TRACKING WITH DENSE CORRESPONDENCES

Doctoral Thesis

Ing. Jonáš Šerých

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

Supervisor: prof. Ing. Jiří Matas, Ph.D.

Study Programme: Electrical Engineering and Information Technology (P2612)
Field of Study/Specialization: Artificial Intelligence and Biocybernetics (3902V035)

Prague, September 2024

ABSTRACT

Long-term point tracking is a computer vision task, in which query points given in a template frame are to be located throughout a video. Tracking of a dense set of query points enables applications that would not be possible with standard standard bounding-box-level or segmentation-level tracking. In this thesis, three instances of the dense long-term tracking task are addressed. First, a novel method for estimating homographies from optical flow achieves state-of-the-art planar tracking performance. Second, two novel methods for dense point tracking – tracking of all points from a template frame – are proposed. The trackers combine optical flows estimated between both adjacent and distant frames to form long-term tracks, and achieve good performance while tracking fast. Last, a novel coin-tracking task is introduced, together with a baseline coin-tracking method and a coin-tracking benchmark. In coin-tracking, the target objects are flat and two-sided. Which of the two sides is currently visible changes frequently, leading to new challenges that mostly do not occur in common planar object tracking.

Keywords: visual tracking, dense correspondences, long-term optical flow, point tracking, planar object tracking, coin-tracking

ABSTRAKT

Dlouhodobé sledování bodů je úloha počítačového vidění, ve které se mají body označené v jednom snímku videa lokalizovat v celém videu. Sledování hustě rozmístěných bodů umožňuje aplikace, které by nebyly možné se standardním sledováním na úrovni obdélníků ohraničujících objekt nebo segmentačních masek. V této práci se řeší tři úlohy dlouhodobého sledování husté sady bodů. Za prvé, nová metoda odhadu homografie z optického toku dosahuje špičkových výsledků ve sledování plochých objektů. Za druhé jsou navrženy dvě nové metody pro sledování všech bodů z daného vzorového snímku. Obě kombinují optický tok odhadnutý mezi sousedními i mezi vzdálenými snímky do dlouhých trajektorií a dosahují dobrých výsledků i rychlosti. Nakonec je představena úloha coin-tracking společně se základní coin-tracking metodou a datovou sadou pro vyhodnocování. V úloze coin-tracking jsou sledované objekty ploché a dvoustranné. To, která z obou stran je právě viditelná, se často mění, což vede k novým výzvám, které se při běžném sledování plochých objektů obvykle nevyskytují.

Klíčová slova: vizuální sledování, husté korespondence, dlouhodobý optický tok, sledování bodů, sledování plochých objektů, coin-tracking

ACKNOWLEDGEMENTS

I am grateful to my whole family for supporting me all these years and for creating a safe space where I could grow, especially to my wife Anička, who bravely endured all of my high-stress periods and gave birth to our wonderful children Anežka, Josífek, and Bětka. Also, thanks for the support from both of my Christian fellowships.

Many thanks to my advisor Jiří Matas for his patience, for coming up with suitable experiments when I was stuck, for all the night hours spent together polishing the papers before submission deadlines, and for taking into account that I have a family. I also appreciate the broad range of random facts about dogs, linguistics, history, and many other topics he shared with me.

Big thanks go to my colleagues from TLab. Thanks to Michal for enduring all my stupid jokes, sharing many good ones, training the MFT/IQ networks, performing tons of experiments, and sharing all the joys of Ph.D. study. Thanks to Štěpán for sharing my deep learning enthusiasm, for the ice creams, and for always being calm, relaxed, and smiling. Thanks to Jéňa for all the lunch discussions, all his well-prepared teaching materials, and for leading the RPZ lab-teacher team. Thanks to Tomáš for his funny grumpiness and brutal jokes and for sharing the ups and downs of being a father.

I am also grateful for all the other awesome people I met at CMP, including the administrative staff and people from G2, who tolerated my children-caused higher priority in The Queue. I also appreciate all the great teachers at CTU and AG.

Thanks to Toyota Motor Europe for financing a large part of my studies. My research was also supported by CTU student grants SGS17/185/OHK3/3T/13 and SGS20/171/OHK3/3T/13, Technology Agency of the Czech Republic project TH0301019, and by the Research Center for Informatics project CZ.02.1.01/0.0/0.0/16_019/0000765 funded by OP VVV.

Thank God I could do all this.

CONTENTS

1

INTRODUCTION

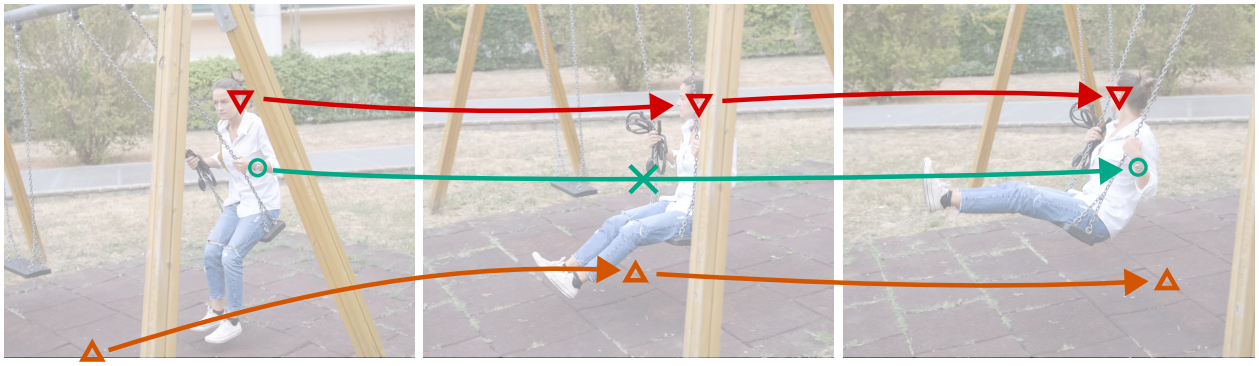


Figure 1: Sparse point tracking in a video. The task is to locate query points *(in the left image)* throughout the video and indicate if they are visible. The middle *(green circle)* point is occluded in the middle image. Images from the *swing* video in the DAVIS [1] dataset.

One of important problems in computer vision is establishing local correspondences between images. Given a pair of images I_a, I_b capturing overlapping parts of a scene (possibly taken with different cameras, from different viewpoints, at different times), the task is to automatically find corresponding pairs of image coordinates $(x, y) \leftrightarrow (x', y')$, such that the image I_a at the coordinates $(x, y) \in \mathbb{R}^2$, and the image I_b at the coordinates $(x', y') \in \mathbb{R}^2$ depict the same scene point. Establishing such correspondences enables further computer vision processing, like camera pose estimation, reconstruction of a 3D model of the captured scene, creating panoramas from overlapping low field-of-view pictures, object tracking, motion prediction, and so on.

In this thesis, we deal with establishing correspondences in a video sequence. A video is a sequence of $T \in \mathbb{N}$ images $(I_i)_{i=1}^{T}$ captured in quick succession[1], *e.g.* at 30 frames per second (FPS). In the short time between the frame acquisitions, the camera pose, the focal length, the illumination, and the scene do not change much, if at all. This makes finding correspondences between nearby frames a simpler task than in the general two-image correspondence problem. Even when these changes accumulate throughout a video, the task is simpler thanks to the information provided by the intermediate frames.

When dealing with video, the correspondences are sequences of image coordinates $((x_t, y_t))_{t=1}^{T}$ or a set coordinates paired with frame numbers $\{((x_t, y_t), t), \dots\}$ when the correspondences are not

[1] We ignore time-lapse videos, which have different properties.

Figure 2: Video editing with dense long-term correspondences. The edits (*red text*) made on the first frame (*left*) were automatically propagated to the rest of the video and "stick" to the surface and its deformations. Best viewed as a video: https://cmp.felk.cvut.cz/~serycjon/MFT/visuals/lioness.mp4

established in each frame. The task typically is to *track*, *i.e.*, locate throughout a video, a given set of *query points* (fig. 1). This is related to estimating correspondences between images, but it is a different task. Whereas in the case of pairs of images we were looking for finding *some* corresponding points, here we want to know locations of given points specified in one of the frames. In consequence, we have to deal with point *visibility* as the query points may be occluded or out of the camera field of view in some frames. Note that it may be possible to estimate the point location even when the point is not visible, based on contextual information like the motion of nearby points.

Both estimating correspondences between image pairs and tracking points in a video are usually done *sparsely*, *i.e.*, only a relatively small number of correspondences is estimated and only a relatively small set of query points is tracked.

In this thesis we focus on tracking *densely*, *i.e.*, we want to efficiently track *every* point in a query frame or a query object template. The dense tracking is a generalization of the *optical flow* estimation problem, in which the positions of all the points in one frame of a video I_t are to be tracked into the next frame $I_{(t+1)}$. One of the problems this thesis addresses is how to go from two-frame optical flow estimation to *long-term* tracking throughout the video and in the presence of occlusions.

For texture-less symmetric objects (chapter 5) where point tracking may be impossible we also consider tracking by segmentation. The task is then to densely estimate which pixels belong to the tracked object, without estimating fine-grained point-to-point correspondences.

1.1 MOTIVATION / APPLICATIONS

Although tracking points in a video can be a useful building block for various computer vision tasks, the dense long-term tracking studied

in this thesis is particularly well applicable to video editing. In film post-production and special visual effects, the artists frequently need to edit the video locally, *i.e.* altering just some part of the scene, as opposed to global edits, like brightness, contrast, or color adjustment of the whole frame. Doing the edits manually on each frame would be infeasible and the results would be jittery and not temporally stable. Having a dense tracker allows the artist to do the edits on one frame of the video and propagate them automatically to the whole video, such that they "stick" to the surfaces in the scene as shown in fig. 2. Visual tracking is part of standard post-production tools, like Adobe After Effects, BorisFX Mocha Pro, DaVinci Resolve Fusion, and others.

Another interesting application is tracking in surgical videos [2, 3, 4]. Dense correspondences can be used to provide augmented reality for the surgeons, *e.g.*, highlighting spots that were not inspected in detail yet, showing various guidance markers, or overlaying the image with data from other sensors.

1.2 THESIS OVERVIEW

In this thesis, we address three instances of the problem of tracking with dense correspondences. In chapter 3, we deal with planar object tracking, where the goal is to track all points on the surface of a rigid planar (flat) object or surface. The motion of all the surface points is described by a simple geometric model (homography) when the video is captured by a standard projective camera. In fact, the task is slightly more general, because the homography also covers the case where a non-planar target position is constant with respect to the camera and the only motion is caused by rotation of the camera around its center. We use the 8-degree-of-freedom (DoF) homography motion model to filter out correspondence estimation noise and suppress gross errors (outliers). Our approach is to estimate the homography from dense optical flow correspondences. We train a neural network to predict a quality score for each correspondence and use it as weights for weighted least squares homography fitting. The resulting planar object tracker achieves state-of-the-art results (as of the time of writing of this thesis) on multiple planar object tracking benchmarks [5, 6, 7, 8], including one published after the tracker [6]. We have published the code, the trained model, and improved ground-truth annotations of a standard planar-tracking benchmark [8]. The chapter is based on

J. Šerých and J. Matas, "Planar object tracking via weighted optical flow," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1593–1602, 2023.

In chapter 4, we address a more general problem of tracking all pixels of a reference video frame. Unlike in planar object tracking, the

query points may belong to multiple independently moving 3D objects of unknown shape. Also, the motions in the scene are no longer constrained to be rigid. The problem is similar to optical flow estimation but extended to long video sequences. This introduces the need to handle occlusions, non-trivial illumination and appearance changes, and longer motions. The proposed dense point trackers achieve good benchmark performance while tracking orders of magnitude faster than state-of-the-art alternatives. This line of work was first published in

M. Neoral, J. Šerých, and J. Matas, "MFT: Long-term tracking of every pixel," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6837–6847, 2024,

extended in

T. Jelínek, J. Šerých, and J. Matas, "Dense matchers for dense tracking," in *Proceedings of the 27th Computer Vision Winter Workshop (CVWW 2024)*, 2024,

and further improved in

J. Šerých, M. Neoral, and J. Matas, "MFTIQ: Multi-flow tracker with independent matching quality estimation," *under review*, 2025.

In chapter 5, we go back to planar objects and introduce a novel task called coin-tracking, in which a thin double-sided planar object, like a coin, a playing card, or a smartphone, is to be tracked from both sides. The rotation of such objects poses new challenges, including a flip between the two sides. A thin object like a playing card can become practically invisible during the flip. Without understanding the underlying 3D structure, the tracked object seems to be disappearing (due to the out-of-plane rotation) and suddenly a possibly very different object seems to start appearing. In typical coin-tracking videos, the target objects are dynamic, and their motion is the primary source of the tracking challenge. This contrasts with the standard planar object tracking task, where the targets are usually static, and the camera movement is most significant.

The coin-tracking was chronologically the first part of our research, started during my MSc studies. In the master thesis [13], we first introduced the coin-tracking task and proposed a coin-tracking method. The coin-tracking related materials in this PhD thesis are based on the following paper

J. Šerých and J. Matas, "Visual coin-tracking: Tracking of planar double-sided objects," in *German Conference on Pattern Recognition*, pp. 317–330, Springer, 2019.

We have significantly extended the MSc work in the following ways. First, we compare the proposed CTR coin-tracking dataset to standard visual object tracking datasets, showing some of its unique challenges, like large and fast changes in aspect ratio of the target objects and low textureness of the targets. Second, we significantly improved the CTR dataset ground truth, increasing the frequency of annotations from every 30th frame to every 5th frame and switching from bounding-box to segmentation annotations. Finally, we introduce and experimentally evaluate a novel coin-tracking method CTR-BASE.

We have set a new state-of-the-art in planar tracking (chapter 3) and achieved very good results in dense point-tracking (chapter 4), yet the coin-tracking problem has not been satisfactorily solved yet (even in its simplest form in which templates of both sides of the target are available) and remains an open challenge.

1.3 CONTRIBUTIONS

The contributions of this thesis are:
WOFT [9]. A state-of-the-art planar object tracker with 8DoF homography pose representation.
POT-210 re-annotation [9]. Improved ground-truth annotations on a uniformly spaced subset of the standard planar object tracking benchmark POT-210 [8]. The original ground truth was not sufficiently precise to reliably benchmark current state-of-the-art methods.
Coin-Tracking task and benchmark [14]. Publication of the coin-tracking task (work started in the master thesis) and the CTR coin-tracking benchmark with segmentation mask annotations.
CTRBase [14]. A baseline coin-tracking method, which, however, still outperforms current dense point trackers on the coin-tracking task.
MFT [10]. A dense point tracker that extends the RAFT optical flow to enable long-term tracking. It achieves good benchmark results and tracks densely orders of magnitude faster than published alternatives.
MFTIQ [12]. An improved version of MFT that allows for plug'n'play integration of arbitrary optical flow. It outperforms the original MFT with RAFT and achieves results close to the point-tracking state-of-the-art.

Both MFT and MFTIQ are a joint effort with Michal Neoral. We both participated in all parts of the research and development process. I implemented the trackers, caching mechanisms, evaluation, and visualization tools. Michal implemented and trained the neoral networks.

2

RELATED WORK

First we review concepts and related work that appear in all the problems we address. Specific related work and methods competing with the proposed approaches are reviewed in related work sections in chapter 3, chapter 4, and chapter 5.

2.1 OPTICAL FLOW ESTIMATION

The optical flow (OF) estimation is a classical computer vision task. Given two *consecutive* frames of a video with spatial resolution $H \times W$, the OF is a $H \times W$ array of point-to-point correspondences encoded by point-position differences (Δ_x, Δ_y) between the two frames. Some methods also estimate a $H \times W$ occlusion map, indicating whether the point is visible in the second frame or not. But most often, the occlusions are ignored, and the methods attempt to estimate the optical flow even when the point is occluded or out-of-view in the second image, based solely on contextual clues.

2.1.1 *Optical Flow vs Motion Field*

Originally, the OF was representing the apparent movement of brightness patterns between the frames [15]. The OF methods were based on the brightness constancy assumption, *i.e.* that the brightness of a pixel corresponding to a particular 3D point does not change between two consecutive frames. To resolve ambiguities an additional smoothness constraint had to be used. Instead of dealing with the brightness of each pixel separately, [16] considers the brightness patterns in a small neighborhood of the pixel. Neither the brightness constancy assumption, nor the smoothness assumption hold in practice.

Modern deep learning OF methods estimate the *motion field* of the scene instead. Assuming that each pixel corresponds to a single 3D scene point, the motion field represents the difference between the position in the first image and the position in the second image that shows the same 3D point.

The two quantities, *i.e.* the original optical flow and the motion field are not the same. For example in a static-camera video of a single color non-textured sphere rotating around its axis, the optical flow is zero everywhere, because the image brightness does not change at all. On the other hand, the motion field represents the rotation of the sphere. From this example it is clear that the motion field estimation task is ill-posed - the video is indistinguishable from another one

where the sphere rotates with different speed, around different axis, or doesn't move at all. In the configuration where both the camera and the sphere are static, but a light source moves, resulting in a specular highlight moving on the sphere, the motion field is zero, but the original optical flow represents the motion of the specular highlight.

Nowadays the optical flow methods, *e.g.* [17, 18, 19, 20, 21, 22, 23], estimate the *motion field*, but call it *optical flow*. Also in this thesis, when we say *optical flow* we mean the motion field, *i.e.* the 2D projection of the motion of a 3D point.

The assumption that each pixel represents only one 3D point is not correct in practice, because of effects like (partial) transparency, reflections, mixed pixels on object boundaries, or blur. Amodal OF [24] represents multiple motions per pixel, but is not widely used.

2.1.2 *Modern Optical Flow Methods*

Traditional optical flow estimation methods [15, 16, 25] were based on optimization of some hand-crafted objective function. The modern methods based on deep-learning instead train a neural network to regress the optical flow. In the simplest form, FlowNetSimple [20], the two input images are concatenated and the resulting tensor of size $H \times W \times 6$ is passed through a multi-layer convolutional network with the optical flow on the output. Most recent OF methods are instead based on *correlation cost-volume* introduced in FlowNetCorr [20]. The two input images are not concatenated and Convolutional Neural Network (CNN) features are extracted instead for each of them. Then each feature vector from one image gets compared to feature vectors in the other image by computing a dot-product of each such feature pair. The resulting map is called the *correlation cost-volume*. In an ideal case it would be enough to select the position with the biggest cost-volume value, *i.e.* the most similar, for each pixel in the first image. However the cost-volume is typically processed by another neural network regressing the final flow. Some methods [26, 21, 23] use a local cost-volume, in which the feature similarities are computed only within a small neighborhood of each pixel from the first image. These methods are multi-scale, first estimating the OF on a low spatial resolution and iteratively refining it to the final high-resolution result.

The RAFT [22] optical flow computes the full 4D $\frac{H}{8} \times \frac{W}{8} \times \frac{H}{8} \times \frac{W}{8}$ cost-volume once, followed by an iterative procedure that samples the cost-volume in some neighborhood of the current flow estimate and computes an updated estimate. Current state-of-the-art OF methods add various improvements over the original RAFT. The GMA [19] OF adds a transformer block that enables modeling long-range relation between the motion of distant pixels. The FlowFormer [27] and FlowFormer++ [28] use transformers also for image feature extrac-

tion and the iterative flow estimation. Another direction is adding multi-scale processing, like in MS-RAFT+ [29, 30].

Another recent trend in optical flow estimation is taking into account more than just the two video frames between which the flow is to be estimated. ContFlow [23] uses the OF estimated between previous two frames as an input into the flow regression network. In VideoFlow [18], the flow is estimated between triplets, or quintuplets of frames in both forward and backward direction. MemFlow [17] maintains a feature representation in a small memory buffer and updates it in a learned way on each frame of the video.

In all the deep-learning methods the pixels are represented by some feature embedding computed by a neural network with typically large receptive field. The currently used features are unable to represent only the object in the center of the receptive field and they incorporate the surrounding context and background in a unpredictable black-box manner. This sometimes causes the flow to be incorrectly influenced by the background and/or other objects, or even to completely ignore the central object, especially when it is relatively thin.

2.1.3 *Optical Flow datasets*

Tho OF neural networks are trained mostly on synthetic data, typically on the FlyingChairs[20], FlyingThings3D[31], and MPI Sintel[32, 33]. The first two contain chairs and random everyday objects from [34] respectively, randomly moving in the view. Although the resulting pairs of images are very far from being realistic, these datasets proved to be helpful for the OF training. The MPI Sintel (which is often called just Sintel) dataset contains scenes from the open source movie Sintel, whose authors have released all the source files needed for rendering. Thanks to this, selected scenes could be re-rendered with additional optical flow and occlusion (a $H \times W$ array storing a binary visible/occluded state of each pixel) outputs. Compared to the FlyingChairs and the FlyingThings3D datasets, Sintel contains more realistic scenes and non-rigid motions. Apart from being a standard dataset for OF training, MPI Sintel also contains a test set and defines a benchmark on it. Inspired by Sintel, the recently published Spring[35] dataset contains renders from the Spring open source movie, which contains more fine structures, *e.g.* fur and grass, and is rendered at high resolution.

The reason for using mainly synthetic data for OF training and evaluation is practical impossibility to obtain OF ground truth for real-world videos. There are two works attempting to measure flow ground truth. The authors of the HD1K dataset [36] acquired videos with a car-mounted camera on two selected streets. The two streets were first scanned with a LIDAR without people or moving cars, providing a 3D model of the whole scene. Next, videos were captured

including people and moving cars. The OF (pseudo) ground truth was measured from 2D to 3D correspondences and estimated camera poses. This process does not work with dynamic objects, so these were manually segmented and marked as invalid.

The KITTI dataset [37] also contains traffic scenes captured from a car, but on larger area. The (pseudo) ground truth of the static parts of the scenes was constructed by transforming LIDAR point-clouds with rotations and translations from a GPS/IMU device combined with ICP fitting of the LIDAR 3D point-clouds. Unlike HD1K, KITTI also provides a (pseudo) ground truth for some moving objects. In particular, the authors have fitted parametric CAD models of cars to the point clouds. Projecting the model 3D points into two frames gives the flow correspondences. The (pseudo) ground truth provided by both of these datasets is generated by measurement and thus contains errors, hence we call them *(pseudo)* ground truth. Also the annotations are only semi-dense, *i.e.* not every pixel has a ground-truth flow.

The main evaluation metric is the end-point-error (EPE), which is the euclidean length of the difference between the estimated and the ground-truth flow

$$EPE\big(\Delta_x, \Delta_y\big) = \|\big(\Delta_x, \Delta_y\big) - \Big(\Delta_x^*, \Delta_y^*\Big)\|, \tag{1}$$

or metrics derived from it, like the fraction of points with EPE below certain threshold. Apart from the overall mean EPE metric, the Sintel benchmark also measures the EPE on occluded (*EPE unmatched*) and unoccluded (*EPE matched*) flows separately. However, flow methods are not expected to predict occlusion masks and the standard benchmarks do not evaluate it, so most methods completely ignore the occlusion problem. Note that it is often possible to estimate the optical flow correctly even during occlusions, based on the motion of nearby non-occluded points. Some methods, *e.g.* [26, 23, 38], estimate occlusions internally to aid the accuracy of the optical flow, but they do not provide occlusion maps as outputs and do not evaluate the accuracy of the estimated occlusions.

We use optical flow as a component in our dense point trackers (chapter 4). Detecting the optical flow occlusions is an essential part of the proposed trackers, which enables long-term stable tracking.

2.2 FEATURE MATCHING

Feature matching is another important computer vision task. The goal is to estimate point-to-point correspondences in a pair of images capturing the same 3D scene. Unlike the optical flow task, the two images are now not consecutive frames of a video. Often they are captured independently, with different cameras, from different viewpoints and at different times — so called wide multiple baseline stereo (WxBS) problem. Traditionally, some keypoints [39, 40, 41]

Figure 3: Example pairs of frames from synthetic optical flow datasets. Left to right: FLYINGCHAIRS [20], FLYINGTHINGS3D [31], and MPI SINTEL [33].

or regions [42] were identified first, before finding matching pairs between the two images.

