)\right)
\]

Here, \(D\) is a function that measures the distance to the nearest edge, as determined by the Canny method and \(h_i\) are points on the virtual horizon. \(\pi\) is a projection operator mapping 3D space onto the image plane (cylinder) and \(\pi(\theta, \phi)\) is a 3D rotation and \(\pi(\psi)\) is effectively a horizontal shift in the image plane.

\(^1\)To be fair, an accelerometer recording would have provided the camera attitude as well.
4. Localization

An example of the corrected horizon, with estimated horizon and water profiles, is visualized in the bottom part of figure 4.2. Further details of this procedure are found in paper C and in the dissertation of Bertil Grelsson[14].

4.3 Horizon regression by semantic segmentation

The classical approach outlined the previous section can determine both camera orientation and the shape of the horizon profile. However, it is brittle, and requires a fair number of manual adjustments to produce quality results. This is why we now set out to replace it with neural networks, that are less sensitive to noise and appearance variations. But, before diving into this, a short digression is in order.

Feature extractors

As is probably well known, the AlexNet [22] neural network architecture won the classification category the Large Scale Visual Recognition Challenge (ILSVRC) in 2012, where the goal is to correctly classify the dominating object in a large number of images in the ImageNet dataset [12]. AlexNet could provide the correct answer among its top-5 suggestions 83.6 percent of the time, which was a huge improvement over the previous winner [32] at 74.3 percent.

With its eight layers and 60 million parameters, this kick-started the research into large-scale neural networks. Predictably, AlexNet was quickly superseded by other architectures such as VGG [34] and ResNet [20]. In particular, a 152-layer ResNet [20] managed to reach a top-5 accuracy of 96.4 percent just three years after AlexNet. Subsequently, ResNets were found to be useful in a whole range of computer vision applications, including in paper C discussed in this chapter and papers D and E treated in chapters 5 and 6.

An outline of the ResNet architecture and its five stages is shown on the left hand side of figure 4.4. Each stage, except the first one, is made up of multiple stacked residual blocks. A slightly simplified form of one such block is shown on the right hand side. The residual block represents one of the main ResNet novelties. Its identity mapping pathway helps preserving the
Horizon regression by semantic segmentation

input signal and enables the building of deeper networks without the vanishing gradient problem.

To recognize an object in an image, a network must somehow transform descriptions of the object’s color, shape and texture into more abstract representations of the object type. To do so, neurons need to gather visual information from an area large enough to cover at least part of the object, possibly the entire image depending on its size. However, a neuron capable of seeing a complete image would be very expensive to store and use, and so would their activation maps, (also known as feature maps). A practical solution is to progressively reduce the resolution as image features, (pixels in the first layer) are transformed in layer by layer, moving deeper into the network. At increasing depths, the ability to localize the object is lost, while semantic understanding is gained.

A task that needs to know about object appearances, including edges and simple shapes, can extract features from shallow layers at high spatial resolution. Another task more concerned with the object class, can extract features with semantic meaning but lower resolution, from deep layers. When given a different purpose, the classifier at the bottom of the layer stack is generally removed. What is left, is referred to as a feature extractor or a backbone.

Horizon attitude normalization and segmentation

We now return to the discussion on finding and regressing the horizon from a panoramic image. We keep the task organization of the classical method, and design one network for camera orientation estimation and another for horizon profile regression. Both are built on top of a ResNet-50 backbone with the classifier removed, to supply deep features to our new designs from the input images.

One possible issue with these features, is that the backbone is trained on ImageNet to understand its one thousand classes of “things”. We are more interested in amorphous regions of sky, water and land, also known as “stuff” [7], which are not part of the ImageNet classes. Nevertheless, it is reasonable to assume that ImageNet already has imagery with “stuff” in it, and it is likely beneficial for a classification network to understand the context (i.e. “stuff”) where it correlates to some “thing”. One relevant example could be that boats occur on water more than anywhere else. Given that the output feature maps have 2048d vectors, we assume that some portion of that feature space is dedicated to describe “stuff”.

With deep features available, the next step is to decide how to design the two networks, and we start with the camera orientation network, in paper C referred to as HorizonFinder. It is provided panoramic images where the horizons are warped into S-curves, and is supposed to predict the two camera orientation angles so that the warping can be removed. A natural choice is to implement the output layer as a set of fully-connected neurons, that each see
4. Localization

all backbone features at the same time. This is similar to how one would set up
a network for image classification, though the output variables are continuous
and not categorical.