More recently, several methods [43, 44, 45, 46, 47, 48] estimate the correspondences densely, *i.e.* for each pixel. In addition to an $H \times W$ map of corresponding coordinates, they also output an $H \times W$ score map that can be thresholded to reject pixels without a match. Similar to the optical flow methods (see section 2.1.2), these dense feature matching methods use the cost-volume representation (in case of [48] the cost-volume computation is hidden inside the transformer cross-attention block). However, they are not trained on consecutive frames of synthetic videos, but rather on pairs of real-world photos. These typically come from the MEGADEPTH [49] dataset of famous buildings and other landmarks, where a (pseudo) ground-truth depth and camera calibration is available via COLMAP [50, 51] 3D reconstruction. The training pairs have wide baseline compared to the optical flow data, but lack complex motions as the whole 3D-reconstructed scene is static and usually has simple geometry, *e.g.*, a façade. Some methods [45, 46, 52] also use self-supervised training with warp consistency constraints and spatial training image augmentation.

2.3 VISUAL OBJECT TRACKING

In the visual tracking task, the goal is to establish correspondence in a video, but on an object level, instead of pixel level. Traditionally [53, 54, 55], the target was represented by a 2D bounding box (rectangle), usually with fixed aspect ratio or fixed scale. More recently the focus has shifted to segmentation level trackers [56, 57, 58, 59, 60], which represent the target by a segmentation mask, *i.e.*, a per-pixel target / background classification.

Most of the published trackers are *short-term*, meaning that the target is at least partially visible during the whole video. In *long-term* tracking the target may become fully occluded or out-of-view and some kind of re-detection is required to resume the tracking after

it reappears. The long-term trackers report the visibility status flag on top of the target pose. Note that this long-term / short-term classification [61] is independent on the length of the tracked videos. The methods proposed in this thesis are all long-term in this sense.

There exist many variations of the tracking task depending on the type of the tracked object, number of tracked objects, camera motion, availability of some a-priory knowledge of the target, *etc*. For example in multi-object tracking (MOT) [62, 63] there are multiple targets of the same type (usually people or cars) to be tracked at the same time. The targets are first detected by a class-specific detector, the tracking task is then to consistently assign the detections to the targets.

2.3.1 *Visual Object Tracking Benchmarks*

The progress in single-object model-free short-term tracking is captured by the Visual Object Tracking challenge (VOT) held annually since 2013 [64]. Large scale tracking benchmarks [65, 66] became available more recently. The segmentation level tracking became popular after the introduction of the DAVIS [67] video object segmentation benchmark in 2016. The segmentation trackers are also evaluated on YouTubeVOS [68] or MOSE [69].

There are different evaluation metrics, but they are usually based on the Intersection-over-Union (IoU) score – both for the bounding-box and for the segmentation tracking. The IoU, also called the Jaccard index, measures the similarity between the ground truth and the predicted region by computing the areas of their intersection and union and taking their ratio. With the set of pixel positions representing the ground truth named A and the set of pixel positions representing the prediction named B, the IoU is computed as

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{2}$$

Various statistics can be computed based on the IoU scores, *e.g*. computing its mean over all frames (mIoU), thresholding IoU to detect number of tracking failures, or varying the IoU threshold and plotting the fraction of frames with failure, followed by computing the Area Under Curve (AUC) of the curve.

3

PLANAR OBJECT TRACKING

In this chapter we address the rigid planar object tracking problem and present WOFT [9], a planar tracker which is state-of-the art on multiple benchmarks, namely POIC [5], POT [7, 8], and PlanarTrack [6], the last of which was published after WOFT. We also analyzed the ground truth quality of POT and because it was not high enough to properly compare state-of-the-art methods, we precisely re-annotated a uniformly spaced subset of frames and published the corrected ground truth. The annotation was done under my supervision by Mrs. Larisa Ivashechkina using an annotation tool I have created for very precise homography annotation.

3.1 INTRODUCTION

In planar rigid object tracking, the object pose is related to its initial pose by an 8 degrees-of-freedom (DoF) homography when using a perspective camera, and the target is fully specified by the initialization mask. Planar trackers can output precise 8-DoF object poses, enabling applications not possible with bounding-box or segmentation mask level trackers, in areas such as film post-production, visual servoing [70, 71], SLAM [72], or markerless augmented reality [73, 74, 75]. Man-made objects are often either completely planar or consist of planar surfaces, allowing for planar object tracking in a wide range of scenarios.

Current state-of-the-art methods struggle on seemingly toy-like sequences in standard planar object tracking datasets, POT-210 [8] and POIC [5]. The target planarity poses challenges, *e.g.*, strong

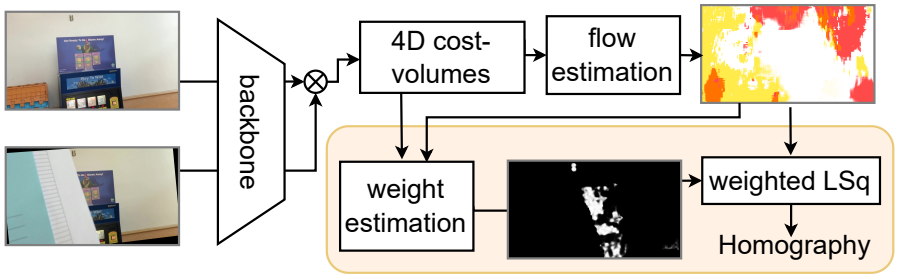


Figure 4: Planar object tracking with a homography estimated by a novel weighted least squares (LSq) homography module called WFH (*orange box*) on optical flow correspondences. The proposed trainable flow weight CNN assigns a weight $w_i \in [0, 1]$ to each flow vector based on samples from correlation cost-volume.

perspective distortion, significant illumination changes caused by specular highlights, and motion blur caused by a shaking hand-held camera.

In this chapter, we introduce a novel model-free planar object tracker. The proposed method estimates dense correspondences between the template (initial image) and the current image with a deep optical flow network. A novel homography estimation module then assigns a weight to each optical flow correspondence, and a homography is estimated as a solution to a weighted least squares problem. The network assigns low weights to incorrect flow vectors and thus it is not necessary to use robust outlier detection algorithms like RANSAC.

Using dense OF correspondences has several advantages. First, OF estimation is well researched and high-quality methods are available off-the-shelf. Second, the dense per-pixel correspondences help on low-textured objects, where sparse key-point correspondences fail. Finally, having dense correspondences enables us to compute a homography correspondence support set and detect a tracking failure if the support is small.

The proposed homography estimation procedure is fully differentiable, allowing us to train both the weight estimator and the optical flow network using homography supervision. The main contributions of this work are the following.

- We propose a novel fully differentiable homography estimation neural network module.
- We propose a novel planar target tracker employing the weighted flow homography estimation (code public[1]).
- The proposed tracker sets a new state-of-the-art on the POT-210 [8], POT-280 [7], and POIC [8] datasets, performing well across all challenge types. On POT-210, the tracker error is half of the error of the best competing method.
- We analyze the ground truth on the POT-210 dataset and publish[1] a precise re-annotation of its subset. The inaccuracy of the original annotation accounted for half of the errors of the proposed tracker.

3.2 GEOMETRY OF PLANAR OBJECT TRACKING

In this section we describe the geometry of the planar object tracking.

PROJECTIVE SPACE. A point in 2D Euclidean space is usually represented by coordinates $(x, y) \in \mathbb{R}^2$, however it can also be represented in the *homogeneous coordinates* $(x, y, 1)$. In this so called *2D projective*

[1] https://cmp.felk.cvut.cz/~serycjon/WOFT

space $\mathbb{P}^2$, all the coordinate triplets $(\lambda x, \lambda y, \lambda), \lambda \neq 0$ represent the same 2D point (x, y). The points with coordinates $(x, y, 0)$ are called *points at infinity*, or *ideal points* and they represent intersections of parallel lines. Lines are represented by homogeneous coordinates $\vec{l} = (a, b, c)$ with $\vec{l}^\intercal \vec{x} = 0$ for all points $\vec{x}$ on the line. All the points at infinity lie on the *line at infinity* $\vec{l}_\infty = (0, 0, \lambda), \lambda \neq 0$. This extension with the points and line at infinity makes manipulation with point and lines simpler, because there are no special cases — any two lines meet at a single point, any two points lie on a single line. In the same way the 3D Euclidean space $\mathbb{R}^3$ can be extended to the 3D projective space $\mathbb{P}^3$.

PINHOLE CAMERA MODEL. To describe how the points in the 3D world are projected into a 2D image, we assume the usual *perspective camera model*. It is derived from the working of an ideal *pinhole camera*, in which the light from the captured scene passes through a single point. This model captures the working of commonly used real-world cameras well, unless the camera has a wide-angle fish-eye lens.

The projection is performed by a rank 3 matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$ as

$$\begin{bmatrix} x' \\ y' \\ \lambda' \end{bmatrix} = \mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3}$$

The 3D point $\mathbf{X} = (x, y, z)$ is first transformed into homogeneous coordinates $(x, y, z, 1)$ and then multiplied by the projection matrix $\mathbf{P}$ to get homogeneous coordinates of the projected 2D point. In the standard case, $\lambda' \neq 0$, we can recover the image 2D Euclidean coordinates (u, v) by simply dividing by λ'.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\lambda'} \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{4}$$

PLANE-INDUCED HOMOGRAPHY Let's now consider projection of points lying on a plane such that $(1, 1, 0)^\intercal \mathbf{X} = 0$, *i.e.* with zero z-coordinate $\mathbf{X}_{(3)} = 0$. Now

$$\begin{bmatrix} x' \\ y' \\ \lambda' \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \\ | & | & | & | \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_4 \\ | & | & | \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5}$$

The projection of points on that plane simplifies to multiplying the homogeneous form of the 2D in-plane coordinates by a *homography* matrix $\mathbf{H} \in \mathbb{R}^{3\times3}$. Any plane can be converted to this case by changing the world coordinate system appropriately. Since $\mathbf{P}$ is a full-rank matrix, $\mathbf{H}$ is also full rank and thus invertible. With two cameras, their

projection matrices $\mathbf{P}_1, \mathbf{P}_2$ and the homographies $\mathbf{H}_1, \mathbf{H}_2$ from the 3D plane to the image, we can construct a homography $\mathbf{H}_{12}$ mapping directly from the first into the second image by first transforming from the image in the first camera to the 3D plane and then back to the second camera.

$$\mathbf{H}_{12} = \mathbf{H}_2 {\mathbf{H}_1}^{-1} \tag{6}$$

The homography matrix has nine parameters, but since it maps into homogeneous coordinates where scale does not matter (see Eq. (4)), it has only eight degrees of freedom. A homography mapping is typically estimated from at least four pairs of correspondences $((u_1, v_1), (u_2, v_2))$ between the two images of the plane, where (u_1, v_1) and (u_2, v_2) are the pixel coordinates of a 3D point lying on the plane projected into the respective cameras.

3.2.1 *Robust Homography Estimation*

Usual way to estimate a homography relating two images of a plane is to first automatically find a large ($N \gg 4$) set of point correspondences. Due to measurement noise and presence of gross matching errors (outliers), the homography matrix is typically estimated using a *robust* method, like RANSAC [76]. For homography estimation, RANSAC, or RANdom SAmple Consensus, creates a homography hypothesis estimated from a randomly drawn *minimal sample* of the input data, *i.e.*, sample of minimal size needed to estimate an model, which is 4 correspondences in case of homography. Then the hypothesis is verified against all the available correspondences by counting how many of them are inliers to the homography hypothesis. The inlier / outlier decision can be based on thresholding the distance between the coordinates (u_2, v_2) and the homography-projected (u_1, v_1) (one-way transfer error), or other similar metrics.

This random minimal sample, hypothesis, and verification procedure is repeated many times and the best found model (most inliers) is kept. Finally, the best model is further refined by finding a least-squares fit to all its inliers.

There are many variations of this algorithm, *e.g.*, LO-RANSAC [77, 78], PROSAC [79], or MAGSAC++ [80]. While they improve on the plain RANSAC speed and/or accuracy, these methods are still stochastic and require unknown-in-advance number of iterations. In contrast, we designed a single-iteration homography estimation neural network suitable for planar tracking as we will describe in section 3.4.1.

3.3 RELATED WORK

General visual object tracking methods have been improving consistently, with deep-learning-based trackers dominating classical methods[61,

81]. In contrast, planar object trackers have only recently started using deep learning.

The homography trackers can be roughly divided into three main categories [8]: keypoint methods, direct methods, and deep methods. Traditional keypoint-based tracking consists of three steps: (i) keypoint detection and description using, *e.g.*, SIFT[40] or SURF[82], (ii) keypoint matching by nearest neighbor search in the descriptor space, and (iii) robust homography estimation with RANSAC [76]. The SOL [83] tracker uses SVM to learn keypoint descriptors and PROSAC [79] ordering. In GRACKER [84], the keypoints are not matched independently based only on descriptor similarity, but instead a graph-matching approach is used. The OBD [85] tracker uses ORB keypoints for target detection and optical flow tracking. In the POT-280 [7] benchmark, the authors compare several deep-learning based homography trackers. The best ones use the SIFT keypoint detector, a deep learning descriptor such as GIFT [86], MATCHNET [87], SOSNET [88], or LISRD [89], followed by RANSAC.

Direct methods formulate the tracking task as image registration. Given the current frame, they attempt to find a homography warping that optimizes the alignment of the current frame with the object in the initial frame. In the classical Lucas-Kanade [16] and the Inverse Compositional [90] methods, the warp quality is measured directly on the image intensities by a sum of squared differences. The ESM [91] tracker avoids the costly computation of Hessian in Lucas-Kanade by using an efficient second-order minimization (ESM) technique. GO-ESM [92] improved robustness to illumination changes by adding a gradient orientation feature on top of the image intensities and generalizing the ESM tracker to multidimensional features. The GOP-ESM [5] tracker extends GO-ESM with a feature pyramid and a coarse-to-fine iterative approach. Chen *et al*. [93] proposed to use the ESM algorithm as a differentiable layer in a siamese neural network architecture. The ESM layer iteratively aligns the template and the query frame feature maps obtained from a RESNET-18 [94] backbone pre-trained on IMAGENET. The whole architecture is then fine-tuned on image pairs synthesized from the MS-COCO dataset [95]. Direct methods perform well on the POIC[5] dataset, but typically fail on motion blur, partial occlusions and partially out-of-view targets, *e.g.* in the POT-210 [8] dataset.

Deep learning homography estimation is typically done by regression of four control points. The HOMOGRAPHYNET [96] and UDH [97] feed a concatenated pair of homography-related images through a CNN and formulate the homography estimation as direct regression of four control points. Rocco *et al*. [98] proposed to regress the four homography control points from a correlation cost-volume containing each-to-each similarities between Siamese VGG-16 [99] feature maps. The four-point regression is also used by the recently proposed

HDN [100] method, which decomposes the homography into a similarity transform and a homography residual. These control-point regression methods struggle with occlusions and often assume that the whole images are related by a homography, and do not distinguish between the target and the background motion. The PFNET [101] uses a custom convolutional architecture to estimate a dense flow field, which is then used in RANSAC, making the method not differentiable and end-to-end training not possible.

3.4 METHOD

We propose a weighted flow homography module (WFH) that assigns a flow weight $w_i \in [0, 1]$ to each OF correspondence and estimates a homography using a weighted least squares formulation (Sec. 3.4.1). The WFH is differentiable, making end-to-end training of both the WFH and the OF network possible. In Sec. 3.4.2, we propose a weighted optical flow tracker (WOFT) built around the WFH homography estimator.

3.4.1 *Weighted Flow Homography Module*

The idea of the WFH module is to compute a *flow weight* $w_i \in [0, 1]$ for each optical flow vector and to predict a homography by solving a weighted least squares (LSq) problem. The standard least squares homography fitting is sensitive to grossly incorrect correspondences (outliers). This is usually addressed by RANSAC, which uses repeated hypothesis sampling to find a homography and its outlier-free correspondence support set. The WFH instead eliminates outliers by setting their flow weights close to zero, allowing for a robust, single iteration, and fully differentiable weighted least squares fitting.

We process a pair of images with an optical flow estimation network, such as RAFT [22] to get OF correspondences $(\mathbf{p}_i, \mathbf{p'}_i)$, where $\mathbf{p}_i = (x_i, y_i)$ is a position in one image and $\mathbf{p}'_i = (x'_i, y'_i)$ the corresponding position in the second image. We then pass a suitable inner representation of the OF network to a weight-estimation CNN that predicts the flow weight w_i for each OF vector. Finally, we estimate homography by solving a system of equations by weighted least squares. First, we introduce plain least squares homography estimation, then we describe the weighted variant and the training loss function. Finally, we describe the weight estimation CNN in detail.

3.4.1.1 *LSq Homography*

Given the optical flow correspondences, we want to find a homography matrix $\mathbf{H} \in \mathbb{R}^{3\times3}$ mapping $(x_i, y_i, 1)$ to $(\lambda x'_i, \lambda y'_i, \lambda)$, $\lambda \neq 0$. This leads to an overdetermined homogeneous system of equations $\mathbf{Ah} = \mathbf{0}$, with

Figure 5: High weights of optic flow (*yellow*) appear mainly on corners and well-textured areas. *Bottom*: the POT-210 target with the highest average flow weights (*left*); weight values drop (*purple*) when "occluded" by a reflection (*right*). Best viewed in color.

$\mathbf{h} \in \mathbb{R}^{9\times 1}$ being the flattened $\mathbf{H}$-matrix and $\mathbf{A} \in \mathbb{R}^{2N\times 9}$ encoding the data constraints. The system is usually solved in the least-norm sense via a singular value decomposition (SVD) of $\mathbf{A}$. We use the PyTorch machine learning framework which includes differentiable SVD, but the back-propagated gradients are often unstable. To overcome this issue, we constrain the homography by fixing its bottom-right element $h_{3,3} = 1$, leading to a non-homogeneous system $\tilde{\mathbf{A}}\tilde{\mathbf{h}} = \mathbf{b}$, which can be solved in the least-squares sense using the QR decomposition with more stable gradients. Not all homographies are representable with this constraint (see Sec. 4.1.2 in [102]), but we have not encountered such a case in the tracking scenario.

In the non-homogeneous formulation, each correspondence adds two equations into $\tilde{\mathbf{A}} \in \mathbb{R}^{2N\times 8}$ and $\mathbf{b} \in \mathbb{R}^{2N}$:

$$\begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i \end{bmatrix} \tilde{\mathbf{h}} = \begin{bmatrix} -y'_i \\ x'_i \end{bmatrix} \tag{7}$$

We solve the least squares problem

$$\min_{\tilde{\mathbf{h}}} \sum_{j=1}^{2N} \|\tilde{\mathbf{A}}_{j,\cdot}\tilde{\mathbf{h}} - \mathbf{b}_j\|_2^2 \tag{8}$$

by QR decomposition of the data matrix $\tilde{\mathbf{A}} = \mathbf{QR}$ followed by solving the triangular system $\mathbf{R}\tilde{\mathbf{h}} = \mathbf{Q}^\intercal\mathbf{b}$ (triangular system solver available in PyTorch).

3.4.1.2 *Weighted LSq Homography*

In the proposed weighted least squares formulation, we weight each pair of equations with the corresponding estimated flow weight w_i and find

$$\min_{\tilde{\mathbf{h}}} \sum_{j=1}^{2N} w_j \|\tilde{\mathbf{A}}_{j,\cdot}\tilde{\mathbf{h}} - \mathbf{b}_j\|_2^2 \tag{9}$$

$$= \min_{\tilde{\mathbf{h}}} \sum_{j=1}^{2N} \| \left(\sqrt{w_j}\tilde{\mathbf{A}}\right)_{j,\cdot} \tilde{\mathbf{h}} - \left(\sqrt{w_j}\mathbf{b}_j\right) \|_2^2 \tag{10}$$

The weighted problem (9) is transformed into non-weighted (Eq. (8)) by multiplying each row of $\tilde{\mathbf{A}}$ and each element of $\mathbf{b}$ by the square root of the corresponding weight $\sqrt{w_i}$.

3.4.1.3 *Training WFH*

We train the WFH weight estimation CNN using a loss function on the predicted homography. We warp points forward by the ground truth homography $\mathbf{H}_{GT}$ then backward by the inverse of the estimated $\mathbf{H}$ and finally compute L1 loss on the projection errors as:

$$L(\mathbf{H}) = \frac{1}{N} \sum_{i=1}^{N} \|p_i - \mathbf{H}^{-1}\mathbf{H}_{GT}p_i\|_2 \tag{11}$$

Both the optical flow network and the flow weight estimation CNN are trained using a single loss function, and we do not use additional direct supervision of the flow weight predictor. The learned flow weights resemble a keypoint detector output (corners, well-textured patches), but with information from both images, therefore giving low weights on occlusions or significant appearance changes as shown in figure 5.

3.4.1.4 *Weight Estimation CNN*

The proposed WFH module operates on the correlation cost-volume pyramid of the RAFT [22] optical flow estimator, but the idea is applicable to other OF networks (Sec. 3.5.2). RAFT computes a correlation volume $\mathbf{C}^1 \in \mathbb{R}^{H/8 \times W/8 \times H/8 \times W/8}$ that captures the similarity between all pairs of feature vectors extracted from the two input images. Next, they construct a 4-layer correlation pyramid $\{\mathbf{C}^1, \mathbf{C}^2, \mathbf{C}^3, \mathbf{C}^4\}$. Finally, 9×9 patches centered on current flow vector estimates are sampled from this pyramid and processed by a neural network that produces a flow vector update. This is repeated several times to produce the final optical flow field.

In WFH we sample the correlation pyramid once more on the final OF positions, resulting in a $9 \times 9 \times 4$ feature map for each flow vector in the spatial resolution of $H/8 \times W/8$. To capture the global

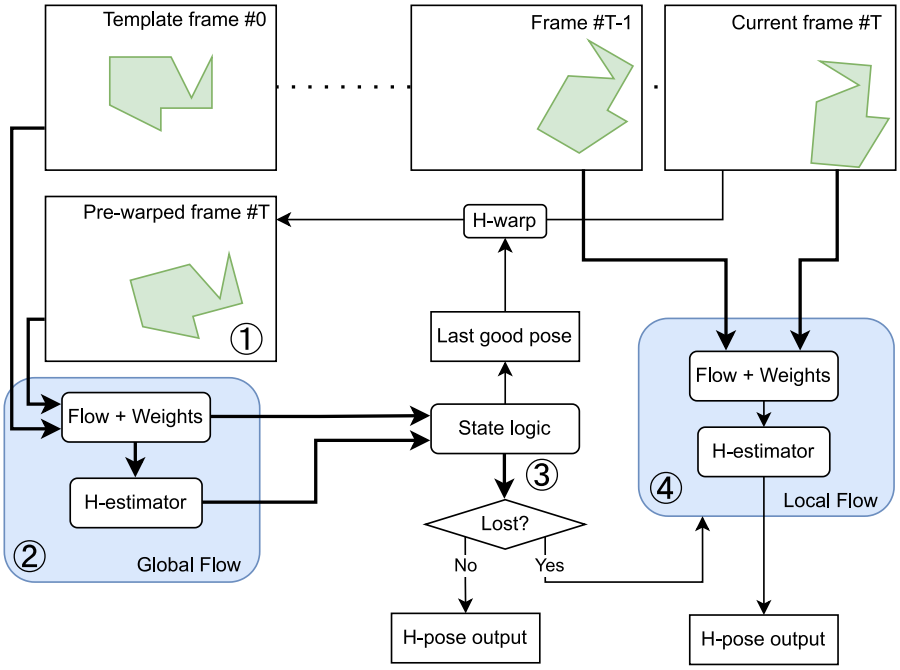


Figure 6: The WOFT tracker pre-warps the current frame using the last good pose (1). It then estimates a homography between the template and the pre-warped frame (2) and the reliability of the estimated homography is assessed (3). When the estimate is not reliable ('lost' state) a homography based on a local flow (4) is returned instead.

context, we then append an additional channel containing the mean correlation volume response $M(i,j) = \sum_{m=1}^{H/8} \sum_{n=1}^{W/8} \mathbf{C}^1_{i,j,k,l}$ for the given position $(i,j) \in \{1,\dots,H/8\} \times \{1,\dots,W/8\}$ in the first input image feature map. We process the resulting features $f_{i,j} \in \mathbb{R}^{9\times9\times5}$ with a three-layer convolutional network (kernel size 3, 128 output channels, ReLU) followed by a 1×1 convolution (single output channel) and global average pooling. Finally, we up-sample the results with the RAFT up-sampling module and apply a sigmoid activation to get a $H \times W$ score map with weights between 0 and 1.

3.4.2 *Homography tracker*

We propose a planar object tracker based on the weighted flow homography module, WFH. Our weighted optical flow tracker, denoted WOFT, consists of four main parts as shown in Fig. 6.

First, we apply a pre-warping technique to reduce large pose differences, which are not handled well by OF methods. The current video frame I_t is pre-warped ① by the homography from the last reliable frame I_G, with $G = 0$ initially. The pre-warp $\tilde{I}_t = \mathcal{W}(\mathbf{H}^{-1}_{0\to G}, I_t)$ ① reduces the pose difference between the template and the current images, resulting in a motion similar to the typical $I_{(t-1)} \to I_t$ optical

flow scenario. The possible appearance difference between the template and a temporarily distant frame (caused mainly by illumination changes and motion blur) is implicitly handled by the optical flow feature encoder.

Second, we compute the *global* optical flow ② between the template frame I_0 and the pre-warped current frame $\tilde{I}_t$ and the corresponding flow weights. We mask the flow correspondences, only leaving the ones starting inside the template mask and ending inside the current image. To speed up the homography estimation, we randomly subsample the correspondences, only keeping 500. We then estimate homography $\mathbf{H}_{0\to\tilde{t}}$ using weighted least squares as described in Sec. 3.4.1. Computing the homography between the template and the pre-warped current frame prevents error accumulation and target drift (Sec. 3.5.2).

We pass the weighted optical flow together with the computed homography to a state logic block ③ that decides whether the tracking was successful or not. The lost/not-lost decision is made based on the support set size of the estimated homography. In particular, with optical flow correspondences $(\mathbf{p}_i, \mathbf{p}'_i)$ we warp each position $\mathbf{p}_i = (x_i, y_i)$ using the homography $\mathbf{H}_{0\to\tilde{t}}$ and compute the Euclidean distance to the position $\mathbf{p}'_i = (x'_i, y'_i)$. The i-th correspondence is an inlier when $\|\mathcal{W}(\mathbf{H}_{0\to\tilde{t}}, \mathbf{p}_i) - \mathbf{p}'_i\| \leq 5$ pixels – a standard threshold on planar tracking benchmarks [8, 5]. We declare the tracker lost when it has a small support set, *i.e.* less than 20% inliers.

When the tracker is not lost, we return $\mathbf{H}_{0\to t} = \mathbf{H}^{-1}_{0\to G}\mathbf{H}_{0\to\tilde{t}}$ and update the last good frame index used for pre-warping $G = t$. When the tracker is lost, we make a second attempt to estimate the pose using a *local* optical flow $I_{(t-1)} \to I_t$. The local flow tends to drift, but it helps to keep track of the target pose in the short term. The temporarily close input images are close in appearance (similar illumination, similar motion blur, *etc.*). We estimate $\mathbf{H}_{(t-1)\to t}$ ④ by weighted least squares as described above and output $\mathbf{H}_{0\to t} = \mathbf{H}_{(t-1)\to t}\mathbf{H}_{0\to(t-1)}$. Moreover, when the tracker is lost for more than 10 frames, we reset the pre-warping last good frame index $G = 0$. The target pose can change significantly over the 10 frames, making the pre-warp information outdated. Moreover, a bad pre-warp homography can ruin any chance of recovering, *e.g.* an outdated strong perspective change pre-warp distorts the current target area beyond being recognizable, and the identity homography with $G = 0$ is the safest choice.

3.4.3 *Implementation details*

For optical flow, we use the author-provided RAFT checkpoint trained on Sintel. We then train the weight estimation CNN for 10 epochs on a synthetic dataset with 50000 image pairs. We generate the training set by repeatedly sampling a random MS COCO[95] image and warping

it with two random homographies representing the template and the current frame pose. The random homographies are generated by perturbing each corner of the image with a random vector of length up to 20% of the image diagonal. We blur the second warped image by a random linear motion of length up to 20 pixels. Finally, both images are passed through JPEG compression with quality set to 25.

We train with AdamW [103] optimizer with an initial learning rate $1e^{-3}$, which is then halved after every epoch. Finally, we fine-tune the whole network, including RAFT for 2 epochs, starting from the learning rate $1e^{-5}$ and again halving it after every epoch. To stabilize the training procedure, we discard training samples achieving loss over 100.

The tracker runs at around 3.5 FPS on a GeForce RTX 2080 Ti GPU (i7-8700K CPU @ 3.70GHz). The majority of time is spent on the optical flow computation (275ms). The weight computation (2ms), the weight up-sampling (1ms), and the least squares homography estimation (5ms) take negligible time. Image pre-warping (done on CPU), optical flow masking, and subsampling cost an additional 7ms.

A faster variant $\text{WOFT}_{\downarrow s}$ downscales the input images to $H/s \times W/s$ and rescales the output homographies to the original resolution.

3.5 EXPERIMENTS

We evaluate the proposed tracker on two standard planar object tracking datasets, POT-210 and POIC and show that it consistently achieves high accuracy and robustness.

POT-210[8]: The Planar Object Tracking in the Wild benchmark contains 210 videos of 30 objects. Each object appears in 7 video sequences with different challenging attributes – *scale* change, in-plane *rotation*, *perspective* distortion, motion *blur*, *occlusion*, *out-of-view*, and *unconstrained*. The sequences have a fixed length of 501 frames. POT-280 [7] extends POT-210 by 10 new objects.

POIC[5]: the Planar Objects with Illumination Changes dataset consists of 20 sequences of varying length giving a total of 22971 frames. The dataset contains sequences with translation, in- and out-of-plane rotations, and scale changes, but mainly focuses on strong specular highlights and other significant illumination changes, making it complementary to POT-210.

PlanarTrack[6]: the PlanarTrack is a recent large scale benchmark. It has 7× more annotated frames than POT-280 and 25× more targets. Each target is tracked only in one video in a setting of the *unconstrained* sequences from POT benchmarks. The dataset also contains some unconventional targets, like a transparent glass plate or a TV screen playing a video.

Evaluation protocol: On both all the used datasets a tracker is initialized on the first frame and left to track till the end of the sequence.

Figure 7: Precise re-annotation examples. Original ground truth annotation (*left*), improved ground truth annotation (*right*). The grayscale template in green channel, the GT-warped current frame in red and blue channels. Imprecise annotation causes green and magenta shadows, while precisely aligned images produce a grayscale result. The green bands on top and on right side respectively are caused by a partial occlusion on current frame. The alignment error of the original GT evaluated on the improved ground truth is 15.8px (*top*) and 7.2px (*bottom*).

Figure 8: Additional POT-210 [8] re-annotation examples. Left: original GT annotation, right: our precise re-annotation. The grayscale template in green channel, the GT-warped current frame in red and blue channels. Imprecise annotation causes green and magenta shadows around contours, while precisely aligned images produce a grayscale result. In some cases, there are still green and magenta visible in the well-aligned images - these are due to reflections and change of illumination.

Figure 9: The *homography discrepancy* is not a good metric. Left: ground truth annotation (green box) on frame 242 of sequence V18_3 in the POT dataset [8, 7]. Middle: shifting the top-left corner 155 pixels up (red box) results in homography discrepancy score of 918. Right: shifting the top-left corner just 0.5 pixels right (red box) results in homography discrepancy score of 2929, *i.e.* a sub-pixel error in the position of a single corner results in big discrepancy, much bigger than the official benchmark threshold $t_s = 10$. An over $300\times$ bigger error in corner position results in around $3\times$ smaller homography discrepancy in this case.

The *alignment error* e_{AL} is computed for each annotated frame. Given four reference points $\mathbf{x}_i \in X$ in the first frame, the alignment error is defined as root-mean-square error between their projection into the current frame by the ground truth homography $\mathbf{H}^*$ and by the tracker homography $\mathbf{H}$,

$$e_{AL}(\mathbf{H}; \mathbf{H}^*, X) = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(\mathcal{W}(\mathbf{H}^*, \mathbf{x}_i) - \mathcal{W}(\mathbf{H}, \mathbf{x}_i))^2}, \tag{12}$$

with $\mathcal{W}(\mathbf{H}, \mathbf{x})$ representing the projection of vector $\mathbf{x}$ by a homography $\mathbf{H}$. Tracker precision is measured as a fraction of frames with $e_{AL} \leq 5$ px (P@5 score). Additionally, we measure $e_{AL} \leq 15$ px (P@15 score), corresponding to the fraction of frames with target not tracked perfectly, but not completely lost either – we call this robustness regime. On average, $e_{AL} = 15$ px corresponds to IoU (Intersection over Union) score of 0.89, much stricter threshold than commonly used on bounding boxes or segmentations.

Apart from the alignment error, POT [8, 7] and PlanarTrack [6] also use a *homography discrepancy* score. The homography discrepancy measures the reprojection error of the corners of a two-pixel wide square at the origin. Depending on the pose of the target, even sub-pixel errors in predicted corner positions can result in arbitrarily high homography discrepancies as shown in Fig. 9. We do not use this error metric, because it heavily depends on the tracked object position in the image and does not have a meaningful and useful interpretation.

3.5.1 *Ground truth quality*

During the analysis of WOFT performance on POT-210, we found that in many cases the ground truth (GT) annotations are less accurate than the official 5px error threshold. We have performed reannotation of a subset of the POT-210 dataset to measure the original GT quality and provide more accurate estimates of tracker performance, see Fig. 7. Our annotation tool shows the template, the object on the current frame warped with the current annotation, and, most importantly, an alignment visualization. We convert both the template frame and the current frame to grayscale and overlay the warped frame over the template, putting the template into the green channel and the current frame into the red and the blue channels. This allows for very precise alignment over the whole extent of the target, unlike the annotation interface used for the original annotation (Fig. 4 in [8]). We have fully manually reannotated frames 82, 172, 252, 332, and 412 from each sequence, without seeing the WOFT estimated poses. On some frames a precise homography alignment was not possible – either due to strong motion blur, or due to imperfect planarity of the targets. A target non-planarity, *e.g.* a slight bend in otherwise flat-looking target, manifests itself the most when the target is viewed from extreme angles. We annotate such cases as precisely as possible (selected examples shown in Fig. 10) and mark the frames as problematic and ignored in evaluation. The new GT is publicly available at https://cmp.felk.cvut.cz/~serycjon/WOFT. More examples of the reannotation overlay are in Fig. 8. The alignment error of the original GT evaluated on our re-annotation is 3.63 on average, and worse than the official 5px threshold in 15% cases.

3.5.2 *Ablation study*

In Table 1, we show the impact of various design choices of WOFT on POT-210 performance (both on the original and the more accurate re-annotated ground truth). First, we show the importance of computing the optical flow between the template and the pre-warped current frame. In rows 1, 2 we only use the local flow (from $I_{(t-1)}$ to I_t). The tracker drifts and quickly loses the target, resulting in overall poor performance. A big performance improvement is achieved by using global flow (from I_0 to $\tilde{I}_t$) and always using the previous frame for pre-warping (rows 3, 4). Another boost in performance is achieved with the controlled pre-warping (rows 5 - 9), where the local flow is used when the global flow fails and the pre-warp homography is reset when the target is 'lost' for more than 10 frames.

Using the weighted least squares homography estimation consistently improves the performance – compare row 2 to row 1 (P@5 +1.3), row 4 to row 3 (P@5 +10.7), and row 6 to row 5 (P@5 +8.3). In row 7,

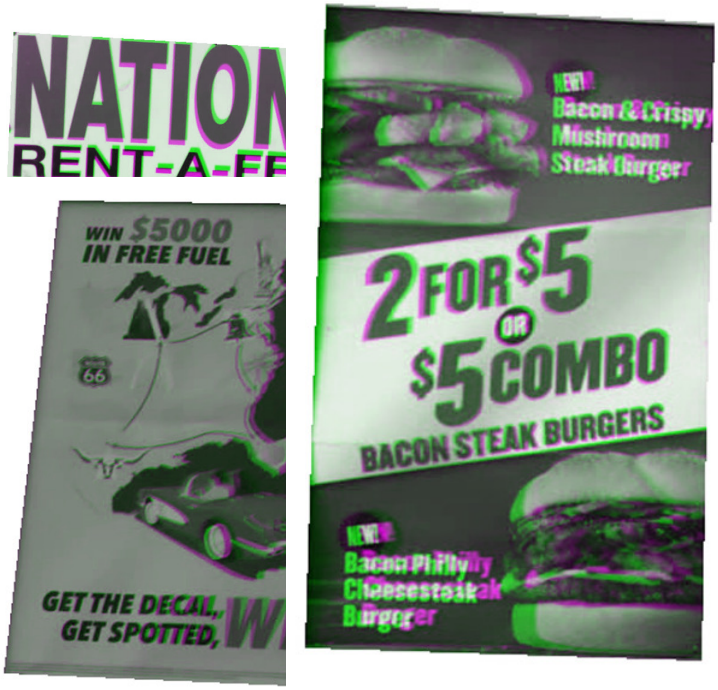


Figure 10: Selected not completely planar targets from POT-210 [8]. When viewed from extreme angle, slight target non-planarity becomes visible. It is then impossible to precisely align the image with the template view on the whole target surface. Top-left image: precise alignment on sides, imprecise alignment in the center. Bottom-left image: imprecise alignment in the center and bottom-right part. Right image: precise alignment in the center, imprecise elsewhere.

	M	PW	H	W	F	P@5 orig	P@5 rean	P@15 orig	P@15 rean
(1)	R	–	LSq	–	✓	5.7	0.8	16.6	10.7
(2)	R	–	LSq	✓	✓	7.0	2.1	22.5	17.3
(3)	R	✓	LSq	–	✓	57.6	63.6	68.1	68.9
(4)	R	✓	LSq	✓	✓	66.7	74.3	75.5	76.4
(5)	R	C	LSq	–	✓	73.1	82.1	89.9	92.0
(6)	**R**	**C**	**LSq**	**✓**	**✓**	**80.6**	**90.4**	**93.9**	**95.6**
(7)	R	C	LSq	✓	–	75.1	83.0	87.3	87.8
(8)	R	C	IRLSq	✓	✓	80.6	90.4	93.9	95.6
(9)	R	C	RSAC	–	✓	79.5	88.8	92.7	93.5
(10)	L	C	LSq	–	–	66.9	74.8	82.3	82.6
(11)	L	C	RSAC	–	–	72.8	80.9	84.4	85.1
(12)	L	C	LSq	✓	–	72.8	81.0	86.1	87.1

Table 1: Ablation study on POT-210, evaluated on the original ground truth (*orig*) and the reannotation (*rean*). In all experiments, weighted least squares perform better than non-weighted alternative in both P@5 and P@15. M – flow method: RAFT (R), LiteFlowNet2 (L). PW – use of the global pre-warped flow: never (–), always (✓), controlled (C). H – homography estimation method: least squares (LSq), iterative re-weighted least squares with Huber loss (IRLSq), RANSAC (RSAC). W – using the estimated weights. F – using the fine-tuned RAFT flow.

we used the same settings as in WOFT (row 6), but without the RAFT fine-tuning, resulting in a drop in P@5 (-7.4). We have also experimented (row 8) with estimating homography by weighted iterative reweighted least squares (IRLSq) instead of ordinary weighted least squares. We have set the IRLSq to optimize the Huber loss (also called smooth L1 loss) which is more robust to outliers than least squares. This did not change the performance (w.r.t. row 6), indicating that our estimated weights already take care of outliers and the robust estimator is not necessary. Next, we compare RANSAC (row 9) with the proposed WOFT (row 6). The weighted least squares approach achieves better results (P@5 $+0.9$) in a single differentiable pass.

Rows 10-12 show WOFT with LiteFlowNet2 [104] flow instead of RAFT. Again, the weighted LSq estimator (row 12) works better than plain LSq (row 10) or RANSAC (row 11). For the LiteFlowNet2 experiment, we have kept the same 3-layer CNN architecture for weight estimation as with RAFT (Sec. 3.4.1.4). For inputs, we have used the cost-volume on the last LiteFlowNet2 NetE pyramid level (level 3). The cost-volume contains a 7×7 correlation response map for each position in the template feature map. We feed each of these 7×7 maps through the weight estimation CNN to get the corresponding flow weights w_i. The weight estimator training was kept the same, except we have only trained for 5 epochs. We did not fine-tune the LiteFlowNet2 and used the liteflownet2_ft_4x1_600k_sintel_kitti_320x768 configuration and checkpoint from MMFlow [105].

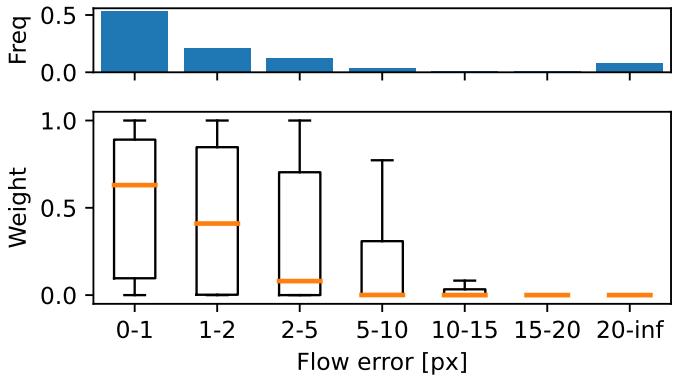


Figure 11: Weight distribution for different optical flow error ranges measured against the re-annotated POT-210 [8] ground truth. Median in orange. Top: frequency of each flow error range. The weight network learned to assign zero weight to incorrect flow vectors (outliers) and high weight to some correct flow vectors.

3.5.3 *Weights Evaluation*

Figure 11 shows how the learned weights correlate with the optical flow quality. Low-textured areas and ambiguous features are often assigned a low weight (Fig. 5) even when the corresponding optical flow is correct. Importantly, the incorrect flow vectors are assigned low weights.

3.5.4 *POT-210 and POT-280 evaluation*

We compare WOFT method against the best performing methods on the POT-210 [8] dataset. Namely keypoint methods: SIFT [40], OBD [85], and GRACKER [84], deep control point regression HDN [100], the deep learning based methods evaluated in [7]: SOSNET [88], SUPERGLUE [106], LISRD [89], the direct methods: GOP-ESM [5], and SIAM-ESM [93] (deep + direct).

The proposed WOFT achieves state-of-the-art on the POT-210 dataset. The Alignment Error e_{AL} results are depicted on Fig. 12 and in Tab. 2. Evaluated over all 210 sequences (*all* plot) The WOFT tracker performs better than all the other methods, both in terms of accuracy (P@5), and robustness (P@15). Note that more than half of the 5px threshold errors of WOFT are explained by imprecise GT.

WOFT also achieves top results on the extension of POT-210 dataset, POT-280 [7]. With 76.9 P@5 and 93.2 P@15 it outperforms state-of-the-art methods by a large margin as shown in Tab. 3.

SPEED-ACCURACY TRADE-OFF The WOFT method runs at 3.5 FPS due to slow RAFT OF computation (275ms per frame). A simple method to gain speed is to compute optical flow on lower-resolution images. We have evaluated WOFT variants $\text{WOFT}_{\downarrow s}$, $s \in \{2, 3, 4\}$ which downsamples the input images to $H/s \times W/s$ resolution before computing the optical flow and re-scales the output homography back into the original resolution. The speed-accuracy trade-off is shown in Fig. 14. The $\text{WOFT}_{\downarrow 3}$ variant operating on $H/3 \times W/3$ images runs close to real-time and achieves state-of-the-art.

method	year	FPS	P@5 orig	P@5 rean	P@15 orig	P@15 rean
GOP-ESM [5]	2019	4.95*	42.9	–	49.7	–
SuperGlue [106, 7]	2020	3.7*	39.1	42.1	58.0	55.7
Gracker [84]	2017	4.8*	39.2	–	63.2	–
SiamESM [93]	2019	–	58.7	–	66.2	–
SOSNet [88, 7]	2019	1.5*	56.6	60.9	69.9	67.0
SIFT [40, 7]	2004	0.8*	62.2	65.8	71.3	69.6
OBD [85]	2021	30*	48.4	54.3	79.3	79.2
LISRD [89, 7]	2020	7*	61.6	68.3	79.6	79.2
HDN [100]	2022	10.6*	61.3	70.9	91.5	92.4
$\text{WOFT}_{\downarrow 3}$ (ours)		19.2	68.9	80.5	91.2	92.3
WOFT (ours)		3.5	**80.6**	**90.4**	**93.9**	**95.6**

Table 2: Results on POT-210 [8] dataset. The proposed WOFT tracker sets a new state-of-the-art performance in both accuracy (P@5) and robustness (P@15). Evaluated on the original ground truth (*orig*) and the re-annotation (*rean*). Tracking speed in frames per second (FPS). * speeds from the papers, different hardware.

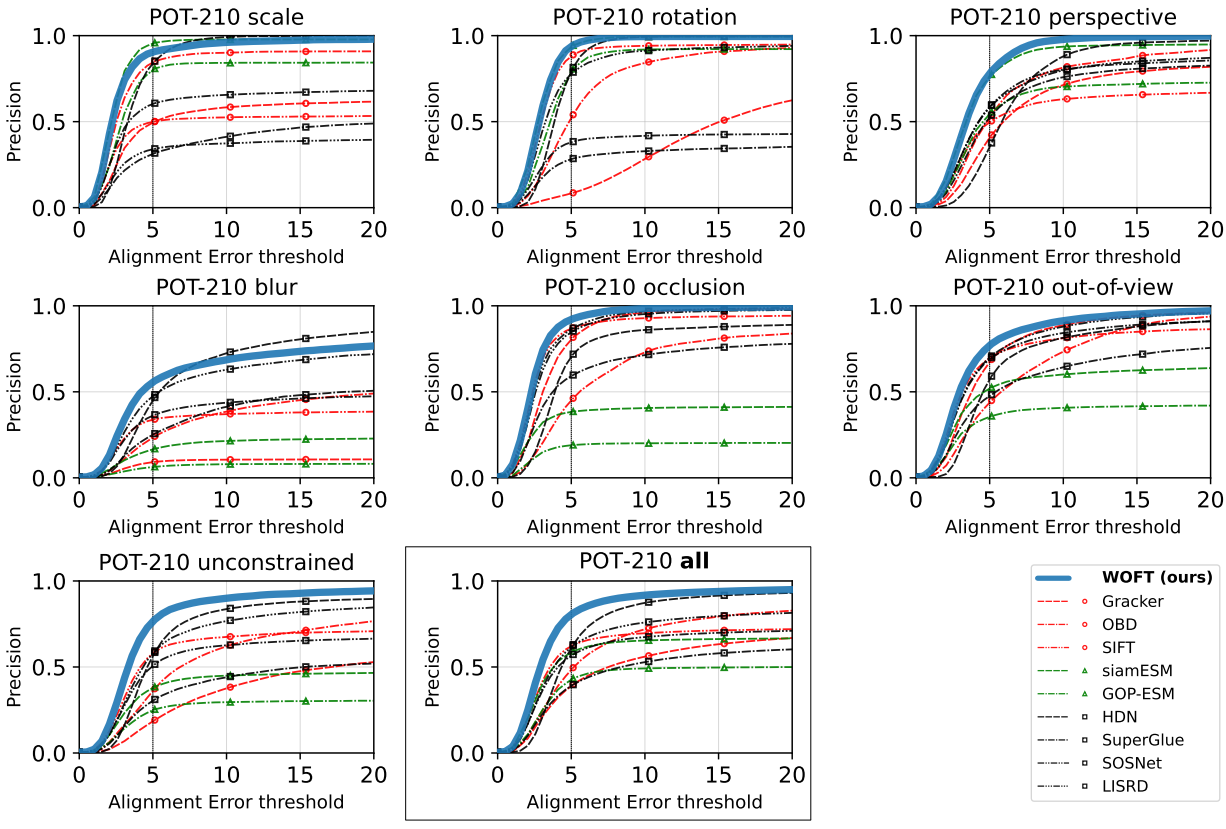


Figure 12: Alignment Error on POT-210 [8] (original GT). WOFT performs well on all sequence types, reducing the error on the official 5px threshold to half of the best competitor. Method types: (*red circle*) – keypoint, (*green triangle*) – direct, (*black square*) – deep.