Now if the backbone features were applied directly to this output layer,
its neurons would have to be constructed with several million weights each,
even though the backbone feature maps are of much lower resolution than
its corresponding input images (1/32nd of the original on both axes). Having
this many weights is likely unnecessary and might even fail, as the concepts of
sky, water and land must be understood as the S-curve from which angles can
be extracted. Consequently, there is a greater chance of it working correctly
if the transformation is allowed to span multiple network layers.

Finally, there is some auxiliary information that can help shape the new
network and reduce the number of weights required in the last layer. First, it
is safe to assume that there is a horizon profile stretching horizontally across
the frame, and the network only needs to concern itself with one or or two lo­
cations per column, where sky, land and water meet. Consequently, it should
be possible to pool features in the same column and reduce the vertical res­
olution. At the same time, this will enlarge the spatial context available to
subsequent neurons, potentially improving their performance. Second, with
only three concepts (sky, land, water) to keep track of, we should be able to
project them from the 2048d backbone feature space into one with consider­
ably fewer dimensions.

The requirements and problem-specific circumstances outlined above,
can be compressed into a very simple design with only two convolutional lay­
ers, shown in figure 4.5. The network output angles are expressed as sine and
cosine pairs, which limits the output range to $[-1, 1]$ and entirely avoids the
modulo $2\pi$ of raw angle values.

![Figure 4.5: The camera orientation regression network.](image)

The requirements on the second network, named HorizonSegmenter in
paper C, is very similar to that of the first, although some adjustments are
made. This network is outlined in figure 4.6. First, as the camera orientation
angles are known, we can correct the individual images from the panoramic
camera and segment those individually, without stitching. Second, it is use­
ful to allow for more detail along the horizon. This is why two layers that
double the resolution horizontally, are inserted into the network. The output
layer, finally, is a fully-connected network producing two sets of vertical pixel coordinates to mark the horizon and water profiles across the input frame$^2$.

Figure 4.6: The horizon segmentation network.

### 4.4 Location estimation

With the two networks, we can now capture a scene from video and then robustly rectify and segment it into sky, land and water, and extract a horizon profile. This is a vector $h(\psi)$, with one element per direction $\psi \in \{0, \delta, 2\delta, \ldots, 2\pi - \delta\}$. Its values are the tangents of angles $\theta$ between sea level and land elevation, from the point in the world where the horizon is observed.

#### Horizon matching

The similarity of $h(\psi)$ to virtual horizons is subsequently estimated with 1-D MOSSE filters, and the location with the highest matching score is expected to be the correct one.

Each of the filters is location-specific and is trained from a set of virtual horizons $h_i(\psi)$ rendered from a 50 by 50 m neighborhood around the location, here indexed by $i$. This data augmentation acts as a regularization, reducing the trained filter’s sensitivity, allowing it to better match the query horizon profile despite slight variations in appearance. Similar to visual object tracking, the target label is a Gaussian function that the filter is optimized to output.

An example map of filter responses centered on the true location in the archipelago DEM, is visualized in figure 4.7. Green and blue indicate ray-traced land and water profiles as seen from the center of the map. Gray scale pixels indicate elevation in locations where no filter response has been evaluated.

Evaluated locations are colored red, with higher brightness indicating stronger matches, although the peak response in the center is significantly stronger than the visualization suggests. The actual response used here is not the raw filter output, but its peak to side-lobe energy ratio. This suppresses spurious detections in locations where the tallest peak is only barely stronger than the background.

$^2$In hindsight, a convolutional output layer would probably have been better as it does not require the input to be of a specific size.
4. Localization

Figure 4.7: MOSSE filter responses from matching horizon profiles in the Karlskrona archipelago in Sweden.

Some additional considerations

To have the approach work well in practice, there are some small but critical issues that must be accounted for.

First, the amplitudes of both the computed and observed horizon profiles will depend on the distance to the land mass, which is analogous to an object’s scale in a 2D image. A MOSSE filter applied to visual object tracking is sensitive to scale changes, and that weakness is carried over to this problem. Far from land, mismatched query and template scales do not present much of an issue, but when approaching a shore, the observed horizon profiles will rapidly change in height.

However, a good way to normalize the horizon profiles is to divide it with its maximum amplitude if any point $h_i(\psi) > 1.0$. The condition prevents amplification of noise in places far from any landmass. With the normalization, the specificity of the filter response is greatly increased.

Second, a MOSSE filter in the Fourier domain is translation equivariant and in this case the filters are equivariant to the direction $\psi$. Despite this, we encountered spurious peaks at other angles when attempting to find the correct yaw over all $360^\circ$ of the filter response. For this reason, we restricted the search for a filter peak response within a $\pm 4^\circ$ band around the compass heading. This band is wider than the expected deviation from the true head-
4.4. Location estimation

Figure 4.8: Qualitative comparison of localization with GPS (red circles) and our method (yellow crosses).

ing in the actual compass used in our experiments, and could be made even wider if necessary.

**Performance evaluation**

The performance of the method was evaluated on separately collected video sequences and GPS tracks. Without ground-truth, the estimates were compared to the GPS recording instead, and were found to have a mean deviation of 2.47 m and a standard deviation of 1.26 m. This is at least as accurate as a consumer-level GPS device, within the archipelago testing area. A qualitative comparison is made in figure 4.8.
5

Video object segmentation

5.1 Introduction

Solving the image segmentation problem entails dividing images into coherent regions. The semantic segmentation problem has the additional condition that the regions are defined by the kind of object found inside. Going one step further, video object segmentation (VOS) seeks to accurately segment out a moving object in video. The target can be of any class and have any appearance, which can change drastically as it moves, due to deformations, occlusions and lighting changes.

VOS currently comes in three flavors; unsupervised, semi-supervised and interactive, which differ by how the target is specified at the start. An unsupervised method is supposed to select a target by itself, for example by choosing the most salient object it can see. An interactive method expects a human operator to annotate the first frame by drawing sparse “scribbles” to roughly indicate where the target and background are. Dense segments are then derived from these hints. A semi-supervised method explicitly needs full segmentation mask to be provided with the first video frame. This is the most generic of the three in that the mask generation is left unspecified, and is what will be discussed here.

5.2 VOS approaches

VOS has been around for a fairly long time; an early example is [18] from 1997, but we will ignore that and start from 2016 when DAVIS [28] (for Densely Annotated VId eo Segmentation), the first of the two current major datasets/benchmarks, was published and the first CNN-based approaches appeared.
5. Video object segmentation

An early method to appear alongside DAVIS is one-shot video object segmentation (OSVOS) [6]. This approach repurposes the VGG network [34] but removes the final classifier layer. After transfer-learning on DAVIS, the network is fine-tuned during inference on the first video frame, to output the correct target segmentation mask. Although very simple, it yielded good results on the 2016 edition of the DAVIS benchmark. However, by its construction it cannot adapt to changes to the target appearance, and the size of the network itself makes it slow to initialize.

A later method, that can adapt to changes, is the Reference-guided mask propagation (RGMP) [26] approach. The authors of this method introduced what they referred to as a Siamese encoder-decoder architecture, which is illustrated in figure 5.1

![Siamese encoder-decoder VOS architecture](image)

Figure 5.1: The siamese encoder-decoder VOS architecture.

This figure is not showing any detail of the internal workings of RGMP, but it outlines a structure that appears in several VOS methods, including that of paper D. On the left is an encoder network to process the input, which are images and possibly segmentation masks, into deep features. The same network is applied to both the reference frame (top branch) and the target frame (bottom branch) to extract deep features.

The reference frame is the first image of the video sequence, and a given mask to start the segmentation. The target frame is a later video image to be segmented. Features from both branches are subsequently merged in a method-specific model (center box), possibly producing output in the form of embedding-vectors in some latent space, typically at lower resolution than the input video. These embeddings are then transformed by the decoder into a full-resolution mask as the output.

In the case of RGMP, the encoders is a ResNet-50 network. The network is pretrained, but the first layer is extended to four input channels rather than three, to allow the starting target mask to be encoded alongside the RGB-image. In the bottom branch, the video image is complemented with the mask propagated from the previous frame. Features from the reference branch are used to guide the RGMP-model to attend to similar features in the new video frame.
A third example is the space-time memory (STM) VOS method [27]. It is the first use of transformers [40] in video object segmentation, and it also employs a Siamese encoder-decoder architecture. Unlike RGMP however, STM maintains a memory of multiple image and mask pairs encoded into key-value tuples.

The target image, without a mask, is separately encoded into another key-value tuple. The two sources are combined with the transformer self-attention mechanism, where the memorized keys in conjunction with the target image key, control the merging of the memorized and target values. Finally, the merged values are decoded into a new mask. The memory can subsequently be extended with newly predicted segments. This approach is very effective and have inspired multiple followups, but the ResNet-50 encoder network is trained from scratch, which requires additional training data.

5.3 Fast and robust target models

We now turn to the method of paper D, which generalizes visual object tracking into segmentation, with discriminative correlation filters to separate the target objects from the background.

Our VOS framework, shown in figure 5.2 merges the tracking structure in figure 3.1 on page 16, with the Siamese encoder-decoder architecture in figure 5.1. The latter allows the target model to operate on deep features.