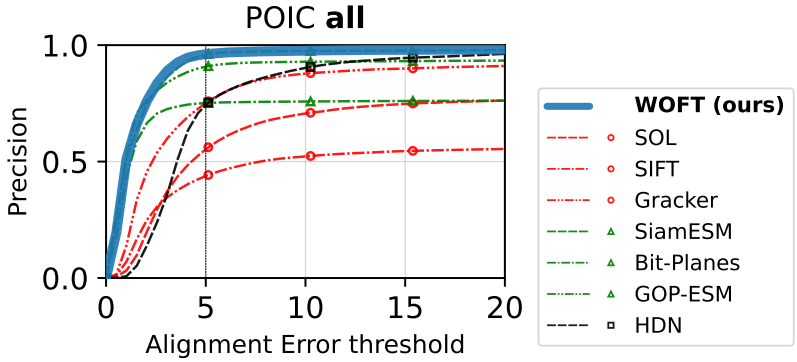


Figure 13: Alignment Error evaluation on POIC [5]. The proposed WOFT achieves state-of-the-art with 96.1 P@5 and 98.0 P@15.

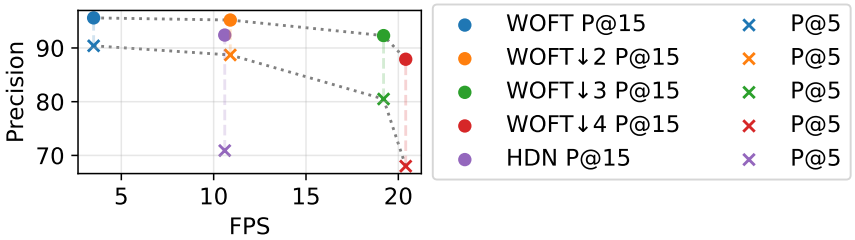


Figure 14: Speed-accuracy trade-off of $\text{WOFT}_{\downarrow s}$ variants as measured on the re-annotated POT-210 dataset. Down-scaling the input images with $s = 2$ or $s = 3$ significantly speeds up ($3\times$, respectively $6\times$) the WOFT tracker while retaining state-of-the-art accuracy. The second-best performing method on POT-210 – HDN [100] in purple.

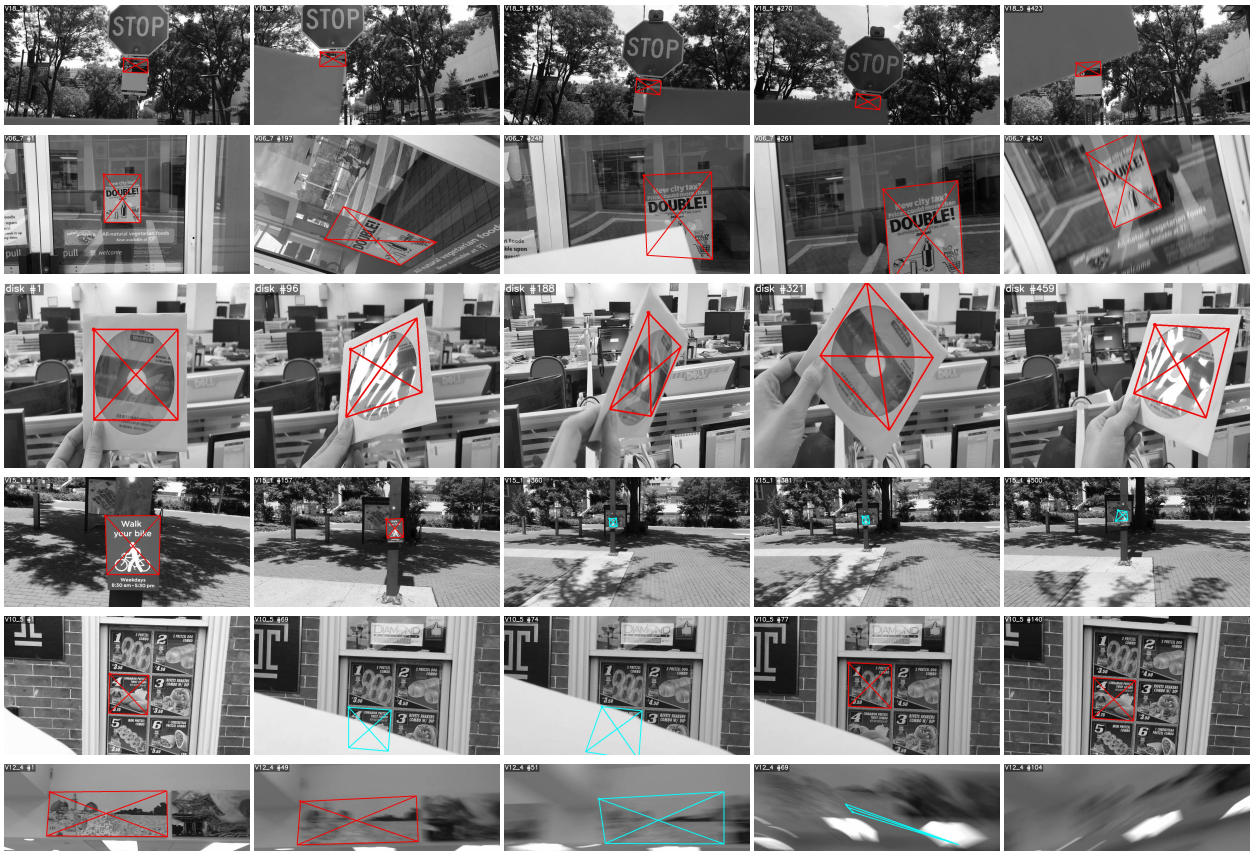


Figure 15: WOFT tracking. State visualization - red: tracking, cyan: lost - switch to local flow. First row: WOFT handles strong occlusions on the POT-210 V18_5 sequence. Second row: successful tracking on the V06_7 POT-210 *unconstrained* with perspective change, partial occlusion, scale change and motion blur. Third row: successful tracking in a POIC disk sequence, where a large part of the target surface changes appearance because of specular reflection. The last two rows show selected tracking failures. Row 4: the tracker is 'lost' and did not recover because of a big scale difference w.r.t. the template frame, however, the local homography estimation prevents complete failure. Row 5: the target becomes almost fully occluded and the tracker switches to track a nearby distractor patch. Later WOFT reacquired the correct target. Last row: WOFT can handle a moderate amount of motion blur, but fails on extremely blurred frames.

POST-PROCESSING INDUSTRY STATE-OF-THE-ART We have additionally evaluated a film post-processing industry standard planar tracking solution Mocha Pro 2022. The software is primarily made for interactive use, but also provides python API enabling fair benchmark evaluation. We have tested three variants of the Mocha Pro tracker hyper-parameters on POT-210. We used perspective (homography) model in all the experiments. We have tried I. the default parameters, II. increasing the *Min % Pixels Used* parameter to 100%, and III. increasing the target initialization by 10%. We have chosen the variants II. and III. according to the recommendations in Mocha Pro user guide. Variant III. performs best, but still significantly worse (P@5 32.8, P@15 52.0) than POT-210 state-of-the-art.

We have observed the Mocha Pro tracker to work well for a short time but completely fail afterwards. The temporal smoothness of the tracking may be more important than pixel-perfect precision as human are very sensitive to visual jitter. A typical tracking workflow is letting the tracker work until it starts drifting, manually fix the tracking output on a problematic frame and resume the tracking. This human-in-the-loop approach ensures visually smooth output and gives the visual effect artist full control of the output. The user interfaces of tools like Mocha Pro or DaVinci Resolve are tailored to such interactive use. This could explain the poor results of the Mocha Pro tracker in the benchmark setting of "initialize once and let track" which it was not designed for. Also the POT-210 dataset probably does not represent the kind of videos that video post-processing tools are typically used for (high resolution, high quality videos without much rotation or blur caused by shaking camera).

3.5.5 *POIC evaluation*

We compare (Fig. 13) the WOFT tracker performance with the top methods evaluated on the POIC [5] dataset. Apart from the methods evaluated on POT-210, this includes SOL [83] and BIT-PLANES [107]. WOFT achieves state-of-the-art results with 96.1 P@5 and 98.0 P@15 as shown in Table 4. See Fig. 15 for WOFT output examples on both POT-210 and POIC.

3.5.6 *PlanarTrack evaluation*

On the PLANARTRACK benchmark [6], WOFT out-performs other tested trackers by a large margin. In particular, it achieves [6] P@5 score of 0.43, while the second-best HDN [100] has only 0.26. The big performance drop be-

method	year	P@5	P@15
SuperGlue [106, 7]	2020	37.7	58.2
SOSNet [88, 7]	2019	51.9	67.1
HDN [100]	2022	56.7	88.9
SIFT [40, 7]	2004	57.2	68.4
LISRD [89, 7]	2020	57.3	77.6
WOFT (ours)		**76.9**	**93.2**

Table 3: Results on POT-280 [7] dataset. The proposed WOFT tracker sets a new state-of-the-art performance in both accuracy (P@5) and robustness (P@15).

method	P@5	P@15
SIFT [40, 5]	43.8	54.5
SOL [83]	55.3	74.8
HDN [100]	74.4	94.5
Bit-Planes [107]	75.1	76.0
Gracker [84]	75.2	89.9
GOP-ESM [5]	90.8	93.1
SiamESM [93]	**96.1**	97.7
WOFT	**96.1**	**98.0**

Table 4: Results on POIC [5] dataset. The proposed WOFT tracker achieves state-of-the-art performance in both accuracy (P@5) and robustness (P@15).

tween POT-210 and PLANARTRACK shows that the planar tracking is still far from being solved on particularly challenging scenarios. We have briefly inspected the PLANARTRACK ground truth and found similar issues like in POT-210, however the current state-of-the-art methods still do not achieve performance so good that the ground truth quality would have significant effect in benchmarking.

3.6 DISCUSSION AND LIMITATIONS

The WOFT tracker handles partial occlusions, a moderate amount of motion blur, and the illumination changes and lack of texture present in the POIC dataset. In comparison, other methods performing well on POIC (SIAMESM [93], GOP-ESM [5]) have low performance on POT-210 and vice versa (LISRD [89], SIFT [40]). Moreover WOFT achieves state-of-the-art performance on the PLANARTRACK benchmark [6], which was published after the WOFT paper [9]. WOFT does not feature a re-detection scheme and estimates only the residual transformation after the pre-warp step. This causes issues when the tracker gets lost for more than 10 frames on the *scale* subset. After resetting the pre-warp source frame to $G = 0$ (pre-warp with an identity homography), the scale component of the residual transformation is sometimes bigger than what the flow network can handle (see Fig. 15).

We tested the proposed WFH homography method on the RAFT OF network, which is accurate (Fig. 11), but slow (275ms per frame). However, the OF estimation is an active area of research and we expect new accurate and fast methods to be published in the future. The core idea of WFH – flow weights computed from an OF cost-volume and a differentiable homography estimation with weighted LSq – is applicable to other OF methods. The ablation study results with RAFT replaced by LITEFLOWNET2 support this claim. We also proposed a simple $\text{WOFT}_{\downarrow 3}$ variant that operates fast (19.2 FPS) and still achieves state-of-the-art.

4

TRACKING ANY POINT

Figure 16: **MFT – Multi-Flow Tracker application: video editing.** A WOW! logo, inserted in frame 0 of sequences from selected standard datasets [1, 68], propagated by MFT. Frames at 0%, 50%, and 100% of the sequence shown. Full videos are available at http://cmp.felk.cvut.cz/~serycjon/MFT.

This chapter is about tracking of any points (TAP), not just points on a single planar target surface. Moreover we focus on tracking *densely*, *i.e.* every point from an initial frame. This can be viewed as an extension of optical flow for long-term tracking. Although dense feature matchers (see section 2.2) can be used to extract dense correspondences between temporarily distant frames of a video, they cannot use the information from the intermediate frames leading to a much harder and sometimes even impossible task (see section 4.5 for an example).

In planar tracking we had a geometric model which we used to pre-warp the video frames to make the task of matching distant frames simpler for the OF estimator, but here we have no such thing. The objects are no longer planar or even rigid and the goal is to track points on multiple objects and the background simultaneously. We thus do not attempt to model the geometry of the whole scene, but instead rely on the optical flow alone. We generalize the idea of estimating OF between pairs of temporarily distant frames that we have used in WOFT, *i.e.* computing the OF between the initial and the current frame.

This chapter presents trackers MFT, published in [10], and MFTIQ [12], which is currently under review. Both papers are a joint work with Michal Neoral, both with equal contribution from both of us. Michal was responsible for adapting and training the neural networks, while I implemented the tracker and the experiments. We contributed equally to writing of the papers, both the first drafts and the final forms, and to developing the algorithms and analyzing them.

4.1 INTRODUCTION

Reliable dense optical flow has a significant enabling potential for diverse computer vision applications, including structure-from-motion, video editing, and augmented reality. Despite the widespread use of optical flow between consecutive frames for motion estimation in videos, generating consistent and dense long-range motion trajectories has been under-explored and remains a challenging task.

A simple baseline method for obtaining point-to-point correspondences in a video, concatenates interpolated optical flow to form trajectories of a pixel, i.e. the set of projections of the pre-image of the pixel, for all frames in a sequence as shown in fig. 17. However, such approach suffers from several problems: error accumulation leading to drift, sensitivity to occlusion and non-robustness, since a single poorly estimated optical flow damages the long-term correspondences for future frames. This results in trajectories that quickly diverge and become inconsistent, particularly in complex scenes involving large motions, repetitive patterns and illumination changes. Additionally, concatenated optical flow between consecutive frames cannot recover trajectories after occlusions. Few optical flow approaches estimate occluded regions or uncertainty of estimated optical flow.

Another baseline approach (also shown in fig. 17) — matching every frame with the reference — is neither prone to drift nor occlusions, but has other weaknesses. As the pose and illumination conditions change in the sequence, the matching problem becomes progressively more difficult. In the datasets used for point-tracking evaluation, match-to-reference performs worse than consecutive frame optical flow concatenation.

Addressing both weaknesses, we proposed a novel method for dense long-term pixel-level tracking. It is based on calculating flow not only for consecutive frames, but also for pairs of frames with logarithmically spaced time differences (see Fig. 18). We show that when equipped with suitable estimates of accuracy and of being occluded, a simple strategy for selecting the most reliable concatenation of the set of flows leads to dense and accurate long-term flow trajectories. It is insensitive to medium-length occlusions and, helped by estimating the flow with respect to more distant frames, its drift is reduced.

The idea to obtain long-term correspondences by calculating a set of optical flows, rather than just flow between consecutive images, appeared for the first time in [108]. This led to a sequence of papers on the topic [109, 110, 111]. The performance of these early, pre-conv-net methods is difficult to assess. They were mainly qualitatively, i.e. visually, tested on a few videos that are not available.

The MFT paper [10] introduced the following contributions: A point-tracking method that is (i) capable of tracking all pixels in a video based on CNN optical flow estimation, (ii) conceptually simple and can be trained

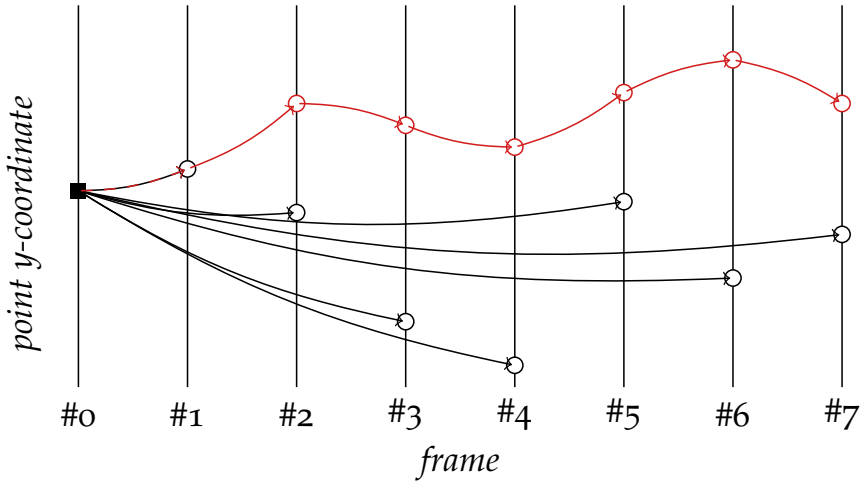


Figure 17: **Baseline approaches to long-term tracking with optical flow.** Chaining optical flows estimated between consecutive frames *(top, red)* and directly estimating optical flow between the reference and the current frame *(bottom, black)*.

and evaluated on a single customer grade GPU. We show (iii) a simple yet effective strategy for selection of long-term optical flow chain candidates, and (iv) how to select the most reliable candidate on the basis of spatial accuracy and occlusion probability obtained by small CNNs trained on synthetic data. We publish the results and the method code [1].

Experimentally the method outperforms baselines by a large margin and provide a good speed/performance balance, running orders of magnitude faster than the state-of-the-art for video point tracking [112, 113] when used for dense point tracking. Fig. 16 shows an application of the proposed method for video editing.

The MFTIQ [12] builds on the ideas of MFT and has the following extra contributions. We have developed (v) an Independent Quality (IQ) module which decouples the occlusion and uncertainty estimation used in MFT from the optical flow computation, which leads to better performance. Thanks to the flow quality estimation being independent on the particular optical flow network, we can integrate (vi) any off-the-shelf OF method with MFTIQ in a plug-and-play manner and without any re-training or fine-tuning. We show (vii) that MFTIQ out-performs MFT and other point trackers, getting near the state-of-the-art, while still being significantly faster for dense tracking.

4.2 RELATED WORK

LONG-TERM OPTICAL FLOW To track points over multiple consecutive frames, some methods [114, 115, 116] have proposed to concatenate estimated optical flow. However, they cannot recover from partial occlusions. Moreover, concatenating optical flow results in error accumulation over time and induce drift in the tracked points. Standard OF benchmarks [33, 37] do not evaluate occlusion predictions and consequently most OF methods do not detect occlusions at all. Although some optical flow methods have been proposed

[1] https://github.com/serycjon/MFT

to estimate the flow from more than two frames [117, 23, 17, 18], they still operate in a frame-by-frame manner and do not handle partial occlusions well.

Another line of work [108, 118, 109] addresses these limitations. These algorithms construct long-term dense point tracks by merging optical flow estimates computed over *varying time steps*, not just on neighboring frames. This enables handling of temporarily occluded points by skipping over the occlusions and establishing the correspondence between frames where the points in question are not occluded. However, these methods rely on the brightness constancy assumption, which leads to failure over distant frames. In subsequent works [110, 111], this approach was extended by statistical flow selection. The idea is to generate a large number of motion path candidates by randomly selecting reference frames and weighting them based on estimated quality. The optimal candidate path is then determined through global spatial-smoothness optimization. However, these methods are computationally intensive and limited to tracking a small patch of a single object.

In comparison, our proposed MFT generates only a small number of candidates and picks the best one based on occlusion and uncertainty estimated by a simple CNN. Although some optical flow methods estimate occlusions [119, 23, 26, 120, 121, 122] or uncertainty of estimated optical flow [123, 124, 125], state-of-the-art optical flow methods [22, 27] do not provide such estimates. We are the first to employ estimation of occlusion and optical flow uncertainty for the dense and robust long-term tracking of points.

POINT TRACKING aims to track a set of physical points in a video as introduced in TAP-Vid [126]. A baseline method TAP-Net [126] computes cost volume (similar to RAFT [22]) for a single query point independently for each frame of the sequence. A two-branch network then estimates the position and visibility of the query point in the targeted frame. PIPs [127] focuses on tracking points through occlusions by processing the video in fixed-sized temporal windows. It does not re-detect the target after longer occlusions. PIPs use test-time linking of estimated trajectories since it is limited to tracking in eight consecutive frames only. Particle Video [128] prunes tracked points on occlusion and creates new tracks on disocclusion, however these are not linked together. TAPIR [112] combines the per-frame point localization from TAP-Net [126] with a temporal processing inspired by PIPs [127], but uses a time-wise convolution instead of fixed size frame batches. BootsTAP [129] is an improved TAPIR model, self-supervisedly fine-tuned on a large amount of YouTube video clips. CoTracker [113] processes query points with a sliding-window transformer that enables multiple tracks to influence each other. However, it works best when a single query point is tracked at a time, supported by an auxiliary grid of queries. SpatialTracker [130] extend it by adding 3D information from a monocular depth estimation method. Compared to our proposed approach, these methods do not track densely, but instead focus on tracking individual query points. It is possible to track all the points in batches, but it is slow. The DOT [131] tracker densifies the CoTracker correspondences with a specialized RAFT-like optical flow network.

OmniMotion [132] was designed to track densely. It pre-processes the video by computing optical flow between all pairs of frames. It represents

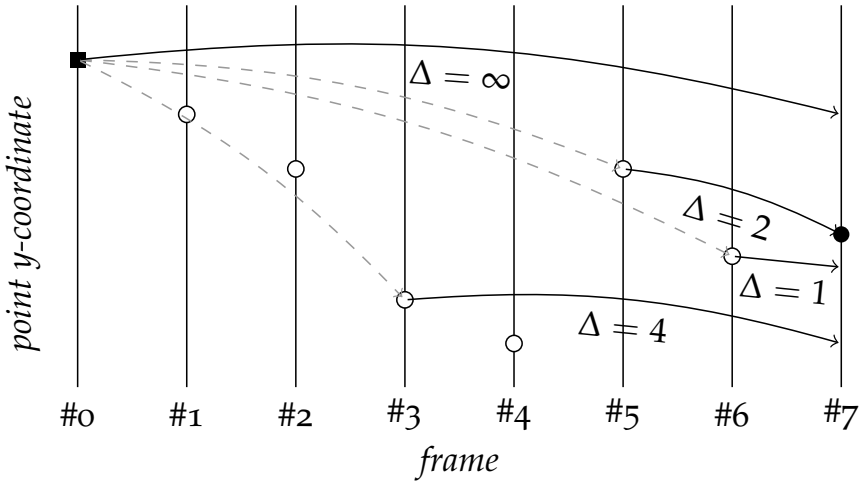


Figure 18: **Overview of the MFT**. MFT tracks a query point (*black square*) by chaining optical flows. Each chain consists of a previously computed chain from frame 0 up to frame $(t - \Delta)$ (*dashed arrow, white dot*), and an optical flow vector computed between frames $(t - \Delta)$ and t (*solid arrow*). MFT forms multiple candidate chains with varying Δ. The best candidate (*black dot*) is selected according to uncertainty and occlusion scores. This is done in parallel, independently for each pixel in the reference frame.

the whole video with a quasi-3D volume, a NeRF[133]-like network and a set of 2D↔quasi-3D bijections. The representation is globally optimized at inference time to obtain consistent motion estimates of all points in all frames of the video. While other test-time optimization approaches [134, 135] improve over the OmniMotion tracking speed, they are still extremely slow when compared to the sparse point trackers.

4.3 METHOD

The proposed method for long-term tracking of every pixel in a template is based on combining optical flow fields computed over different time spans, hence we call it Multi-Flow Tracker, or MFT in short. Given a sequence of $H \times W$ -sized video frames $I_0, I_1, \ldots, I_N$ and a list of positions on the reference (template) frame $\mathbf{p}_{i,0} = (x_i, y_i), i \in \{1, \ldots, HW\}$ the method predicts the corresponding positions $p_{i,t}$ in all the other frames $t \in \{1, \ldots, N\}$, together with an occlusion flag $o_{i,t}$. At time t, the MFT outputs are formed by combining the MFT result from a previous time $t - \Delta$, with the flow from $t - \Delta$ to the current frame t (see Fig. 18). Note that this is not combining only two flows, but appending single flow to a previously computed, arbitrarily long chain of flows. MFT constructs a set of candidate results with varying Δ, then the best candidate is chosen independently for each template position. To rank the candidates, MFT computes and propagates an occlusion map and an uncertainty map in addition to the optical flow fields. Detecting occlusions is necessary to prevent drift to occluding objects as shown in Fig. 19. The position uncertainty serves to pick the most accurate of the candidates. We now describe how the occlusion and uncertainty maps are formed, followed by a detailed description of the proposed MFT.