![Figure 5.2: Our VOS architecture.](image)

Like with tracking, the trainer generates a target model from image features and a label function provided when a new target appears. However, in tracking it is sufficient to train the DCF with a Gaussian label function, so that it can produce a peak at the target location. We now replace it with a pixel-accurate mask, so that the DCF will produce similarly pixel-accurate segments. But unlike STM, our method does not encode masks alongside the image, as they are needed as label functions when training the target model.

As was mentioned earlier, tracking DCFs are implemented in the Fourier domain to reduce the number of calculations. However, the implied circular convolution causes wrap-around effects near the edges and is a significant
5. Video object segmentation

drawback of that approach. These must be mitigated by windowing or by taking special care when training the filter, as was done in [10]. In addition, it is unfortunately the case that the Fourier transform is actually rather inefficient on GPUs, which are needed to efficiently work with deep features.

Fortunately, individual deep features have significant discriminative power. There is no longer any need to depend on the appearance of a target and background over a spatial region, as some of this information is already encoded in a single deep feature vector. Consequently, there is little to gain from training a filter with significant spatial extent, and no longer any benefit to operate in the Fourier domain. The results in paper D show that a $3 \times 3$ filter is quite sufficient for this purpose, which of course is also very efficient on GPUs.

Additional efficiencies are gained by factorizing the target model into a projection $W_p$ and the actual $3 \times 3$ filter $W_f$, like

$$H_W(x) = W_f \ast (W_p \ast x)$$

The factorization drastically reduces the number of trainable parameters, as $W_p$ projects the 1024-dimensional deep features into 96 dimensions. After the parameters have been trained once, $W_p$ is excluded from further training, which allows $W_f$ to be trained even faster during subsequent updates.

Training the target model

As our DCF is now to be trained in the spatial domain, we have no neat closed-form optimal solution available. Rather, the filter is now trained with an optimizer, using a quadratic loss function

$$L(W) = \sum_k \| v_k(y_k - U(H_W(x_k))) \|^2 + \lambda_1\|W_p\|^2 + \lambda_2\|W_f\|^2$$

Here, $U$ is a bilinear upsampling function that matches the spatial resolution of $H_W(x_k)$ to the label function (segmentation mask) $y_k$. $W$ is shorthand for both filter parameters concatenated.

A question remains regarding the training, and that is the choice of optimizer to minimize the loss and train the filter parameters. One straightforward option could be gradient descent optimization, with the update function $W = W - \alpha \nabla L_W$. However, in the development of the DiMP tracker [1], it was discovered that gradient descent training on deep features converged slowly. They opted for the much faster-converging the Gauss-Newton optimization method, and we also adopted this approach.

An interesting trick here, is that the automatic differentiation machinery of a machine learning framework, could be applied in the optimizer. This simplified the implementation greatly since the gradient functions did not have to be calculated manually.
5.3. Fast and robust target models

**Recovering high resolution**

A problem with the target model described above, is that the output segmentation \( s = H_W(x) \) it generates, is only 1/16th of the original size. This is the resolution of the deep features \( x \) provided to the DCF from the ResNet backbone network.

To recover the full resolution, we adapt the DFN semantic segmentation network [42], illustrated in figure 5.3. This network has four stages, each receiving an image feature map \( x_i \), from a ResNet-101 backbone in our case. The network generates its own attention to these features by globally pooling the output of the previous stage into a “channel-attention” vector. After progressing through the stages, from deep to shallow features of progressively higher resolution, the output is a segmentation map.

![DFN segmentation network](image)

Figure 5.3: The DFN segmentation network [42], with the ResNet backbone on the left.

When DFN is applied to semantic segmentation, the network determines which of twenty classes it should assign to each pixel, without outside supervision. For our purposes though, we would rather have the target model decide whether a pixel belongs to the target or the background. To do this we introduce a new block, the target segmentation encoder (TSE). At the input of each DFN stage, the TSE injects the low-resolution target segmentation mask \( s \) into the network, by resizing and concatenating it with the backbone features. This is illustrated for one decoder stage, in figure 5.4.

The outputs provided by the decoder is now of much higher resolution, but still only 1/4th of the original image size, so we upsample a factor four as a final step. In paper D, this is performed by a pair of convolutional layers and \( 2 \times \) bicubic interpolations, but is replaced in paper E with the guided upsampler of [38].
5. Video object segmentation

![Diagram](image-url)

Figure 5.4: One stage of the segmentation decoder, built from a target segmentation encoder (TSE) block and a DFN stage. $s$ is the low-resolution segmentation mask from the target model. $x^d$ are the features output from stage $d$. $\uparrow x^d$ is a spatial up-sampling to match the resolution of feature map $x^d$.

**Merging multiple objects**

So far, we have only discussed the process of segmenting a single object, but what if there are multiple objects of interest at the same time? For this purpose we will now refer to $s$ as a **score map** and let segmentation mask be a binary mask created from it.