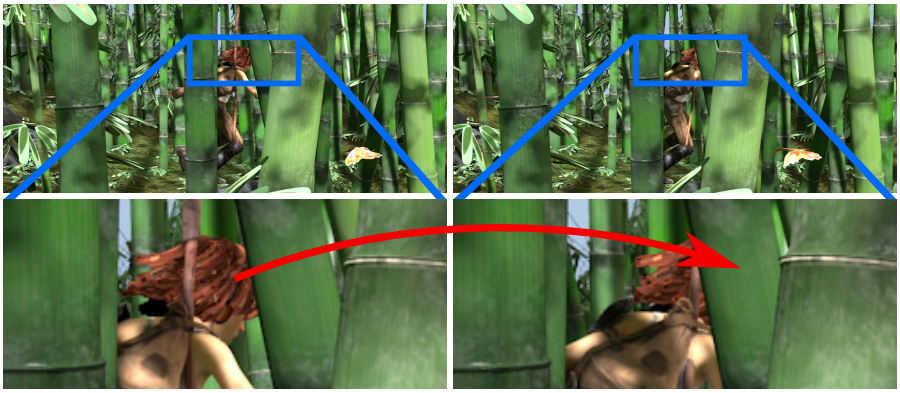


Figure 19: **Optical flow and occlusions.** OF methods are typically [22, 27] trained to ignore occlusions and to predict the ground-truth flow (*red*) even when occluded in the second frame. Continuing tracking after an occlusion would result in the target drifting to the occluding object. Example from Sintel [33].

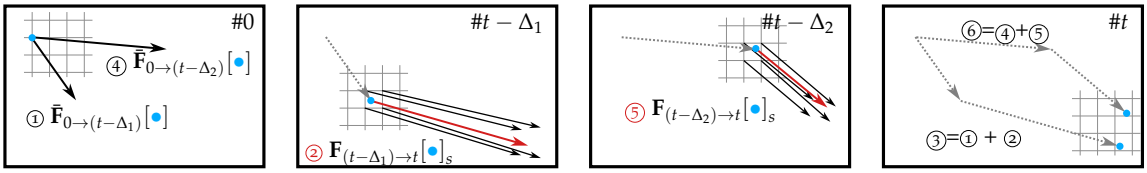


Figure 20: **Schematic explanation of the MFT tracking procedure.** At the current frame, time t (*right*), the tracker creates a set of result candidates, each formed by a different chain of optical flows. In this example, the first candidate ③ is formed by chaining the result ① previously computed in time $(t-\Delta_1)$ with flow ② estimated between frames $(t-\Delta_1)$ and t. We use bilinear interpolation (*red*) to sample the flow field, since the positions in $(t-\Delta_1)$ usually do not align with the pixel grid. The flow ③ into the current frame t is constructed by summing the two flow vectors. We repeat this procedure for Δ_2, again summing the result ④ for frame $(t-\Delta_2)$ with ⑤ – the bilinearly sampled flow field from $(t-\Delta_2)$ to t. When chaining the flows, we also chain their occlusion and uncertainty maps. Finally, we select the candidate (③, or ⑥) with the lowest uncertainty score among the ones not occluded, or mark the result occluded when all candidates predict occlusion. Current point position shown in blue, grid-aligned flow vectors in black, interpolated flow vectors in red.

4.3.1 *Occlusion and Uncertainty*

Current optical flow methods typically compute the flow from a cost-volume inner representation and image features [22, 27, 21]. Given a pair of input images, I_a and I_b, the cost-volume encodes similarity between each position in I_a and (possibly a subset of) positions in I_b. We propose to re-use the cost-volume as an input to two small CNNs for occlusion and uncertainty estimation. In both cases we use two convolutional layers with kernel size 3. The first layer has 128 output channels and ReLU activation. Both networks take the same input as the flow estimation head and each outputs a $H \times W$ map.

Occlusion: Similar to [122, 23, 26], we formulate the occlusion prediction as a binary classification. The network should output 1 for any point in I_a that is not visible in I_b and 0 otherwise. We train it on datasets with occlusion ground-truth labels (Sintel [33], FlyingThings [31], and Kubric [136]) using standard cross-entropy loss. The trained CNN achieves 0.96 accuracy on Sintel validation set.

Uncertainty: We train the uncertainty CNN with the uncertainty loss function from [137, 45]

$$\mathcal{L}_u = \frac{1}{2\sigma^2} l_H(||\vec{x} - \vec{x}^*||_2) + \frac{1}{2}\log(\sigma^2) \tag{13}$$

where x is the predicted flow, x^* the ground truth flow, σ^2 the predicted uncertainty and l_H is the Huber loss function [138]. The uncertainty CNN predicts $\alpha = \log(\sigma^2)$ to improve numerical stability during training. We output σ^2 during inference.

We sum the occlusion loss and $\mathcal{L}_u$ weighted by $\frac{1}{5}$. Note that we only train the occlusion and uncertainty networks, keeping the pre-trained optical flow fixed.

4.3.2 *MFT – Multi-Flow Tracker*

The MFT tracker is initialized with the first frame of a video. It then outputs a triplet $\overline{\mathbf{FOU}}_{0\to t} = (\bar{\mathbf{F}}_{0\to t}, \bar{\mathbf{O}}_{0\to t}, \bar{\mathbf{U}}_{0\to t})$ at each consequent frame I_t. The $\bar{\mathbf{F}}_{0\to t}$ is a $H \times W \times 2$ map of position differences between frame number 0 and t, in the classical optical flow format. The $\bar{\mathbf{O}}_{0\to t}$ and $\bar{\mathbf{U}}_{0\to t}$ are $H \times W$ maps with the current occlusions and uncertainties respectively. On the initialization frame, all three maps contain zeros only (no motion, no occlusion, no uncertainty), on the first frame after initialization, the triplet is directly the output of the optical flow network and the proposed occlusion and uncertainty CNNs. On all the following frames, the results are not the direct outputs of the network, but instead they are formed by chaining two $(\mathbf{F}, \mathbf{O}, \mathbf{U})$ triplets together.

The MFT is parameterized by D, a set of time deltas. We set $D = \{\infty, 1, 2, 4, 8, 16, 32\}$ (logarithmically spaced) by default. For every $\Delta \in D$, we create a result candidate that is formed by chaining two parts – a previously computed result $\overline{\mathbf{FOU}}_{0\to(t-\Delta)}$ and a network output $\mathbf{FOU}_{(t-\Delta)\to t}$ as shown in Fig. 20. To keep the notation simple, we write $(t-\Delta)$, but in fact we compute $\max(0, t-\Delta)$ to avoid invalid negative frame numbers.

To do the chaining, we first define a new map $\bar{\mathbf{P}}_{(t-\Delta)}$ storing the point positions in time $(t-\Delta)$. For each position $\mathbf{p} = (x, y)$ in the initial frame, the position in time $(t-\Delta)$ is calculated as

$$\bar{\mathbf{P}}_{(t-\Delta)}[\mathbf{p}] = \mathbf{p} + \bar{\mathbf{F}}_{0\to(t-\Delta)}[\mathbf{p}], \tag{14}$$

where $\mathbf{A}[\mathbf{b}]$ means the value in a map $\mathbf{A}$ at integer spatial coordinates $\mathbf{b}$. To form the candidate $\mathbf{F}^{\Delta}_{0\to t}$, we add the optical flow $\mathbf{F}_{(t-\Delta)\to t}$, sampled at the appropriate position to the motion between frames 0 and $(t-\Delta)$.

$$\mathbf{F}^{\Delta}_{0\to t}[\mathbf{p}] = \bar{\mathbf{F}}_{0\to(t-\Delta)}[\mathbf{p}] + \mathbf{F}_{(t-\Delta)\to t}\Big[\bar{\mathbf{P}}_{(t-\Delta)}[\mathbf{p}]\Big]_s \tag{15}$$

where $\mathbf{A}[\mathbf{b}]_s$ means the value in a map $\mathbf{A}$ sampled at possibly non-integer spatial coordinates $\mathbf{b}$ with bilinear interpolation. When chaining two occlusion scores, we take their maximum.

$$\mathbf{O}^{\Delta}_{0\to t}[\mathbf{p}] = \max\Big(\bar{\mathbf{O}}_{0\to(t-\Delta)}[\mathbf{p}]; \mathbf{O}_{(t-\Delta)\to t}\Big[\bar{\mathbf{P}}_{(t-\Delta)}[\mathbf{p}]\Big]_s\Big) \tag{16}$$

Since we threshold the occlusion scores in the end to get a binary decision, this corresponds to an "or" operation – the chain is declared occluded whenever at least one of its parts is occluded.

The uncertainties are chained by addition, as they represent the variance of the sum of flows, assuming independence of individual uncertainties.

$$\mathbf{U}^{\Delta}_{0\to t}[\mathbf{p}] = \bar{\mathbf{U}}_{0\to(t-\Delta)}[\mathbf{p}] + \mathbf{U}_{(t-\Delta)\to t}\Big[\bar{\mathbf{P}}_{(t-\Delta)}[\mathbf{p}]\Big]_s \tag{17}$$

We repeat the chaining procedure for each $\Delta \in D$ to obtain up to $|D|$ different result candidates. Finally, we select the best Δ, Δ^* according to candidate uncertainty and occlusion maps. In particular, we pick the Δ that has the lowest uncertainty score among the unoccluded candidates. When all the candidates are occluded (occlusion score larger than a threshold θ_o), all candidates are equally good and the first one is selected.

$$\boldsymbol{\Delta}^*[\mathbf{p}] = \operatorname*{argmin}_{\Delta\in D} \mathbf{U}^{\Delta}_{0\to t}[\mathbf{p}] + \infty \cdot [\![\mathbf{O}^{\Delta}_{0\to t}[\mathbf{p}] > \theta_o]\!], \tag{18}$$

where $[\![x]\!]$ is the Iverson bracket (equal to 1 when condition x holds, 0 otherwise). Notice that we select the Δ^* independently for each position. For example with $D = \{\infty, 1\}$, the flows are computed either directly between the template and the current frame ($\Delta = \infty$), or from the previous to the current frame ($\Delta = 1$) as in the traditional OF setup. For some parts of the image, it is better to use $\Delta = \infty$, because having a direct link to the template does not introduce drift. On the other hand, on some parts of the image the appearance might have significantly changed over the longer time span, making the direct flow not reliable at the current frame. In such case a long chain of $\Delta = 1$ flows might be preferred. Note that MFT usually switches back and forth between the used Δs during the tracking. A single template query point might be tracked using a chain of $\Delta = 1$ flows for some time, then it might switch to the direct $\Delta = \infty$ flow for some frames (possibly undoing any accumulated drift), then back to $\Delta = 1$ and so on.

The final result at frame t is formed by selecting the result from the candidate corresponding to Δ^* in each pixel, *e.g.*, for the flow output $\bar{\mathbf{F}}_{0\to t}$ we have

$$\bar{\mathbf{F}}_{0\to t}[\mathbf{p}] = \mathbf{F}^{\boldsymbol{\Delta}^*[\mathbf{p}]}_{0\to t}[\mathbf{p}] \tag{19}$$

Finally, MFT memorizes and outputs the resulting triplet $\overline{\mathbf{FOU}}_{0\to t}$ and discard memorized results that will no longer be needed (more than $\max(D \setminus \{\infty\})$ frames old). Given query positions $\mathbf{p}_{i,0}$ on the template frame 0, we compute their current positions and occlusion flags by bilinear interpolation of the $\overline{\mathbf{FOU}}$ result.

$$\mathbf{p}_{i,t} = \mathbf{p}_{i,0} + \bar{\mathbf{F}}_{0\to t}[\mathbf{p}_{i,0}]_s \tag{20}$$

$$o_{i,t} = \bar{\mathbf{O}}_{0\to t}[\mathbf{p}_{i,0}]_s \tag{21}$$

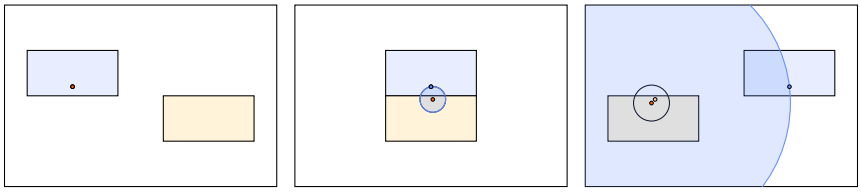


Figure 21: Uncertainty chaining near motion boundaries may significantly underestimate the true error. Two rectangles moving in opposite direction shown on three consecutive frames. Red point is the estimated position. Blue point is the ground-truth position. Large blue circle shows the true uncertainty (position error) of the tracker. Black circle shows the sum of uncertainties estimated (in this example perfectly) between consecutive images. Orange point in the last frame shows the ground-truth position of the red point in the middle frame.

4.3.3 *Implementation Details*

For the optical flow, we use the official RAFT [22] implementation with author-provided weights. Both the occlusion and the uncertainty CNNs operate on the same inputs as the RAFT flow regression CNN, *i.e.* samples from the RAFT cost-volume, context features, and Conv-GRU outputs. We train on Sintel [33], FlyingThings [139], and Kubric [136]. We sample training images with equal probability from each dataset. Because the Kubric images are smaller than the RAFT training pipeline expects, we randomly upscale them with scale ranging between 3.2× and 4.6×. We train the occlusion and the uncertainty network for 50k iterations with the original RAFT training hyperparameters, which takes around 10 hours on a single GPU.

The MFT tracker is implemented in PyTorch and all the operations are performed on GPU. Note that the optical flows and the occlusion and uncertainty maps can be pre-computed offline. When the $\Delta = \infty$ is not included in D, the number of pre-computed flow fields needed to be stored in order to be able track forward or backward from any frame in a video is less than $N2|D|$. Pre-computing flows for $\Delta = \infty$ (direct from template) and all possible template frames is not practical, as the number of stored flow fields grows quadratically with the number of frames N. With the flows for other Δs pre-computed, MFT needs to compute just one OF per frame during inference, so the tracking speed stays reasonably fast.

On a GeForce RTX 2080 Ti GPU (i7-8700K CPU @ 3.70GHz), the chaining of the flow, occlusion and uncertainty maps takes approximately 1.3ms for each Δ candidate with videos of 512×512 resolution. On average, the preparation of all the result candidates takes 8ms. The per-pixel selection of the best one adds additional 0.6ms. Computing a single RAFT flow, including the extra occlusion and uncertainty outputs, takes 60ms. Altogether, the full MFT runs at 2.3FPS. With pre-computed flows MFT runs at over 100FPS, making it suitable for interactive applications in, *e.g.*, film post-production. We set $\theta_o = 0.02$ empirically.

4.3.4 *MFTIQ: MFT with Independent Quality Estimation*

This section describes some improvements to the original MFT. The idea of estimating uncertainty and occlusion status for each optical flow vector is applicable to arbitrary OF methods. However, MFT occlusion and uncertainty estimation network was designed for RAFT in particular. While RAFT works well for small Δ between the frames, it was only trained on pairs of consecutive frames, where the motions are usually small and uncomplicated, the spatial relation between the objects does not significantly change, and the occlusion maps are small. Its performance drops as the Δ increases. There are some dense wide-baseline image matching methods proposed recently, like RoMa [43], that are not trained on consecutive frames, but rather on photos of landmarks taken by tourists. The camera pose changes much more between such photos than between frames in OF training data. The illumination changes are also more challenging as each photo was taken at a different time and a different day. There are also other alternatives to RAFT that have similar estimation quality, but faster runtime.

To integrate a different OF method — like RoMa for better performance on bigger flow Δ, or some faster flow — into MFT, one would first have to come up with some network architecture for the occlusion and uncertainty heads suitable for the given OF method. To address this issue we have developed MFT with Independent Quality, or MFTIQ in short. We propose a standalone network that gets two images and an OF between them as inputs and produces occlusion and uncertainty scores for each of the input optical flow vectors. This way the network does not have to be tailored to fit the particular OF estimator inner representations, like the RAFT cost-volume and features in MFT.

This approach has some advantages. First, after training the quality (uncertainty, occlusion) estimation network once, we can use any OF method in a plug-and-play fashion. The MFTIQ user can choose to use slower, but more accurate OF method, *e.g.* FlowFormer++ [28], RoMa [43], or faster, but less accurate OF like NeuFlow [140]. Also when a new and better OF is published, MFTIQ users get a free upgrade without any retraining.

Second, we can estimate the quality of the chain of flows as a whole, without having to deal with the chaining of the uncertainty scores. While the summation of uncertainties used in MFT is somewhat theoretically justified (see Eq. (17), Sec. 4.3.2), the assumptions about independence do not work well in practice, especially around motion boundaries as shown in Fig. 21. In MFTIQ we feed the quality estimation network the template and the current image and the flow field created by chaining flows in the MFT fashion (Eq. (15)). This way it is harder for the tracker to drift and produce long chains after incorrect jump, which was a weakness of the original MFT as described in Sec. 4.5. On the other hand, estimating the occlusion and uncertainty directly between the template and current frame is a harder task, since appearances and poses of all objects in the scene can change dramatically over the duration of the video.

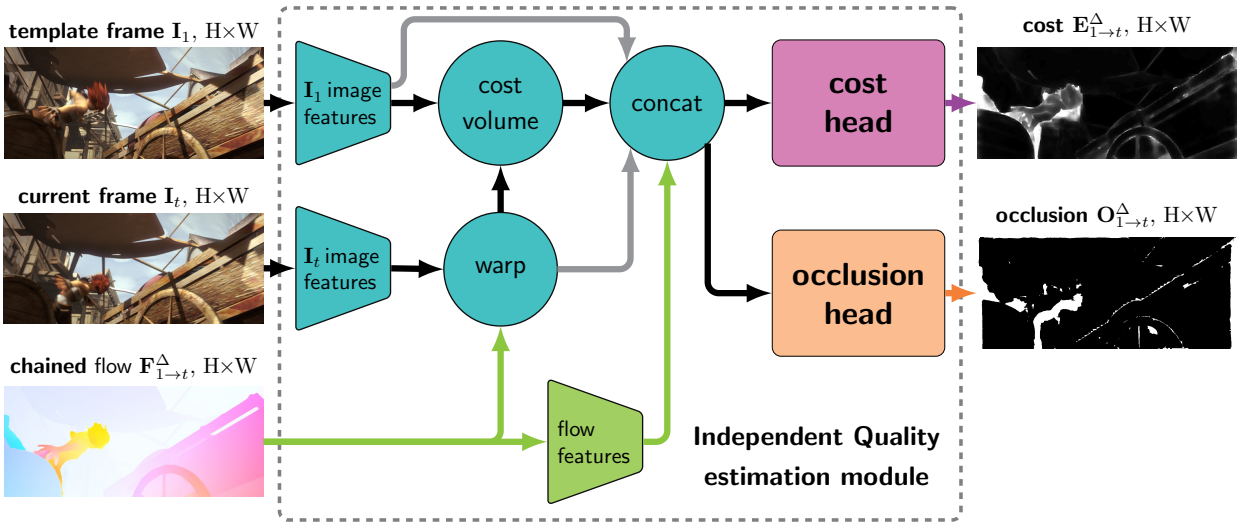


Figure 22: **Overview of the MFTIQ Independent Quality (IQ) estimation network.** First, image features are extracted from the template frame I_0 and the current frame I_t. Then, the current frame features are warped using the positions given by the chained flow $\mathbf{F}^{\Delta}_{0\to t}$. The now-aligned feature maps are compared with a local (displacement up to ± 3) correlation cost-volume. Finally a concatenation of the features extracted from both images and the flow are concatenated with the cost-volume and processed by two small CNNs to output the occlusion map and the cost map which together represent the quality of the input flow chain.

4.3.4.1 *Independent Flow Quality Estimation*

In contrast to MFT, where the uncertainties were chained, in MFTIQ we estimate the cost $\mathbf{E}$ and the occlusion map $\mathbf{O}$ directly as a function of the chained flow $\mathbf{F}^{\Delta}_{0\to t}$ and the two images it relates to, I_0 and I_t.

$$\{\mathbf{E}^{\Delta}_t, \mathbf{O}^{\Delta}_t\} = \mathcal{Q}\left(\mathbf{F}^{\Delta}_{1\to t}, I_1, I_t\right). \tag{22}$$

The cost map $\mathbf{E}$ functions analogously to the MFT flow chain uncertainty $\mathbf{U}$, but is trained with a different cost function. Cost $\mathbf{E}$ is analogous to MFT uncertainty $\mathbf{U}$ in that the lower values means higher positional accuracy. The independent quality estimation function $\mathcal{Q}$ is implemented as a neural network, the architecture and the training of which we describe in this section.

An overview of the architecture is shown in fig. 22. First, we extract image features to produce a $\frac{H}{4} \times \frac{W}{4}$ feature map. In particular, both images I_1 and I_t are processed by the DINOV2 [141] network. We bilinearly upscale the resulting coarse $\frac{H}{14} \times \frac{W}{14}$ feature map into the target $\frac{H}{4} \times \frac{W}{4}$ resolution. To add more spatially fine-grained information, we also compute the IMAGENET1K-pre-trained RESNET50 [94] CNN features and features from a custom shallow CNN. The features from all the feature providers (DINOV2, RESNET, custom CNN) are aggregated and compressed through a convolutional operation (from 5×32 channels down to 32 channels) to produce an additional *fused feature* for the cost-volume.

We resize all the resulting feature maps into the $\frac{1}{4}$ resolution and compress them with a convolutional layer to have 32 channels each.

WARPING + COST VOLUME The next stage in our process is the formation of a local Correlation Cost Volume (CCV), which serves to measure the similarity between the corresponding (as predicted by the optical flow chain) features, while also considering adjacent pixel information. To perform this, the feature maps from the current frame I_t, are warped to the template frame I_1 using the chained optical flow $\mathbf{F}_{0\to t}$, which is scaled to match the featuremap resolution. Then a local CCV (maximum displacement of 3px on the featuremap resolution) is independently computed for each input feature map, like in FLOWNET [20].

Finally, we concatenate the feature similarities computed by the cost-volumes with the image features and features computed from the chained optical flow. The resulting $\frac{H}{4} \times \frac{W}{4}$ featuremap with 424 channels is then used to estimate the flow-chain quality.

FLOW-CHAIN QUALITY ESTIMATION We use two three-layer CNN heads, each followed by a bilinear upsampling to the full image resolution, to estimate the cost and occlusion maps. The occlusion estimation CNN classifies each pixel as either occluded or non-occluded and is trained using standard binary cross-entropy loss, denoted as $\mathcal{L}^{\text{occl}}$.

The cost is constructed from $M = 5$ binary classifiers again trained by binary cross-entropy loss $\mathcal{L}^{\text{match}\,\theta}$. Pixels that have the flow end-point-error (EPE, euclidean distance from the ground truth) over θpx or are occluded belong to the positive class, while the visible and precisely matched (EPE under θpx) belong to the negative class. The binary classifiers differ in the EPE threshold $\theta \in \{1, 2, 3, 4, 5\}$, ranging from 1 to 5px.

During inference, the final cost map is constructed as a weighted average of the soft (Sigmoid activation) classification maps $\mathbf{E}_\theta$,

$$\mathbf{E} = \sum_{\theta=1}^{M} 2^{\theta-1} \mathbf{E}_\theta. \tag{23}$$

The $\mathbf{E}$ should be low for well matched points and high for poorly matched or occluded points.

The overall training loss, $\mathcal{L}$, is computed as follows:

$$\mathcal{L} = \frac{1}{H \times W} \sum_{i=1}^{H \times W} \mathbf{V}_i \left(\mathcal{L}_i^{\text{occl}} + \frac{1}{M} \sum_{\theta=1}^{M} \mathcal{L}_i^{\text{match}\,\theta} \right), \tag{24}$$

where $\mathbf{V}_i$ is a binary ground-truth validity flag of pixel i.

4.3.5 *Implementation Details*

For the DINOV2 features we use the author-provided ViT-S/14-reg network checkpoint. The ResNet50 [94] network, pre-trained on the ImageNet1K [142] dataset, is used to extract features from its first three blocks: the input block, residual block 1, and residual block 2. Each output feature is up-sampled to $\frac{H}{4} \times \frac{W}{4}$ and compressed to 32 channels using a convolutional layer.