First let us consider the score map $s$ produced by the decoder network as log-likelihoods. This means that each pixel $s$ is assumed to be the log-likelihood that the object is present in that location. These can be transformed into probability maps, with $p = \sigma(s)$, and the final segmentation mask is then given by $p > 0.5$. Here $\sigma$ refers to the sigmoid function.

To merge multiple objects $i = \{1, 2, ..., N\}$ that were predicted in separate score maps $s_i$, [26] suggested that the softmax function could be applied to normalize multiple observations into probabilities. Softmax has previously been applied to both image classification and semantic segmentation as a way to merge independent predictions into per-class probabilities. For the purposes of video object segmentation [26] defines the merging function as

$$p_i = \frac{\tilde{p}_i / (1 - \tilde{p}_i)}{\sum_j \tilde{p}_j / (1 - \tilde{p}_j)}$$

(5.3)

where $\tilde{p}_i$ is the single-object likelihood (i.e $\sigma(s)$) of object $i$. The background is given the class $i = 0$, but as there is no separate prediction it is defined as $p_0 = \prod_j (1 - \sigma(s_j))$, i.e. the probability that there is no other object present.

After probabilities have been determined, each pixel is finally assigned the object identity $i$ of the class with the highest probability $p_i$, forming a map of merged segmentation labels $\{0, 1, ..., N\}$.

5.4 Performance evaluation

The methods mentioned so far are compared in figure 5.5. The y-axis score is the mean of two evaluation metrics adopted for video object
5.4. Performance evaluation

Segmentation. The first is the intersection-over-union metric, also known as the Jaccard distance (abbreviated \( J \)), which measures how well the predicted segment area overlaps the ground truth segment area. The second is referred to as the F-score and is intended to measure how well the predicted segment’s edge adheres to the ground-truth edge. A perfect score in either case, is 100%. For more details, see [28].

At 76 percent, our method is approximately 7 percentage points below STM, but is significantly faster at 22 frames per second.

Figure 5.5: Mean per-object frame rate vs mean \((J+F)/2\) score, on the DAVIS 2017 validation dataset.
6

Segmentation with distractors

6.1 Introduction

Paper D was subsequently extended into the learning-what-to-learn (LWL) method [2] for video object segmentation. The main feature of LWL is that, unlike the method in paper D, it can backpropagate through the target model trainer. Taking advantage of this, LWL employs a new neural network, the label encoder. It transforms the segmentation masks into multi-channel embeddings and provide those as training labels for the target model. These labels have the same resolution as the feature maps provided to the target model trainer which, as was mentioned earlier, is just 1/16th resolution of the segmentation masks. However, this does not cause any loss of fidelity.
6. Segmentation with distractors

A possible explanation why this is the case, is found in figure 6.1, showing a segment mask encoded into a multi-channel embedding. It seems higher-resolution details have been encoded as the responses of directed filters. This is reasonable as the decoder network is responsible for recovering fine detail. Providing it with better edge information should improve the results. Another advantage of the multi-channel embeddings, is the possibility to train the target model with additional information. We take advantage of this in paper E, to reduce the problem with *distracting objects*.

What distractors are, and why they would be a problem is illustrated in figure 6.2. The top shows frames from a video sequence with multiple skiers entering the frame from the right, one at a time, and leaving to the left. Each skier is associated with its own color in the segmentations in the middle and bottom rows. In the middle row, when the fourth target (blue) has appeared, it is confused with the earlier green and yellow targets. This is visualized by the multiple colors on the same segmentation mask, in particular in the fourth frame counting from the left. In comparison, the segments in the bottom row are generated by the method of paper E, and the identity problem in frame four has mostly disappeared.

![Distractor example](image)

Figure 6.2: Distractor example. Middle row and bottom rows show the results without and with distractor modeling, respectively.

To understand what is happening in the example, consider that the target model is trained to determine which side of a boundary in feature space (a hyper-plane), that a deep image feature falls on. Features from the target should be on one side of this boundary, and features that are not, on the other side. If the trainer had taken the distractors into account, the boundary would likely have been placed differently.

---

1The last skier to appear in the fifth frame, is not annotated and should be ignored. However, it appears to be very similar to the green skier in frame one, and was likely segmented for that reason.
6.2 Incorporating distractors

There are many possible ways to mark distractors, including, e.g. hard example mining, where objects with deep features similar to the target are detected and added as distractors to the training of the target model. As the number of targets in a video can vary, so can the number of distractors. This is problematic as deep neural networks are generally defined with a fixed number of inputs.