The custom image features CNN is trained from scratch, and it is inspired by NEUFLOW's feature CNN [140]. Initially, an image pyramid is created by subsampling the input image at different scales (1/1, 1/2, 1/4). For each level of the image pyramid, a convolutional layer is applied with specific kernel sizes, strides, and padding to ensure the output resolution is $\frac{H}{4} \times \frac{W}{4}$ (k4:s4:p0 | k8:s2:p3 | k7:s1:p3). The outputs from each pyramid

level are concatenated and compressed to 32 channels using an additional convolutional layer.

We trained the independent quality network using a synthetic dataset from the Kubric rendering tool [136]. The dataset includes 200 sequences with a variable number of static and dynamic objects rendered at a 1024×1024 resolution, each 240 frames long. The sequence length is much longer than the typically used 24 or 48 frames. We had to ensure that the objects do not become static after falling to the ground as in the default Kubric scenario, otherwise the long sequences would not bring much. To do this and keep the objects non-intersecting, we left the default Kubric physical engine to simulate the scene for 48 frames, after which we disabled it and replayed the simulated motions back and forth for the rest of the video. The camera motion is generated independently, with the panning from TAPIR and a random camera shake to introduce motion blur and make the camera movement more realistic. Due to the independent non-looping motion of the camera, the resulting video is not repetitive and information-rich for the whole duration.

The training involved sampling random image pairs with temporal separations, *i.e.*, the flow Δ, ranging from 2 to 150 frames. We generated a pre-sampled set of 20,000 training pairs with dense[2] ground truth optical flow, occlusion, and validity masks $\mathbf{V}$. During each training iteration, the input optical flow chains were uniformly drawn from RAFT [22], ground-truth-initialized FlowFormer++ [28], and the ground truth flow. Both the optical flow and the input images were augmented and resized to 368×768 pixels.

The training was conducted on a single RTX A5000 GPU for approximately one day using a batch size of 8 for 200,000 iterations, with an initial learning rate of 2.5×10^{-3} and OneCycleLR [143] learning rate policy.

INFERENCE-TIME CACHING To speed up the proposed MFTIQ tracker, we cache and re-use intermediate results where possible. Namely, the image features are needed multiple times per frame and especially the DinoV2 network is slow, so we cache them in GPU memory. We also cache the optical flows, which is useful when tracking from multiple query frames, like in the strided TAP-Vid. Depending on the available memory, the caching is automatically done to GPU memory, CPU memory, or to mass storage device (typically a Solid-State Drive). If the application allows it, both the image features and the optical flows can be precomputed to get fast tracking.

When storing optical flows in the mass storage device, each channel is normalized and re-scaled to full range of unsigned 16 bit integers (between 0 and $2^{16} - 1$) and compressed using fast LZ4 algorithm, or as a PNG image (3 8-bit channels, one filled with zeros). The normalization parameters are stored alongside the compressed data for later de-normalization.

With optical flow and image features computed in advance, MFTIQ runs at 3.7 FPS on 720×1080 and at over 10 FPS on 512×512 video resolution. Without the pre-computation, the speed depends on the flow method used. For example on the 512×512 resolution it ranges from 0.2 FPS with RoMa to 2.7 FPS with RAFT, and 6 FPS with NeuFlow.

[2] The public version of the Kubric tool supports only sparse ground truth generation for point-tracking tasks.

	flow delta set D	DAVIS - first AJ	DAVIS - first $<\delta^x_{avg}$	DAVIS - first OA	DAVIS - strided AJ	DAVIS - strided $<\delta^x_{avg}$	DAVIS - strided OA
(1)	$\{1\}$	38.3	54.5	69.3	48.9	61.8	80.8
(2)	$\{\infty\}$	38.3	50.8	65.5	47.9	58.0	76.3
(3)	$\{\infty, 1\}$	46.4	63.7	76.7	55.0	68.1	85.8
(4)	$\{\infty, 1, 2, 4, 8, 16, 32\}$	47.3	**66.8**	**77.8**	**56.1**	**70.8**	**86.9**
(5)	$\{1, 2, 4, 8, 16, 32\}$	**47.4**	66.2	77.3	55.7	70.2	86.5

Table 5: **TAP-Vid Davis benchmark – evaluation of MFT on variants based on different sets** D of time differences Δ used in optical flow; ∞ indicates OF between the template and the current frame. Performance measured by occlusion accuracy (OA), position accuracy ($<\delta^x_{avg}$), and combined measure AJ. For definition of $<\delta^x_{avg}$and AJ, see text. Bold best, underline second.

4.4 EXPERIMENTS

Since there is no benchmark for dense long-term point tracking, we evaluate the MFT on the recently introduced TAP-Vid DAVIS and TAP-Vid Kinetics datasets [126] for sparse point tracking. The datasets consists of 30 videos from DAVIS 2017 [1] and 1189 videos from Kinetics-700 [144, 145] respectively, rescaled to 256×256 resolution, semi-automatically annotated with positions and occlusion flags of ≈ 20 selected points. MFTIQ is additionally evaluated on RoboTAP [146], which contains 265 videos of robotic arms picking up and dropping objects in a lab scenario and annotated like the TAP-Vid datasets.
Evaluation protocol: The TAP-Vid benchmark uses two evaluation modes: "first" and "strided". In the "first" mode, the tracker is initialized on the first frame where the currently evaluated ground-truth tracked point becomes visible, and is only evaluated on the following frames. In the "strided" mode, the tracker is initialized on frames $0, 5, 10, \ldots$ if the currently evaluated tracked point is visible in the given frame. The tracker is then evaluated on both the following and the preceding frames, we thus run our MFT method two times, forward and backward in time, starting on the initialization frame. The resulting tracks are shorter (half the video length on average), making the task simpler. Also, in the *first* mode, the query points are often on the object boundary or just after de-occlusion, further complicating the tracking.
Evaluation metrics: The TAP-Vid benchmark uses three metrics. The occlusion prediction quality is measured by occlusion classification accuracy (OA). The accuracy of the predicted positions, $<\delta^x_{avg}$, is measured by fraction of visible points with position error under a threshold, averaged over thresholds $1, 2, 4, 8, 16$. Both occlusion and position accuracy are captured by Average Jaccard (AJ), see [126] for more details.

4.4.1 *MFT Flow Delta Ablation*

In Table 5, we show the impact of using different sets D of Δs. We evaluate two baselines – (1) basic chaining of consecutive optical flows ($\Delta = 1$), and (2), computing the optical flow directly between the template and the current frame ($\Delta = \infty$). The first one performs better in all metrics, as the OF is

	Resolution H×W	DAVIS - first			DAVIS - strided		
		AJ	$<\delta^x_{avg}$	OA	AJ	$<\delta^x_{avg}$	OA
(1)	256×256	33.0	47.7	70.2	41.4	54.6	83.6
(2)	256×256→512×512	47.3	66.8	77.8	56.1	70.8	86.9
(3)	256×256→256×ratio	40.5	58.5	76.9	49.2	63.8	86.4
(4)	256×256→480×ratio	49.2	69.2	77.9	58.8	73.9	87.7
(5)	orig res.→480×ratio	52.3	71.9	**79.5**	61.9	76.1	**88.8**
(6)	orig res.→720×ratio	**54.0**	**74.0**	79.1	**64.3**	**78.7**	88.1

Table 6: **TAP-Vid Davis benchmark – evaluation of MFT for different image resolutions.** Performance measured by occlusion accuracy (OA), position accuracy ($<\delta^x_{avg}$), and combined measure AJ. For definition of $<\delta^x_{avg}$and AJ, see text. Bold best, underline second.

computed on pairs of consecutive images, which it was trained to do, and the test sequences are not long enough to induce significant drift by error accumulation. Note that the performance in the strided evaluation mode is better, because the sequences are on average two times shorter and contain less occlusions.

Combining the basic chaining with the direct OF, line (3) in Table 5, the performance increases in all metrics, showing the effectivity of the proposed candidate selection mechanism. Row (4) is the full MFT method which achieves the overall best results. The final experiment (5) works without the direct flow. This means that we can pre-compute all the optical flows needed to track from any frame in any time direction, and store them in storage space proportional to the number of frames $2N|D|$. Note that attempting to do that with $\infty \in D$ would result in storage requirements proportional to N^2. The last version achieves second best overall performance. Visual performance of the baselines and full MFT is shown in Fig. 23. All results in Table 5 were obtained on 2× upscaled images as discussed in the next section which is equivalent to adding one upsampling layer to the RAFT feature pyramid.

4.4.2 *MFT Input Resolution Ablation*

The official TAP-Vid benchmark is evaluated on videos rescaled to 256×256 resolution, which is small compared with the RAFT training set. Because of this, we upscale the 256×256 videos to 512×512 resolution. In all the experiments, the output positions are scaled back to the 256×256 resolution for evaluation. Rows (1) and (2) in Table 6 show that this upscaling improves the performance by a large margin on all three metrics. This shows that RAFT is sensitive to input sizes, note that no information was added to the images when upscaling.

The aspect ratio of the original videos is changed during the scaling from full DAVIS resolution to the 256×256. This makes the video contents appear distorted and changes the motion statistics. Consequently we perform several experiments with varying video resolutions but keeping the original aspect ratio. In the first two, (rows (3), (4) in Table 6), we upsample the 256×256 videos. This way we stick as close to the TAP-Vid protocol as possible, only

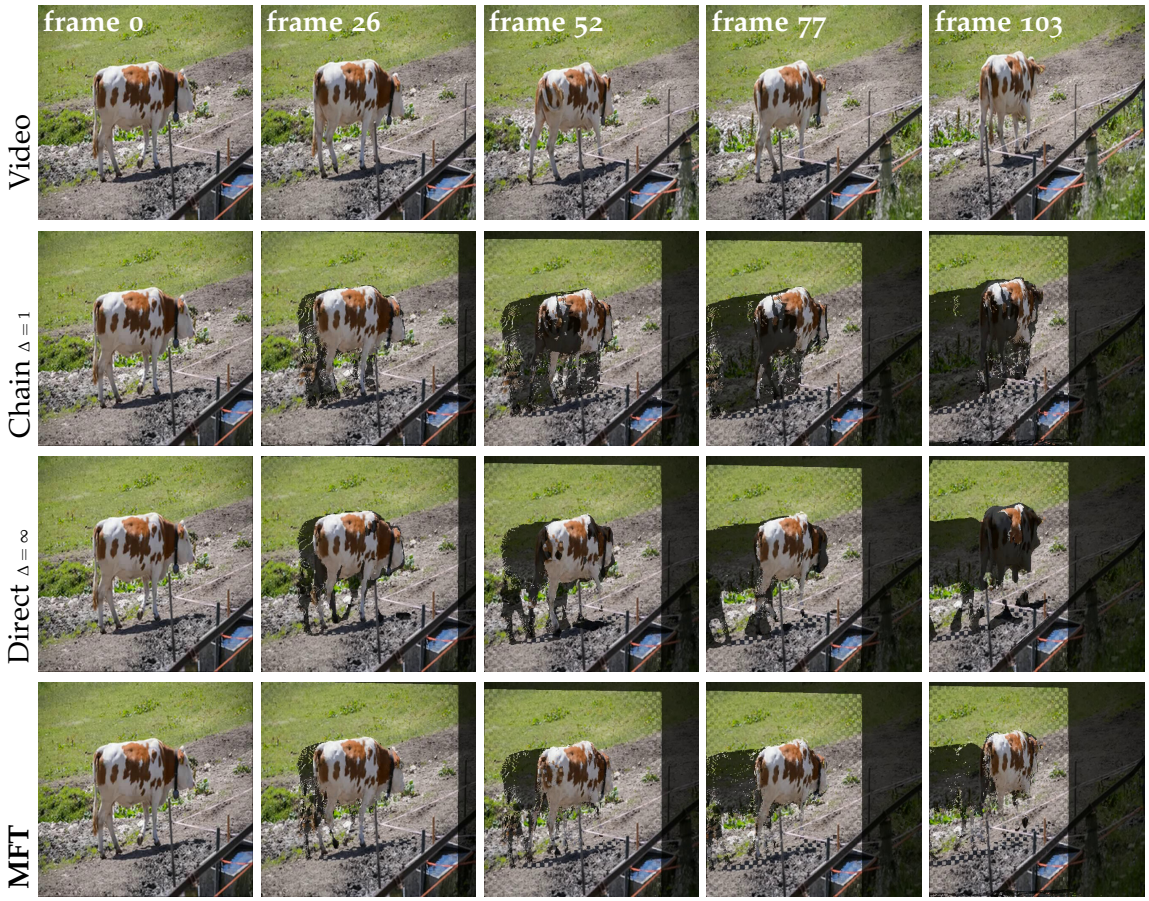


Figure 23: Result visualizations sampled at 25%, 50%, 75% and 100% of the input video (*top*) length. We take the first frame of a video and set its transparency with a checkerboard pattern. We then warp the resulting image using the outputs of each method and overlay the result on the current frame. The checkerboard pattern is visible when the tracking results are incorrect, or when the illumination changed between the template and the current frame. Pixels without a correspondence on the template frame are darkened. *Row 2:* simple flow chaining $\Delta = 1$. A short occlusion by the tail makes the tracker lose track in the back half of the cow. *Row 3:* direct flow $\Delta = \infty$. The tracker survives the occlusion but loses track when the cow rotates away from the camera. *Bottom:* the proposed MFT handles both the short occlusion and the appearance change, tracking well on background and most of the cow's body. All trackers fail on the legs which are too thin for the RAFT optical flow. Best viewed zoomed-in on a screen.

requiring the original video aspect ratio as an extra input. In (3), we keep the image height unchanged and only upscale the width such that the aspect ratio is not changed wrt the full resolution videos. All the metrics improve compared to the no scaling variant (1). Also, when we upscale the images to larger size (4), the performance increases.

In the last two rows (5), (6), we skip the TAP-Vid downscaling to 256 × 256 and instead downscale to the target resolution directly from the full-resolution DAVIS videos. This preserves high-frequency details more than doing the downscale-upscale cycle. Thanks to this, row (5) is better than (4), although the input resolution is the same in both. Even larger resolution (6) again improves the $<\delta^x_{avg}$and the AJ metric for the cost of small (below one percent point) decrease in occlusion accuracy.

Because we downscale directly from the full resolution, without the 256 × 256 intermediate step, the results of (5) and (6) are not directly comparable with the original TAP-Vid benchmark table, but are closer to a real-world scenario.

4.4.3 *MFT Comparison With the State-of-the-Art*

On the TAP-Vid benchmark, the proposed MFT tracker performs third best, after the state-of-the-art sparse point-tracking methods [113, 112], outperforming the other dense point tracker OmniMotion [132]. MFT runs at over 2FPS, which is orders of magnitude faster than the alternative methods evaluated densely, tracking every pixel and not just selected few. The speed/performance balance makes MFT favorable for dense point-tracking. Additionally, the optical flows can be pre-computed (only $2N \log N$ flows needed for a video of length N with logarithmically spaced flow delta set D) resulting in tracking at over 100FPS from any frame in the video, both forward and backward. This makes MFT a good candidate for interactive applications such as video editing. The complete results, including the inference speeds, are shown in Table 7. Both MFT and OmniMotion [132] can be seen as post-processing of a set of RAFT optical flows. The MFT strategy performs better than the complex model and global optimization in OmniMotion.

TRACKING SPEED In Tab. 7, we show inference speeds in a scenario of dense tracking (every pixel) on 512 × 512 video of 50 frames. Here, we describe how we computed the numbers. The 2.32 FPS for MFT was directly measured, including the computation time for RAFT optical flows. We have also measured the official CoTracker v1 [113] implementation in the dense tracking configuration on 50 frame 512 × 512 video, resulting in 0.04 FPS. Note that CoTracker achieves better results in its default setting — tracking a single query accompanied by an auxiliary query grid at a time. This would result in even lower FPS. Both MFT and CoTracker experiments were conducted on a single GeForce RTX 2080 Ti GPU.

We did not measure the speeds of the other methods, and instead estimated them as follows. OmniMotion [132], is trained for approximately 9 hours on each sequence using an A100 GPU[3]. We have estimated the runtime by dividing the 50 frames by those 9 hours. This does not include the pre-

[3] https://github.com/qianqianwang68/omnimotion/tree/ad080e750a8b67cc568a79b0e7f049e420fa895a#training accesed 2023-08-30

Method	FPS	DAVIS - first			DAVIS - strided			Kinetics - first		
		AJ	$<\delta^x_{avg}$	OA	AJ	$<\delta^x_{avg}$	OA	AJ	$<\delta^x_{avg}$	OA
TAP-Net [126]	0.11†	33.0	48.6	78.8	38.4	53.1	82.3	38.4	54.4	80.6
PIPs [127]	2e-4†	-	-	-	42.0	59.4	82.1	31.7	53.7	72.9
OmniMotion [132]	2e-3†	-	-	-	51.7	67.5	85.3	-	-	-
MFT (ours)	**2.32**	51.1	67.1	84.0	56.1	70.8	86.9	39.6	60.4	72.7
TAPIR [112]	0.04†	56.2	70.0	86.5	61.3	72.3	87.6	49.6	64.2	85.0
CoTracker [113]	0.04	60.6	75.4	89.3	64.8	79.1	88.7	48.7	64.3	86.5

Table 7: **Evaluation on TAP-Vid benchmark. MFT performs well while being orders of magnitude faster than other methods when evaluated densely.** Performance measured as in Table 5. Results for other methods are from [126, 113, 132, 112]. FPS: speed of dense (every pixel) tracking on 512 × 512 video in Frames Per Second. Speeds marked with † were extrapolated from timing info in [112, 132].

processing time, which includes among other steps computing RAFT flows between all pairs of frames, making our estimate of 0.002 FPS optimistic.

For TAP-Net [126], PIPs [127] and TAPIR [112], we have used the timing info in the Table 9 in the appendix of [112]. This table lists the execution time of all three methods with varying number of queries and varying sequence length. All the measurements were performed on a 256 × 256 resolution video using a V100 GPU. As we want to access the inference speed on dense tracking on an arbitrary 512 × 512 video (of length 50), we have extrapolated the timing on 50 frames long video with 50 query points by multiplying the reported runtime by $512^2/50$, as if the methods would track densely in batches of 50 query points. While somewhat better parallelization should be possible, tracking all the queries at the same time is not possible due to high GPU RAM usage. Also this estimate does not include the increased computation needed to process 512 × 512 videos.

MFT BADJA EVALUATION In addition to TAP-Vid DAVIS, we evaluate the MFT on BADJA [147] benchmark with videos of animals annotated with 2D positions of selected joints. The benchmark measures the percentage of points with position error under a permissive threshold $0.2\sqrt{A}$, where A is the area of the animal segmentation mask. Thanks to this, the MFT performs well even though the ground truth points (joints) are located under the surface, and thus, MFT cannot track them directly. In Table 8, we evaluate against the BADJA results of PIPs [127] and their RAFT baseline. In terms of median of the per-sequence results, MFT performs the best. The mean score is affected by a single failure sequence, *dog-a*, on which the dog turns shortly after the first frame, making most of the tracklets occluded. The assumption that a joint can be approximately tracked by tracking a nearby point on the surface becomes invalid in such case.

4.4.4 *MFTIQ Plug-n-Play Optical Flow*

After training the MFTIQ flow quality estimation with RAFT [22] and ground-truth-initialized FlowFormer++ [28], we fixed the model and evaluated it with various different OF methods. The table 9 shows that the RAFT-based

	a	b	c	d	e	f	g	Avg.	Med.
RAFT	64.6	65.6	69.5	13.8	39.1	37.1	29.3	45.6	39.1
PIPs	76.3	81.6	**83.2**	**34.2**	44.0	**57.4**	59.5	**62.3**	59.5
MFT	**81.8**	**82.0**	75.7	6.9	**47.9**	55.8	**62.7**	59.0	**62.7**

Table 8: **BADJA [147] benchmark – evaluation of MFT against PIPs [127].** Performance measured by the PCK-T measure, *i.e.*, the percentage of points with error under a threshold. Bold best. Results for PIPs and RAFT from [127]. The labeled individual sequences include (a) bear, (b) camel, (c) cows, (d) dogs-a, (e) dog, (f) horse-h, and (g) horse-l.

				OF runtime [ms] ↓	
method	AJ ↑	$<\delta^x_{avg}$ ↑	OA ↑	512x512	720x1080
MFT [10]	56.28	71.03	86.96	47	142
MFTIQ with					
RAFT [22]	60.54	74.22	84.42	47	142
NeuFlow [140]	55.73	70.26	80.87	**10**	**18**
MemFlow [17]	62.30	75.97	85.95	121	610
FFormer++ [28]	62.72	76.22	86.34	142	782
RoMa [43]	**65.67**	**79.82**	**87.75**	714	729

Table 9: TAP-Vid DAVIS [126] (strided) evaluation with single MFTIQ model using various OF methods. The first two rows compare the original MFT with the proposed MFTIQ both using the RAFT [22] OF. The rest of the table shows MFTIQ results when used with different OF methods. Runtime of a single OF computation shown on right.

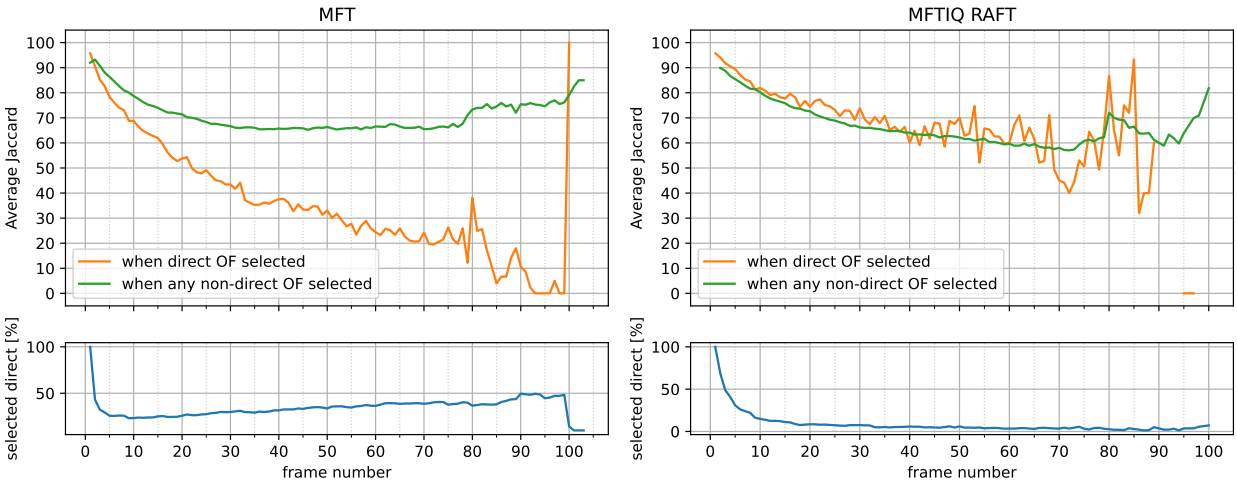


Figure 24: **Comparison of flow candidate selection in MFT *(left)* and MFTIQ *(right)*.** MFT often selects *(blue)* the *direct* optical flow, *i.e.* the flow chain with $\Delta = t - 1$ with probability increasing during the video. The probability of the selected *direct* flow to be accurate as measured by Average Jaccard (AJ) is, however, decreasing with time *(orange)* and choosing a different Δ would be more better *(green)*. In contrast, the proposed MFTIQ *(right)* chooses the direct optical flow more conservatively *(bottom)* and mostly when it has high accuracy *(orange)*. Both methods are evaluated on TAP-Vid DAVIS strided using RAFT OF. Non-direct OF accuracy *(green)* represents the average over all cases, when some $\Delta \neq t - 1$ was selected.

MFTIQ already outperforms the original MFT. More importantly, we get even better results when using other off-the-shelf optical flows and dense matchers. The best performance is achieved with the wide-baseline matcher RoMa [43] thanks to its ability to match densely both between consecutive and between more distant frames. Table 9 also lists the runtime of the respective OF methods, measured on a RTX A5000 GPU. While the best-performing RoMa is also the slowest on smaller images, it scales better than the second-best FlowFormer++ to larger images. Depending on the intended application, one could also use a fast optical flow method, such as NeuFlow [140], for a cost of reduced tracking quality. For the rest of the experiments we use the RoMa-based MFTIQ.

4.4.5 *MFTIQ vs MFT Chain Selection*

We further evaluate the MFTIQ chain selection and how it compares to the original MFT on TAP-Vid DAVIS. The fig. 24 shows that the uncertainty score chaining of MFT leads to a significant preference of selecting short chains with big Δs. In particular, the optical flow matching directly between the template and the current frame ($\Delta = t - 1$) without chaining is selected with probability increasing with the current frame number. However the probability of this selection being accurate decreases rapidly during the video. On the other hand MFTIQ selects the short chains with big deltas conservatively, keeping the result accuracy high.

4.4.6 *MFTIQ evaluation against state-of-the-art*

The overall results of the proposed RoMa-based MFTIQ tracker are shown in table 10. MFTIQ achieves the best (DAVIS) and the second-best (RoboTAP, Kinetics) position accuracy $<\delta^x_{avg}$. This is thanks to the quality of the used RoMa dense matcher. Note that we also used RoMa with MFT in our paper [11], however due to better flow quality estimation, MFTIQ performs much better on all metrics. Also we have designed MFTIQ to be independent on the OF method, so we expect it to get better with future even-higher-quality optical flows and dense matchers without re-training.

The occlusion accuracy (OA) of MFTIQ is comparatively lower, also affecting the overall AJ score. While it is an improvement over MFT, achieving state-of-the-art occlusion accuracy is yet an open challenge.

While MFTIQ does not achieve performance as good as the most recent sparse point trackers, it tracks densely and out-performs the original MFT. Note that the point trackers in 10 are not *causal*, *i.e.*, the trackers can "see" into the future which is helpful to resolve occlusions. Both MFT and MFTIQ only use the previous frames. For dense tracking the inference time is significantly faster than methods with similar accuracy, as measured by the points-per-second metric in table 10.

4.4.7 *Planar object tracking with MFTIQ*

In addition to the point-tracking benchmark, we have evaluated the proposed MFTIQ on the planar object tracking task on the POT-210 [8] benchmark described in section 3.5. This dataset captures the flat objects in seven challenging scenarios: *motion blur*, *occlusion*, *out-of-view*, *perspective distortion*,

method	PPS↑	DAVIS strided AJ↑	DAVIS strided $<\delta^x_{avg}$↑	DAVIS strided OA↑	DAVIS first AJ↑	DAVIS first $<\delta^x_{avg}$↑	DAVIS first OA↑	RoboTAP first AJ↑	RoboTAP first $<\delta^x_{avg}$↑	RoboTAP first OA↑	Kinetics first AJ↑	Kinetics first $<\delta^x_{avg}$↑	Kinetics first OA↑
TAP-Net [126]	† 555	38.4	53.1	82.3	33.0	48.6	78.8	45.1	62.1	82.9	38.5	54.4	80.6
CoTracker [113]	‡ 0.8	64.8	79.1	88.7	60.6	75.4	**89.3**	54.0	65.5	78.8	48.7	64.3	**86.5**
TAPIR [112]	† 200	61.3	72.3	87.6	56.2	70.7	86.5	59.6	73.4	**87.0**	49.6	64.2	85.0
BootsTAP [129]	–	**66.4**	78.5	**90.7**	**61.4**	74.0	88.4	**64.9**	**80.1**	86.3	**54.7**	**68.5**	86.3
MFT[10]	10671	56.3	71.0	87.0	51.1	67.1	84.0	–	–	–	39.6	60.4	72.7
MFT RoMa[11]	–	58.0	77.2	80.5	52.1	72.7	77.1	–	–	–	–	–	–
MFTIQ [12]	709	65.7	**79.8**	87.8	59.9	**75.5**	84.5	60.0	77.5	85.2	48.7	65.9	85.2

Table 10: MFTIQ RoMa evaluation on TAP-Vid [126] and RoboTAP [146] benchmarks. On the Kinetics dataset, MFTIQ was evaluated only on the first 465 sequences due to time constraints. Results of the other trackers were taken from their papers and from [129] in case of RoboTAP. The RoMa-based correspondences chained by MFTIQ provide a very good position precision ($<\delta^x_{avg}$) - best on DAVIS, second on RoboTAP and Kinetics. The occlusion accuracy (OA) is lower, also affecting the AJ score. The speed is compared with points-per-second (PPS). Values with † obtained from [112] and ‡ from [148].

in-plane *rotation*, *scale change*, and *unconstrained* combining all of the previous challenging factors. From these only the partial occlusion factor is present in TAP-Vid point-tracking benchmark.

4.4.7.1 *Point-tracking on POT-210*

The POT-210 annotations can be converted into dense correspondences on the planar target by projecting the pixel coordinates in the first frame initial mask with the ground-truth homography. We use this point-tracking ground truth to evaluate MFTIQ on all the different scenarios. Since there is no occlusion ground truth available on POT-210, we evaluate only the $<\delta^x_{avg}$TAP-Vid metric. We scale the output coordinates to 256×256 resolution as usual [126] and evaluate with the standard $1, 2, 4, 8, 16$ point error thresholds. The results in table 11 indicate overall good performance, with RoMa-based MFTIQ being particularly good on *rotation* and *scale change* scenarios compared to the plain RoMa.

4.4.7.2 *Planar tracking on POT-210*

To use MFTIQ as a planar tracker, we initialize it on the first frame and let it tracking all the initial frame pixels to get dense correspondences between the first and the current frame. On each frame we mask out the background correspondences, *i.e.* outside the initial rectangle on the first frame. Finally we use the correspondences and RANSAC to estimate a planar homography $\mathbf{H} \in \mathbb{R}^{3\times3}$ mapping from the initial to the current frame. Inspired by a optical-flow-based state-of-the-art planar tracker WOFT [9], we further increase the tracking accuracy by the *pre-warping* trick. We warp the current image into the template view using $\mathbf{H}^{-1}$, followed by estimating the residual flow and the residual homography $\mathbf{H}_r$ between the template and the warped current frame. The original homography $\mathbf{H}$ is combined with the residual

challenge	RoMa $<\delta^x_{avg}\uparrow$	MFTIQ $<\delta^x_{avg}\uparrow$
blur	88.4	86.9 (−1.5)
occlusion	99.2	96.9 (−2.3)
out-of-view	90.7	89.2 (−1.5)
perspective	96.8	94.8 (−2.0)
rotation	72.8	96.5 (+23.7)
scale	92.2	98.0 (+5.8)
unconstrained	93.5	93.3 (−0.2)
all	90.5	93.7 (+3.2)

Table 11: MFTIQ RoMa performance on POT-210 using a point-tracking metric, compared to plain RoMa. While the plain RoMa performs slightly better on some of the challenging scenarios, MFTIQ is significantly better on *rotations* and *scale change* due to the flow chaining, making it better on average – *all*.

homography $\mathbf{H}_r$ to get the final estimate $\mathbf{H}^* = \mathbf{H}\mathbf{H}_r$. Finally we transfer the control points from the initial frame into the current frame with $\mathbf{H}^*$ to get their current position.

We set new state-of-the-art on the POT-210 benchmark as shown in table 12. The MFTIQ planar tracker performs particularly well on the *blur* subset of POT-210, which contains many frames on which trackers fail due to big motion blur. MFTIQ is able to recover from such failures by "jumping" over the problematic frames using the optical flows with bigger frame delta.

4.5 LIMITATIONS

One MFT weakness we have observed are spurious re-detections. MFT sometimes matches out-of-view parts of the template to visually similar parts of the current frame. Single such incorrect re-detection can "restart" a flow chain, affecting the performance for the rest of the video. A typical example is tracking of a point on a road surface. When the camera moves such that the original point moves far out of view, the tracklet sometimes suddenly jumps to a newly uncovered patch of the road. Both the appearance of the incorrectly matched point and its image context is often very similar to the template frame, *e.g.*, a relatively texture-less black road some distance below a car wheel.

One of the goals of the MFTIQ was to get a better quality occlusion (or match/non-match) decision. Since the uncertainties are not chained but estimated directly between the template and the current frame, MFTIQ doesn't have the failure mode of MFT where a single bad re-detection could make the tracker lost for the rest of the sequence. On the other hand, there are two cases in which MFTIQ cannot work well. First, when the point (and the surrounding context) appearance changes too much, making the features extracted from the initial and the current frame dissimilar.

Imagine a video in which a target object is close to the camera on the initial frame and gradually moves away as shown in fig. 25. This way, its images area shrinks by many orders of magnitude to just few pixels. In

method	BL	OCCL	OOV	PERS	ROT	SC	UNC	all
LISRD [89, 7]	54.1	93.8	83.7	65.0	86.3	30.0	67.1	68.3
HDN [100]	48.8	78.2	66.1	54.4	91.4	94.8	60.7	70.9
CGN [149]	41.6	88.1	82.8	76.5	96.1	90.3	72.4	78.5
WOFT [9]	60.4	**98.6**	96.3	95.4	99.3	94.0	88.2	90.4
HVC-Net [150]	60.5	**98.6**	**97.2**	92.7	99.3	100.0	**90.1**	91.4
MFTIQ (ours)	**72.0**	**98.6**	95.0	**96.6**	**99.5**	**100.0**	89.1	**93.1**

Table 12: MFTIQ evaluation on planar tracking POT-210 [8] benchmark. Percentage of frames with alignment error under 5px threshold evaluated on the improved ground truth (section 3.5.1). The RoMa-based MFTIQ followed by a RANSAC homography estimation on the resulting correspondences sets a new state-of-the-art performance. It achieves the most significant performance gain +11.5% on the *BLur* sequences.

Figure 25: MFTIQ failure case. Example of a video with zooming out. Every 100th frame shown. Even with flow correctly chained *(red circle)* it is impossible to estimate the occlusions and uncertainties directly between the first and the last frame. Video source: https://www.youtube.com/watch?v=r1bIXAV9Cnc

Figure 26: Tracking failure due to mixing of information from foreground and background. Frames cropped around the tracked point (*red circle*). Query point on the first frame of the horsejump-high sequence [126] (*left*), the ground-truth position on frame 17 (*middle*), and the incorrect MFTIQ prediction on the same frame (*right*). The features of the query point describe both the horse (foreground) and the background structures. The incorrect prediction (*right*) is supported by large area of unchanged background.

such case it could be possible to track by chaining flows, uncertainties, and occlusions, but completely impossible to track by chaining flows and estimating uncertainties and occlusions directly between the initial frame and the current one.

A second failure case of the MFTIQ flow-independent estimation of flow quality are repetitive structures. For example a video of a fish school, where most of the fish look practically identically. In order to correctly evaluate the uncertainty and occlusion state, the independent quality module would have to take the motion history into account.

Another failure mode is related to large receptive fields of the optical flow and quality estimation networks and mixing of feature representation of the foreground and the background. Sometimes the tracker stops tracking the target point and switches to tracking the background that was behind the query point on the initial frame, as shown in fig. 26. Both the optical flow estimation and the flow quality estimation get distracted by well-matched background (which may span most of the receptive field) and ignore the mismatch on the foreground object.

We have performed an experiment in which a segmentation tracker (SAM2 [151]) is initialized with the query point prompt and is left to track alongside the MFTIQ. Flow chain candidates that fall outside the segmentation mask predicted by the segmentation tracker are marked as occluded regardless of the MFTIQ occlusion prediction. This procedure improved the TAP-Vid DAVIS first-mode AJ score from 59.9 to 61.7, outperforming both the CoTracker and the BootsTAP (see table 10).

5 COIN-TRACKING

This chapter describes the coin-tracking task introduced in our paper [14], which is a continuation of work done in my masters thesis [13] and predates our work on planar object tracking (see chapter 3 and point-tracking (see chapter 4). The chapter is based on our paper [14], in which the coin-tracking task and dataset was first published on peer-reviewed conference. While the coin-tracking task itself was introduced in the masters thesis, here we have evaluated various statistics of the dataset to show its unique challenges and how it is distinct from the standard tracking benchmarks. We also propose a completely new baseline coin-tracking method, CTR-Base, and although it relies on currently rather old neural network models and is hand-crafted ad-hoc, we find that our best point-tracking method does not out-perform it (see section 5.6). The current state-of-the-art coin-tracker [152] outperforms the CTR-Base by 8%, but the dataset is nowhere near to be solved even in the simplest setting where the tracker is initialized by templates of both the front and the back side of the target. The coin-tracking is thus still an open challenge.

5.1 INTRODUCTION

Visual tracking is an active research field and performance of trackers improves significantly every year. This holds for bounding-box and segmentation trackers [153, 154, 155], for planar trackers [7, 150] and for point trackers [126, 129]. Nevertheless, a particular class of every-day objects remains challenging even for state-of-the-art methods, namely, rigid flat double-sided objects like cards, books, smartphones, magazines, coins[1], tools like knives, hand saws, sport equipment like table tennis rackets, paddles etc. Such objects often rotate fast producing unique challenges for trackers like fast incident light and aspect ratio change and rotational motion blur. The results of recent bounding-box level trackers on the PlanarTrack$_{BB}$ [6] benchmark show that the flat targets contained in PlanarTrack are more challenging to track than the general objects captured in standard tracking benchmarks [65, 66]. For example, the SwinTrack [156] tracking success score SUC$_{BB}$ drops from 0.840 on TrackingNet and 0.713 on LaSOT to just 0.663 on PlanarTrack$_{BB}$.

Tracking of double-sided objects introduces even more extreme object poses / views than usual in planar tracking datasets [6, 7, 5], making the tracking even harder. Also the extreme poses occur more frequently in coin-tracking sequences than in other datasets. For example the targets often *flip* from being seen from the front side to the back side and vice versa. Near the moment of flipping the target surface normals are often close to being perpendicular to the camera rays.

In this paper, we introduce an annotated *coin-tracking dataset*[2], CTR dataset in short, containing video sequences of coin-like objects. We then show that the proposed dataset is fundamentally different from the standard

[1] Hence the problem name.

[2] Available at http://cmp.felk.cvut.cz/coin-tracking.

ones [154, 157]. Finally, we propose a baseline coin-tracking method, called CTR-BASE, that outperforms classical state-of-the-art trackers in experiments on the CTR dataset.

5.2 THE COIN-TRACKING TASK

We define coin-tracking as tracking of rigid, approximately planar objects in video sequences. This means that at any time only one of the two sides - *obverse* (front) and *reverse* (back) - is visible. Unlike general objects, the rigidity and planarity of the coin-like objects means that the boundary between their two sides is always visible, except for occlusions by another object and position partially outside of the camera field of view. In this settings, the currently invisible side is fully occluded by the visible side and the visible side does not occlude itself at all. The state of a coin-like object is thus fully characterized by a visible side identification and a homography transformation to a canonical frame together with a possible partial occlusion mask.

However, because the objects in the CTR dataset are often symmetric, reflecting the real world coin-like object properties, the homography transformation might not be uniquely identifiable and thus we characterize the object state by a segmentation mask instead. Notice that unlike in standard general tracking sequences, where the exact extend of the tracked object is often not well defined due to the ambiguity of the initialization bounding box or segmentation, there is an unambiguous correspondence between a segmentation mask and a physical object in the case of coin-tracking.

Recent video object segmentation datasets [67, 68] represent the object pose by segmentation as well, nevertheless, they contain mostly outdoor sequences of animals, people and vehicles. Therefore, there is a significant domain gap between these datasets and the proposed coin-tracking problem. Other datasets for tracking planar object exist, such as [8, 5], but they only contain sequences with single side of the planar object visible. Moreover, in most cases the objects are fixed and the camera moves around them. This induces both different dynamics and appearance changes in the sequences as discussed in section 5.3.1.

The are multiple levels of tracking of coin-like objects. In the simplest form, level one, the tracker is initialized by a template of each side of the object and the object pose on the first frame of the sequence. Level two coin-tracker is initialized only on the first side of the target and has to discover the reverse side without any supervision. Level three requires a full 6D pose output, *i.e.* rotation and translation, together with a complete object surface reconstruction, including even the initially occluded parts of the object. In this work we focus on the level one coin-tracking task.

5.3 THE COIN-TRACKING DATASET

The introduced CTR dataset contains 17 video sequences of coin-like objects, with total of 9257 frames and segmentation ground truth masks on every fifth frame. See Fig. 27 for examples of the sequences in the CTR dataset.

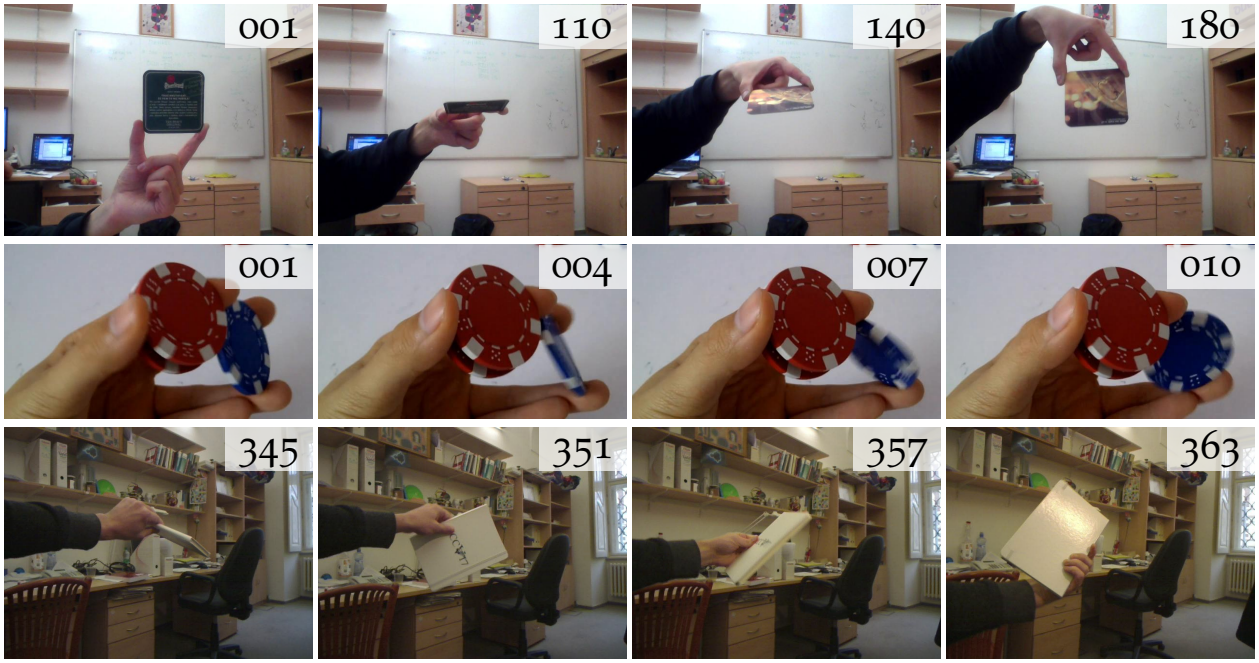


Figure 27: Examples from the coin-tracking dataset (frame number in the top-right corner). Notice the effects of the out-of-plane rotation – fast illumination change, blur and significant aspect ratio change of the objects.

5.3.1 *A Comparison with Other Datasets*

The main motivation for introducing a new tracking dataset is its difference from the currently available tracking sequences. In this section we show some of the novel aspects of the proposed dataset.

The planar object tracking datasets [5, 8] are the closest to the CTR dataset, but they only contain a single sided view of the object; the viewing angle range is limited. In most of the sequences the tracked object is fixed to the background behind it, e.g. a poster fixed on a wall and the object motion in the sequence is induced by the camera motion only. On the contrary, the camera is static or close to static in many of the CTR sequences and it is the object that causes the motion. This difference is important since the two situations introduce different challenges to the visual tracking task.

When a planar object is fixed and a camera moves around it, the perceived out-of-plane rotation is relatively slow as the camera needs to move along a long arc in order to change the viewing angle significantly. On the other hand, when the main part of the perceived motion of the object in the sequence is caused by the physical motion of the object itself, as it is the case in the proposed sequences, the object out-of-plane rotation happens faster as it is physically easy to rotate coin-like objects.

Most state-of-the-art trackers, e.g. the winners of the VOT2018 tracking challenge [154] – MFT [158] (name clash with our MFT dense point tracker from chapter 4) and UPDT [159], represent the object pose as axis-aligned or rotated bounding box, while the aspect ratio change modeling is not common. Later in this section, we show that both the range and the speed of aspect ratio change in the CTR sequences is higher than in the VOT [160]

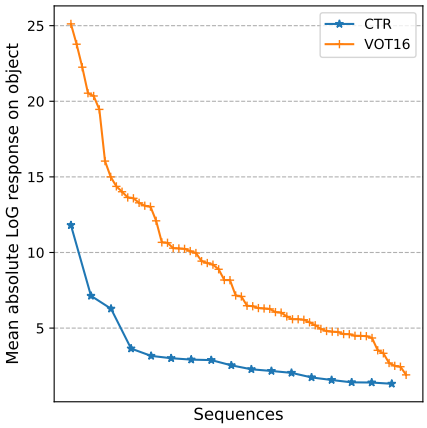


Figure 28: Comparison of object "textureness" in the proposed CTR and VOT 2016 datasets, measured by the absolute value of Laplacian of Gaussian $\sigma = 0.8$ averaged over the tracked object pixels.

and OTB [157] tracking datasets. Besides causing significant aspect ratio changes, the 3D rotation of the coin-like objects often induces fast changes of illumination as the object plane normal direction relative to the light sources changes rapidly. Apart from these differences, the objects in the CTR dataset are also less textured than the ones appearing in standard visual tracking datasets as discussed in the next section.

Textureness. As a measure of object textureness, we computed the Laplacian of Gaussian (LoG) responses and averaged their absolute values over the object pixels and all frames. Fig. 28 shows that the typical object textureness in the CTR dataset is significantly lower than on the VOT 2016 dataset [160]. The lack of texture prevents tracking to be implemented by classical methods for homography estimation based on key-point correspondences.

Aspect ratio change. One of the unique properties of the coin-tracking dataset is the presence of strong changes in object aspect ratios, not usually encountered in the standard visual tracking datasets as shown in the following two experiments. In order to compute the aspect ratio statistics, we first compute minimal (rotated) rectangle bounding the ground truth segmentation mask on each frame. The aspect ratio (25) of the resulting rectangle with sides a, b is defined as

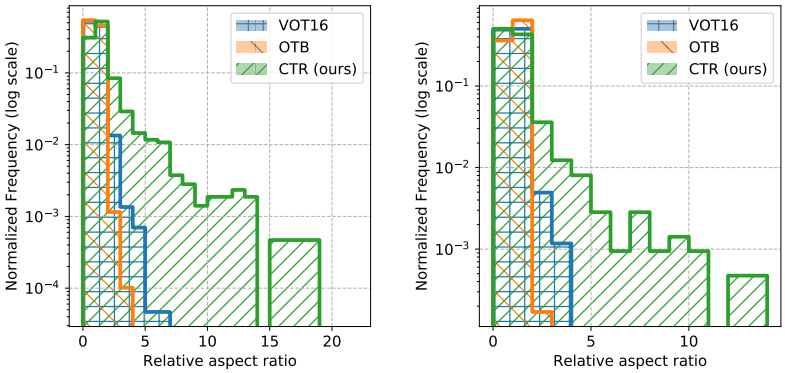
$$r(a,b) = \max\left(\frac{a}{b}, \frac{b}{a}\right) \tag{25}$$

We define the relative change in aspect ratios of two rectangles A, B with sides a_1, a_2 and b_1, b_2, respectively, as (26)

$$\Delta r(A,B) = \max\left(\frac{r(a_1,a_2)}{r(b_1,b_2)}, \frac{r(b_1,b_2)}{r(a_1,a_2)}\right) \tag{26}$$

The maximum of the two ratios is chosen because only the magnitude of the aspect ratio change matters.

ASPECT RATIO CHANGE RELATIVE TO THE FIRST FRAME. We have computed aspect ratio changes $\Delta r(R_1, R_t)$ between the bounding rectangle on the first frame and each of the other annotated frames in the sequence.



(a) relative to the first frame. (b) relative to frame t-5.

Figure 29: Histogram of aspect ratio changes

We then represent each tested dataset (VOT2016, OTB, CTR) by a histogram of these aspect ratio changes in all the dataset sequences as shown in Fig. 29a. Notice that although the VOT2016 and OTB datasets are not restricted to rigid objects, i.e. their segmentation masks can change shape arbitrarily during the sequences, the CTR dataset contains significantly bigger changes in the aspect ratios.