In paper E, we address both issues by defining the the distractor segment of target $i$ as the union of all other known targets $j \neq i$. The segmentation map is then extended to a second channel to hold the distractor, before it is transformed into an embedding.

However, as the distractor to any one target is an amalgamation of all the other targets, it is reasonable to assume that it incorporates a more diverse set of deep features compared to a single object. In addition, we know the target model has fairly limited expressive power, and it could be difficult for it to maintain a correct segmentation of the distractor over time.

Fortunately, we can sidestep these issues easily, in particular since perfectly accurate distractor masks are unnecessary. First, we can regenerate distractors from all newly predicted objects, after a new frame has been processed. Second, the confidence of a pixel being a distractor need not be high and it is acceptable to incorrectly classify a distractor as background and vice-versa.

To merge targets into a distractor, we let the the most certain predictions of target and background “win” in every pixel. Unlike softmax aggregation, this does not normalize the probabilities, so uncertain predictions can remain uncertain.

**Distractor generation**

More formally, we approach this as follows:

Let $p_t_i(x) \in \mathbb{R}^{H \times W}$ be the target segment probability map of the target with index $i \in \mathbb{I}$. This can either be the network decoder output passed through a sigmoid activation function, or set to zero or one from the ground-truth mask pixels, if the target is new.

Now let

$$p_{max}(x) = \sup_j p_{t_j}(x) \quad \forall j \in \mathbb{I}, \tag{6.1}$$

$$p_{min}(x) = \inf_j p_{t_j}(x) \quad \forall j \in \mathbb{I}. \tag{6.2}$$

This merges the highest and lowest probabilities (per pixel) of all target maps, into $p_{max}$ and $p_{min}$.

Then let $L(x) \in (\mathbb{I} \cup \{0\})^{H \times W}$ be the map of merged segmentation labels after softmax-aggregation, with zero being the background label.
Finally let

\[ 1_f(x) = \begin{cases} 
1 & \text{if } L(x) > 0 , \\
0 & \text{if } L(x) = 0 .
\end{cases} \quad (6.3) \]

indicate regions with any foreground pixel, and

\[ 1_d(x) = \begin{cases} 
1_f(x) & \text{if } L(x) \neq i , \\
0 & \text{otherwise} .
\end{cases} \quad (6.4) \]

indicate distractors of target \( i \). A new probability map of distractor \( i \) is generated as

\[ p_{d_i}(x) = 1_d(x)p_{\text{max}}(x) + (1 - 1_f(x))p_{\text{min}}(x) . \quad (6.5) \]

**Training**

To not unnecessarily over-constrain the training, we partially disable the computation of the loss in training samples with a single target and no distractors. Specifically, we require the distractor probability to be zero in areas under the target, but allow it to take any value elsewhere.

With this modified training loss, the decoder is allowed to spontaneously create its own distractor masks. An example of this is found in the “camel” sequence of DAVIS, shown in figure 6.3. In this video, there is only one target, the camel in the foreground. Yet, when the camel in the background appears, it is marked as a distractor by the decoder network.

![Decoder output with a both target and a distractor channel present. Red/yellow indicates positive log-likelihoods, blue/cyan negative. Larger magnitudes are brighter.](Image)
6.3 Performance evaluation

Finally, we can place papers D and E in context to the state-of-the-art, figure 6.1 shows the improvement over time, of mean $J$ and $F$ score improvement. The methods were evaluated on the DAVIS 2017 validation split, and includes both the methods mentioned before, i.e OSVOS [6], RGMP [26], STM [27] and LWL [2], as well as the more recent method CFBI [41]. Both STM and CFBI were considered state-of-the-art at the time of writing.

Table 6.1: $(J+F)/2$ scores on the DAVIS 2017 validation set for paper D and E, as well as other methods.
Concluding remarks

This chapter provides a summary of the results and reflects on possible paths forward.

7.1 Results

As was noted in the introduction chapter, this thesis has two distinct directions. Papers A and B move in the direction of geometry and 3D reconstruction while the theme of papers B through E is to explore the uses of discriminative correlation filters to both low-level (e.g. segmentation) and high-level (e.g. localization) computer vision problems.

In paper A we showed how three-view triangulation through optimization, can be made significantly more robust and precise, and yet be fast compared to algebraic approaches. These properties could perhaps allow the method to be employed as a faster alternative to bundle-adjustment for 3D reconstruction. Camera pose estimation still requires joint optimization with observed landmarks, but it need not be performed over the full set of points.

In paper B we developed a highly robust and precise 2D point tracker based on discriminative correlation filters and demonstrated that it outperforms the Lucas-Kanade method. This can be applied in the same context as the triangulation of paper A.

Moving away from geometry, we introduced a new application of DCFs by tackling localization in paper C. We showed how discriminative correlation filters are sufficiently powerful for precise place recognition in a large coastal region, given only horizon profiles. This approach can be applied as a backup solution when satellite navigation is unavailable.

Paper D introduced a third application for DCFs, generalizing visual object tracking to video object segmentation. In this paper it was demonstrated that a correlation filter can exploit deep features to robustly and efficiently
7. Concluding remarks

classify target and background in video, while staying updated to appearance changes. As this method is real-time, it can readily be employed in other contexts with minimal performance impact. Segmentation is a low-level vision task, and as such its main application would be as an attention mechanism to filter out unwanted visual features.

Paper E extended the discriminative model introduced in paper D, to concurrently handle both a target and distracting objects in the same filter, useful in situations where multiple targets have very similar appearance.

7.2 Future work

With the papers in this thesis as a starting point, there are several interesting avenues of research to explore.

For example, as plain discriminative correlation filters are currently applied as linear classifiers, they are fast to train. However the capability disparity compared to neural networks is quite large. It would be useful to study additional DCF formulations on the continuum between the very fast linear DCFs on one end, and powerful but slow neural networks on the other. For example, one possibility is to construct filters with residual branches, and determine whether this improves their discriminative power, and at what additional computational and data cost.

Another avenue of research related to segmentation, is self-supervised detection and training of distracting targets. This would help the DCF maintain a minimal margin between classes over time and reduce the risk of target model drift.

On the applied side, it would be interesting to merge segmentation, tracking and triangulation to create real-time, dense 3D reconstructions of individual objects. If regions of interest are segmented out first, there should be fewer outliers in the reconstruction, leading to less wasted computation and more accurate results. Also, given deep features with semantic meaning, DCF-based segmentation could possibly be applied to part-based processing, potentially needed for dense 3D reconstruction of deformable or piecewise-rigid objects.

With regards to the localization application, the current bottleneck is the brute-force matching. However, if we consider horizon-profiles to be analogous to raw image pixels, it would be interesting to see whether the place recognition process can be moved into a space of deep-feature embeddings. Such representations could be more compact, better suited to the specific application, and very likely much faster.
Bibliography

[1] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. “Learning discriminative model prediction for tracking.” In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, pp. 6182–6191 (p. 34).

[2] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. “Learning What to Learn for Video Object Segmentation.” In: Computer Vision - ECCV 2020. 2020 (pp. 8, 9, 39, 43).

[3] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. “Visual object tracking using adaptive correlation filters.” In: 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE. 2010, pp. 2544–2550 (p. 16).

[4] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. “A naturalistic open source movie for optical flow evaluation.” In: European conference on computer vision. Springer. 2012, pp. 611–625 (p. 19).

[5] Martin Byröd, Klas Josephson, and Kalle Åström. “Fast optimal three view triangulation.” In: Asian conference on computer vision. Springer. 2007, pp. 549–559 (p. 13).

[6] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. “One-shot video object segmentation.” In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, pp. 221–230 (pp. 32, 43).

[7] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. “Coco-stuff: Thing and stuff classes in context.” In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 1209–1218 (p. 25).
[8] John Canny. “A computational approach to edge detection.” In: IEEE Transactions on pattern analysis and machine intelligence 6 (1986), pp. 679–698 (p. 23).

[9] Martin Danelljan. Learning Convolution Operators for Visual Tracking. Vol. 1926. Linköping University Electronic Press, 2018 (p. 18).

[10] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. “Learning spatially regularized correlation filters for visual tracking.” In: Proceedings of the IEEE international conference on computer vision. 2015, pp. 4310–4318 (p. 34).

[11] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. “Beyond correlation filters: Learning continuous convolution operators for visual tracking.” In: European conference on computer vision. Springer. 2016, pp. 472–488 (pp. 6, 65).

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database.” In: 2009 IEEE conference on computer vision and pattern recognition. 2009, pp. 248–255 (p. 24).

[13] Richard O Duda and Peter E Hart. “Use of the Hough transformation to detect lines and curves in pictures.” In: Communications of the ACM 15.1 (1972), pp. 11–15 (p. 23).

[14] Bertil Grelsson. Vision-based Localization and Attitude Estimation Methods in Natural Environments. Vol. 1977. Linköping University Electronic Press, 2019 (p. 24).

[15] Bertil Grelsson, Michael Felsberg, and Folke Isaksson. “Highly accurate attitude estimation via horizon detection.” In: Journal of Field Robotics 33.7 (2016), pp. 967–993 (pp. 22, 23).

[16] Bertil Grelsson, Andreas Robinson, Michael Felsberg, and Fahad Shahbaz Khan. “GPS-level accurate camera localization with HorizonNet.” In: Journal of Field Robotics 37.6 (2020), pp. 951–971 (pp. 6, 85).

[17] Bertil Grelsson, Andreas Robinson, Michael Felsberg, and Fahad Shahbaz Khan. “HorizonNet for visual terrain navigation.” In: 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS). IEEE. 2018, pp. 149–155 (p. 7).

[18] Chuang Gu and Ming-Chieh Lee. “Semantic video object segmentation and tracking using mathematical morphology and perspective motion model.” In: Proceedings of International Conference on Image Processing. Vol. 2. IEEE. 1997, pp. 514–517 (p. 31).

[19] Richard I Hartley and Peter Sturm. “Triangulation.” In: Computer vision and image understanding 68.2 (1997), pp. 146–157 (p. 12).
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition.” In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, pp. 770–778 (p. 24).

[21] Johan Hedborg, Andreas Robinson, and Michael Felsberg. “Robust three-view triangulation done fast.” In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014, pp. 152–157 (pp. 5, 55).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks.” In: Advances in neural information processing systems 25 (2012), pp. 1097–1105 (p. 24).

[23] MLA Lourakis and Antonis A Argyros. “Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?” In: Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1. Vol. 2. IEEE. 2005, pp. 1526–1531 (p. 13).

[24] Bruce D Lucas, Takeo Kanade, et al. “An iterative image registration technique with an application to stereo vision.” In: Vancouver, British Columbia. 1981 (p. 15).

[25] Kristoffer Öfjäll, Michael Felsberg, and Andreas Robinson. “Visual autonomous road following by symbiotic online learning.” In: 2016 IEEE Intelligent Vehicles Symposium (IV). IEEE. 2016, pp. 136–143 (p. 9).

[26] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. “Fast video object segmentation by reference-guided mask propagation.” In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 7376–7385 (pp. 32, 36, 43).

[27] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. “Video object segmentation using space-time memory networks.” In: Proceedings of the IEEE International Conference on Computer Vision (CVPR). 2019 (pp. 33, 43).

[28] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. “A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation.” In: Computer Vision and Pattern Recognition. 2016 (pp. 31, 37).

[29] Andreas Robinson, Abdelrahman Eldesokey, and Michael Felsberg. “Distractor-aware video object segmentation.” In: Submitted. 2021 (pp. 8, 123).
Bibliography

[30] Andreas Robinson, Felix Järemo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. “Learning Fast and Robust Target Models for Video Object Segmentation.” In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, pp. 7406–7415 (pp. 7, 109).

[31] Andreas Robinson, Mikael Persson, and Michael Felsberg. “Robust accurate extrinsic calibration of static non-overlapping cameras.” In: International Conference on Computer Analysis of Images and Patterns. Springer. 2017, pp. 342–353 (p. 9).

[32] Jorge Sanchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. “Compressed fisher vectors for large-scale image classification.” In: Rapport de recherche RR-8209, INRIA (2013) (p. 24).

[33] Jianbo Shi et al. “Good features to track.” In: 1994 Proceedings of IEEE conference on computer vision and pattern recognition. IEEE. 1994, pp. 593–600 (p. 19).

[34] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In: International Conference on Learning Representations. 2015 (pp. 18, 24, 32).

[35] Noah Snavely, Steven M Seitz, and Richard Szeliski. “Photo tourism: exploring photo collections in 3D.” In: ACM siggraph 2006 papers. 2006, pp. 835–846 (p. 14).

[36] Henrik Stewenius. “Gröbner Basis Methods for Minimal Problems in Computer Vision.” PhD thesis. Lund University, 2005 (p. 13).

[37] Henrik Stewénius, Frederik Schaffalitzky, and David Nistér. “How hard is 3-view triangulation really?” In: Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1. Vol. 1. IEEE. 2005, pp. 686–693 (p. 13).

[38] Zachary Teed and Jia Deng. “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow.” In: Computer Vision – ECCV 2020. 2020, pp. 402–419 (pp. 19, 35).

[39] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Tech. rep. features. Technical Report CMU-CS-91-132, Carnegie, Mellon University, 1991 (p. 15).

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is All you Need.” In: NIPS. 2017 (p. 33).

[41] Zongxin Yang, Yunchao Wei, and Yi Yang. “Collaborative video object segmentation by foreground-background integration.” In: European Conference on Computer Vision. Springer. 2020, pp. 332–348 (p. 43).
[42] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. “Learning a discriminative feature network for semantic segmentation.” In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 1857–1866 (p. 35).
Part II
Publications
Papers

The papers associated with this thesis have been removed for copyright reasons. For more details about these see:

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-174939