ASPECT RATIO CHANGE SPEED. In the proposed CTR dataset, the change in object aspect ratio is also faster than in the other compared datasets as shown in Fig. 29b. Instead of computing the aspect ratio change with respect to the first frame, the change is computed relative to the previous frame. Notice that because the CTR dataset does not contain ground truth segmentation masks on every frame, but only on every fifth, we measure $\Delta r(R_{t-5}, R_t)$ on all three datasets.

5.3.2 *Evaluation Metric*

We address the simplest form of the coin-tracking task, in which the tracker is initialized by an image of the front side of the tracked object on the first frame and an image of the back side later in the sequence, together with the respective ground truth segmentation masks.

We use *intersection over union* (IoU) as the evaluation metric – it is the standard metric for evaluating both segmentation and bounding box quality. In order to deal with frames with empty ground truth segmentation, i.e. with the object fully occluded or fully outside of the view, we augment the scoring function such that these frames do not contribute into the per-sequence total as proposed in [154].

5.4 THE BASELINE COIN-TRACKING METHOD

Standard trackers represent the object by a bounding box and are thus unable to capture the perspective transformations common for coin-like objects. Trackers based on key-point correspondences can estimate homographies, but the low textureness of CTR objects prevents their use. Convolutional neural networks recently used for video object segmentation, e.g. [57, 161, 162],

classify pixels as object or background taking into account large context thanks to large receptive fields of the neurons in the final layers. They do not consider the underlying homography transformations, but the segmentations capture the object extent in the image with high granularity.

Most video object segmentation methods use a deep neural network trained offline for general object segmentation. The network is then fine-tuned for tracking of a particular object at the initialization. One of the significant challenges in visual tracking is object appearance change and changes in the background in the video sequence. Because of this, trackers usually have to perform some kind of *online adaptation* to prevent performance deterioration soon after initialization. A simple adaptation scheme for video object segmentation has been proposed in OnAVOS [162], where the pixels classified as object with high confidence are treated as new object appearance examples. Background examples are taken from the parts of the image over a certain distance from the object. However, the online adaptation requires lengthy fine-tuning of the segmentation neural network on each frame, making the method slow.

An alternative approach has been proposed in FAST-VOS [163], where the segmentation is done by k-nearest neighbor search in an embedding space learned offline by a CNN. Instead of fine-tuning the embedding network on the first frame or later during online adaptation, the FAST-VOS method inserts dense embeddings into a k-NN classifier index. This makes the adaptation to a particular object faster and easier to interpret, compared to the network fine-tuning methods. The online adaptation proposed in [163] is similar to the original method in [162], selecting high confidence pixels – all of their $k = 5$ neighbors agree with the label – for the model update.

With all this in mind, we propose a baseline tracking method CTR-Base, which is based on the tracking-by-segmentation FAST-VOS [163] method. After an input frame is segmented using the k-NN classifier, we explicitly model the object pose and possibly perform online adaptation.

5.4.1 *Object Pose Estimation*

We have performed experiments with the adaptation scheme of FAST-VOS but it did not work well on the coin-tracking sequences. The adaptation has quickly drifted and led to a complete failure of the tracker, either segmenting almost all of the background as the object or vice versa. Our experiments with distance-threshold based background adaptation as in [162] as well as experiments with other heuristics based on analysis of the connected components and other properties of the segmentation mask were not successful either. We hypothesize that one of the reasons that those adaptation techniques work reasonably well on the DAVIS dataset, but fail on the coin-tracking task, might be the length of the sequences. The mean number of frames in the DAVIS 2017 sequences is only 69.7 [1] while the mean number of frames in the coin-tracking sequence in the CTR dataset is 544, with several sequences as long as 1000 frames. The robustness of the online adaptation scheme is crucial on sequences of such length.

In order to address the online adaptation in coin-tracking more robustly, we explicitly model the object pose using the homography to the ground-truth canonical frame. Both the object and the background pixel online adaptation is controlled by the agreement between the segmentation output by the k-NN classifier and the estimated pose model.

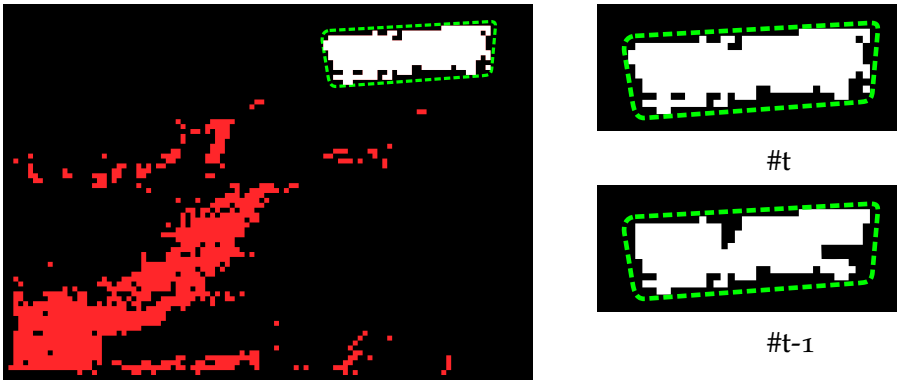


Figure 30: Homography score computation. Left: the segmentation mask split into pixels inside (white) the object pose hypothesis (dashed green) and the rest (red). Right: Object visibility mask for the current and the last frames.

5.4.1.1 *Objective Function.*

In each video frame, we search for the homography $\mathbf{H}_{*\to t}$ mapping the object on a ground truth frame into the current one, optimizing the objective function s, Eq. 29, composed of four parts computed as follows. First, we map the segmentation mask from the ground truth frame into the current frame using the homography. This splits the segmentation mask in the current frame into two parts, one inside and the other one outside of the hypothesized object contour as shown in Fig. 30. The s_{obj} part of the score function is set to the fraction of the segmentation mask located inside the contour, indicating the fraction of the segmentation explained by the object. This part of the score function penalizes segmentation outside of the object with the pose given by $\mathbf{H}_{*\to t}$.

The s_{cover} part of the score function s is the fraction of the pixels inside the hypothesized object contour being classified as the object. This part penalizes homographies mapping the object contour such that it is not well covered by the segmentation. Notice, however, that in the case of partial occlusion by other object, the segmentation should not cover the whole object. Since the occlusion mask is changing relatively slowly in CTR sequences, the s_{occl} component of the score function s is the IoU overlap of the current and last visibility mask, which is transformed to the current frame by $\mathbf{H}_{t-1\to t} = \mathbf{H}_{*\to t}\mathbf{H}^{-1}_{*\to t-1}$. This prefers homographies with a small occlusion change with respect to the previous frame.

Finally, the appearance score $s_{\text{appearance}}$ is the zero-offset coefficient of the zero-normalized cross-correlation (ZNCC) score

$$s_{\text{appearance}} = \frac{1}{2} + \frac{\sum\limits_{x,y\in O}(I_t(x,y)-\mu(I_t))(I_*(x,y)-\mu(I_*))}{2\sqrt{\sum\limits_{x,y\in O}(I_t(x,y)-\mu(I_t))^2 \sum\limits_{x,y\in O}(I_*(x,y)-\mu(I_*))^2}} \tag{27}$$

of the object image in the current frame and the template from the ground-truth frame, where $I_t(x,y)$ and $I_*(x,y)$ are the image values at coordinates

$[x, y]$ in the current frame and the ground truth frame projected using the homography $\mathbf{H}_{*\to t}$ respectively and

$$\mu(I) = \frac{1}{|O|} \sum_{x,y \in O} I(x, y) \tag{28}$$

with O being the set of points segmented as object in both the ground truth and the current frame. The rationale behind introducing the appearance score is that it helps distinguishing a correct homography in case of objects with symmetric shape or partial occlusions. The final score, Eq. 29, of the homography is the product of these four components giving a number in 0-1 range:

$$s = s_{obj} \cdot s_{cover} \cdot s_{occl} \cdot s_{appearance} \tag{29}$$

Notice that compared to summing the score components, taking their product highlights drops in any of the score components and thus it is preferable for making our adaptation method conservative.

5.4.1.2 *Optimization.*

Since the cost function described above is not differentiable, we use a probabilistic optimization procedure based on simulated annealing for finding $\mathbf{H}_{*\to t}$ for each frame. The optimization is initialized using either the homography found in the previous frame or using optical flow from the previous frame, in which case we uniformly sample 4 points from inside the object and transform them by the flow field to get 4 correspondences necessary for estimating the inter-frame homography. This is repeated 50 times and the $\mathbf{H}_{*\to t}$ maximizing the score function is chosen as the initialization of the following iterative optimization procedure.

In each step of the optimization a random homography matrix is sampled by randomly perturbing 4 control points at the corners of the object bounding box and computing the homography from the resulting 4 correspondences. Next, the homography score s is computed and compared to the current best score, s^*. The $\mathbf{H}_{*\to t}$ hypothesis is accepted as the current estimate of the optimum with probability

$$p(s, s^*, T) = \begin{cases} 1 & \text{if } s > s^*, \\ e^{-\frac{s^* - s}{T}} & \text{otherwise,} \end{cases} \tag{30}$$

where the T is decreasing in each iteration, allowing jumps from local minima but with decreasing probability during the optimization procedure. We also decrease the control point perturbation σ in each of the 350 iterations.

Depending on the ratio of pixels being classified as belonging to the obverse or the reverse side of the object, the optimization procedure is run against the respective ground truth frame. Finally, when the score of the best found homography is low, the tracker switches into a *lost* state and stays in it until a successful re-detection of the object.

The re-detection procedure is the same as the optimization described above, except for spending more time (400 iterations) sampling for the initialization pose and not using the information from the previous frame. The previous visibility mask used in computation of s_{occl} is replaced by the full object mask.

5.4.2 *Online Adaptation*

The proposed homography optimization procedure reduces the overall speed of the tracker, but we have observed that it finds a good solution reliably, unless the segmentation is grossly incorrect, enabling us to use online-adaptation on the long sequences in the CTR dataset. In particular, no online adaptation is attempted when the tracker is in the *lost* state, reducing the probability of making incorrect adaptation.

If the tracker is in the *tracking* state, new background and object embedding examples are added into the segmentation k-NN classifier. To stay on the safe side, only the pixels that are far from the object boundary and were incorrectly classified (with respect to the hypothesized object pose) are used as new background examples. Moreover, these pixels must not be connected to the object by the segmentation mask, otherwise they are not used for adaptation even if they are very far from the image.

For the new object examples, we select the pixels classified as background by the segmentation k-NN classifier that are not connected to the object edges, in other words only closed 'holes' in the object segmentation are adapted.

Altogether, the proposed online adaptation technique allows for conservative online adaptation, not making severe mistakes that would lead to complete failure of the tracker, as shown in the experiments in section 5.5.2.

5.4.3 *Implementation details*

We use a DeepLabv3+ [164] segmentation head on top of MobileNetv1 [165] backbone architecture. The MobileNet backbone was pretrained[3] on ImageNet [142], then trained for semantic segmentation on PASCAL VOC 2012 [166] enriched by the *trainaug* augmentations by [167]. We have used the Adam [168] optimizer with batch size 5 and initial learning rate of 7×10^{-4} decaying to 10^{-6} according to the *poly* schedule with decay power 0.9 for 53000 iterations. Finally, using the augmented triplet loss proposed by [163], we have fine-tuned the network for 492000 iterations on the YouTubeVOS dataset [68] to output dense 128-dimensional embeddings useful for segmentation by k-NN classifier. Given an $H \times W$ image, the network produces a per-pixel 128-D embeddings with output stride 4 (resolution $\frac{H}{4} \times \frac{W}{4}$). We use FAISS [169] library[4] with a flat L2 index for speeding up the nearest neighbor searches used in the segmentation. For the optical flow computation, we use ContinualFlow [23].

The method runs at around 7 seconds per frame at 1280×720 resolution with the majority of time spent optimizing the pose. The runtime drops without losing much performance when the pose optimization is done on lower resolution.

5.5 EXPERIMENTS

In this section we show that the proposed CTR-BASE method outperforms general state-of-the-art trackers on the CTR dataset and retains good performance on the POT-210 [8] dataset. Then we demonstrate that the homography-

[3] Code and weights available at https://github.com/tensorflow/models/

[4] Available at https://github.com/facebookresearch/faiss

based pose modeling prevents the CTR-Base tracker from making fatal mistakes.

5.5.1 *Baseline Experiment*

In the standard visual tracking formulation, the tracker is initialized by the ground truth object pose, which can be represented by axis-aligned bounding box, rotated bounding box or segmentation mask [154, 67, 68]. This means that standard state-of-the-art trackers cannot be directly evaluated on the coin-tracking task in which the tracker is initialized on one frame from each side of the object. On the other hand, the coin-tracking task can be viewed as a long-term tracking on single side, enabling us to evaluate state-of-the-art long term trackers MBMD [170] and DaSiam_LT [171] – the winners of the VOT 2018 [154] long-term tracking challenge on the CTR dataset. Moreover, the VOT long-term tracking challenge requires a tracker confidence output on each frame, which allows us to run each tracker two times - once initialized from the obverse and once from the reverse side, merging the results by picking the one with higher tracker confidence. We have represented the axis-aligned bounding box outputs of the long-term trackers as segmentation masks and evaluated using the IoU metric. The results are shown in Tab. 13.

The proposed CTR-Base method significantly outperforms both state-of-the-art bounding box trackers and a bounding box oracle, which outputs the bounding boxes of the ground truth segmentation masks. Computing IoU from the bounding boxes might not seem fair, but the performance gap demonstrates the need of representing the tracked object by segmentation, even with relatively compact objects present in the CTR dataset.

In order to further test the CTR-Base method, we evaluated it on the POT-210 [8] dataset, converting the ground – object corners – to segmentation (not modeling occlusions). The mean IoU (mIoU) is 0.81, showing that our method generalizes to POT-210 well. The best results were achieved on the *out-of-view* and the *perspective distortion* subsets of [8] with mIoU 0.89 and 0.88 respectively, while the worst on the *motion blur* subset with mIoU of 0.71.

5.5.2 *Results on confident frames*

The mean IoU score computed only on the frames where the CTR-Base method is in the *tracking* state, i.e. online adaptation is allowed, improves from 0.70 to 0.88. This shows that the proposed tracker can correctly detect its own failures and only adapt when tracking reliably. Overall the tracker spends 47% of the frames in the *tracking* state as shown in Tab. 14.

5.6 COIN-TRACKING USING THE MFTIQ POINT TRACKER

In this section we take the dense long-term tracker described in chapter 4 and evaluate it on the coin-tracking task. To convert the point-tracking to the target segmentation masks needed for coin-tracking evaluation, we used two approaches. First, taking the initial target mask and forward-warping it directly with the flow field output of the MFTIQ. Second, computing a homography from the flow field and using that to warp the initial mask. We estimate the homographies with RANSAC, which should ignore grossly incorrect flow correspondences (outliers) and average out small inaccuracies

sequence	MBMD	DaSiam_LT	bbox oracle	CTR-Base (ours)
beermat	0.70	0.18	0.78	**0.83**
card1	0.72	0.71	0.73	**0.79**
card2	0.71	0.68	0.79	**0.93**
coin1	0.60	0.62	0.71	**0.80**
coin3	0.32	**0.46**	0.63	0.38
coin4	0.33	0.41	0.56	**0.65**
husa	0.35	0.40	0.51	**0.73**
iccv_bg_handheld	0.27	0.31	0.54	**0.33**
iccv_handheld	0.32	0.39	0.55	**0.50**
iccv_simple_static	0.37	0.31	0.51	**0.65**
iccv_static	0.34	0.40	0.55	**0.67**
pingpong1	**0.42**	0.38	0.64	0.33
plain	0.44	0.50	0.60	**0.74**
statnice	0.53	0.57	0.67	**0.87**
tatra	0.47	0.54	0.66	**0.86**
tea_diff_2	0.54	0.57	0.61	**0.87**
tea_same	0.53	0.52	0.63	**0.85**
Mean over all frames	0.47	0.44	0.63	**0.70**

Table 13: The evaluation of the IoU overlap metric on the proposed CTR dataset. Notice that the CTR-Base method outperforms both state-of-the-art long-term trackers and the bounding box oracle.

sequence	beermat	card1	card2	coin1	coin3	coin4	husa	iccv_bg_handheld	iccv_handheld	iccv_simple_static	iccv_static	pingpong1	plain	statnice	tatra	tea_diff_2	tea_same	**average**
IoU ×100	89	89	96	82	94	84	87	90	85	85	83	67	88	89	92	92	86	88
frames in *tracking* state %	89	68	93	64	02	21	69	17	15	29	28	17	42	46	34	87	47	47

Table 14: The IoU score of the CTR-Base tracker evaluated only on the frames, where it is in the confident *tracking* state and the online adaptation is enabled. Notice that indeed the tracker is confident on the frames, where it performs well.

on inliers. We evaluate the RoMa[43]-based MFTIQ, which performs the best on the point-tracking task. The results are shown in table 15. Apart from few sequences where the RoMa-based correspondences help, namely *card1*, *coin1*, and *beermat*, the MFTIQ performance is poor. This may be due to the targets being usually quite small, non-trivial amount of complicated (induced by in- and out-of-plane rotations) motion blur, and overall low quality of the CTR dataset videos. RoMa was trained on the MegaDepth [49] dataset of landmarks where the main object usually occupies most of the image and there is very little or no blur.

method	beermat	card1	card2	coin1	coin3	coin4	husa	iccv_bg_handheld	iccv_handheld	iccv_simple_static	iccv_static	pingpong1	plain	statnice	tatra	tea_diff_2	tea_same	**average**
CTRBASE	83	79	**93**	80	**38**	**65**	**73**	33	**50**	**65**	**67**	**33**	**74**	**87**	**86**	87	**85**	**70**
MFTIQ-WARP	87	87	91	**88**	15	38	58	28	34	33	25	21	22	46	52	78	47	49
MFTIQ-HOMO	**90**	**90**	91	84	31	52	67	**43**	**50**	44	40	32	54	71	65	**89**	59	63

Table 15: Coin-Tracking with RoMa-based MFTIQ compared to the CTRBASE baseline in the segmentation mask mean IoU score. The MFTIQ point tracker tracks well on relatively simple sequences, like *beermat*, or *card1* where the motions are slow and the object is big and well textured. However, overall it did not achieve the baseline performance on the whole dataset.

6 CONCLUSIONS

In this thesis we have studied the problem of dense long-term visual tracking. Optical flow gives dense correspondences, but only between pairs of consecutive frames. We have adapted it to long-term tracking of planar objects and also to long-term dense point-tracking in arbitrary scenes.

The combination of frame pre-warping, optical flow correspondences and a differentiable homography estimation neural network introduced in our planar tracker WOFT results in state-of-the-art performance on multiple standard planar object tracking benchmarks. On the PlanarTrack [6] dataset (which was published after the WOFT [9] tracker) and on the POT [8, 7] datasets, WOFT out-performs other trackers by a large margin.

The work on dense point-tracking in arbitrary scenes resulted in two trackers based on optical flow, MFT and MFTIQ. Both work by chaining optical flows estimated over varying number of frames, which allows them to "jump over" temporary occlusions, blurred frames and other difficult cases. The final prediction is formed by selecting the most reliable from a small number of such flow chain candidates in each pixel. The most reliable flow chain selection from MFT was improved in MFTIQ to achieve point-tracking performance close to the current state-of-the-art. In contrast to other point trackers, our trackers are causal and track densely (every point in the reference frame) effectively. We have also tested the MFTIQ point tracker on the planar object tracking task and it sets a new state-of-the-art performance, slightly outperforming WOFT. The WOFT is still our preferred planar tracker, because the accuracy gains of MFTIQ are not significant enough to justify the lower tracking speed. The results however show that MFTIQ generalizes well to challenging videos atypical for the point-tracking community.

During our research, we have identified issues with annotation quality in standard planar object tracking POT benchmarks. We have precisely re-annotated the ground truth on a uniformly spaced subset of frames and published it. In the planar tracking benchmark, the original ground-truth annotation errors accounted for half of the benchmark error of the top trackers. Thus the re-annotation should help the planar tracking community to benchmark the trackers more accurately and to prevent over-fitting to the incorrect ground truth.

Finally, we have published the coin-tracking problem, significantly extending the work started in my masters thesis. The coin-tracking is a special version of the planar object tracking task, in which thin planar rigid objects are tracked from both front and back side. This poses unique challenges. We have selected 17 coin-tracking videos and manually annotated them with segmentation masks, and published the resulting CTR dataset. We have also shown its dissimilarity to standard tracking datasets.

We have proposed a new CTR-Base coin-tracking method that enables robust online adaptation through explicit modeling of the target pose and through failure detection. It outperforms our best MFTIQ point tracker, showing the difficulty of the coin-tracking task and the CTR dataset. The advanced variants of the coin-tracking task described in section 5.2, like the

unsupervised back side discovery or full surface reconstruction, are even more challenging topics left for future research.

We have seen significant advances in many areas of computer vision, with performance of CNN- and vision transformer-based methods improving constantly. New self-supervised approaches combined with huge amount of training data and vast computational power resulted in foundation models that provide off-the-shelf visual features that perform well on various tasks. However, even for these state-of-the-art models, the performance drops fast when applied to low-quality videos, non-standard difficult tasks, under presence of non-trivial amount of motion blur, lack of texture, presence of repetitive structures and small thin objects. For example, the currently used neural networks have a tendency to unpredictably mix the information coming from the particular pixel and its neighborhood on the same object with the information coming from the background and other objects, making it impossible to reliably deal with thin objects. Future research in this direction has potential to greatly improve dense long-term tracking.

Another interesting open research topic is tracking of *all points in a video*, not just all points on a single reference frame. This is addressed by OmniMotion [132] and follow-up works [134, 135], but all of them are computationally demanding and slow test-time optimization techniques, in which the tracker has to be trained for the particular video. Designing an efficient tracker of all points is an important next step for the point-tracking community.

Also we think computer vision researchers should pay more attention to the quality of the benchmarks, ensuring high-quality annotations and/or knowing their accuracy and using meaningful metrics. Chasing tiny performance improvements on saturated or low quality benchmarks leads to manual over-fitting and does not help to advance knowledge.

BIBLIOGRAPHY

[1] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. V. Gool, "The 2017 davis challenge on video object segmentation," *arXiv preprint arXiv:1704.00675v2*, 2017.

[2] B. Lin, Y. Sun, X. Qian, D. Goldgof, R. Gitlin, and Y. You, "Video-based 3d reconstruction, laparoscope localization and deformation recovery for abdominal minimally invasive surgery: a survey," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 12, no. 2, pp. 158–178, 2016.

[3] G. Wei, G. Feng, H. Li, T. Chen, W. Shi, and Z. Jiang, "A novel slam method for laparoscopic scene reconstruction with feature patch tracking," in *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 287–291, IEEE, 2020.

[4] L. Maier-Hein, P. Mountney, A. Bartoli, H. Elhawary, D. Elson, A. Groch, A. Kolb, M. Rodrigues, J. Sorger, S. Speidel, *et al.*, "Optical techniques for 3d surface reconstruction in computer-assisted laparoscopic surgery," *Medical image analysis*, vol. 17, no. 8, pp. 974–996, 2013.

[5] L. Chen, H. Ling, Y. Shen, F. Zhou, P. Wang, X. Tian, and Y. Chen, "Robust visual tracking for planar objects using gradient orientation pyramid," *Journal of Electronic Imaging*, vol. 28, no. 1, pp. 1–16, 2019.

[6] X. Liu, X. Liu, Z. Yi, X. Zhou, T. Le, L. Zhang, Y. Huang, Q. Yang, and H. Fan, "PlanarTrack: A large-scale challenging benchmark for planar object tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20449–20458, October 2023.

[7] P. Liang, H. Ji, Y. Wu, Y. Chai, L. Wang, C. Liao, and H. Ling, "Planar object tracking benchmark in the wild," *Neurocomputing*, vol. 454, pp. 254–267, 2021.

[8] P. Liang, Y. Wu, H. Lu, L. Wang, C. Liao, and H. Ling, "Planar object tracking in the wild: A benchmark," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 651–658, IEEE, 2018.

[9] J. Šerých and J. Matas, "Planar object tracking via weighted optical flow," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1593–1602, 2023.

[10] M. Neoral, J. Šerých, and J. Matas, "MFT: Long-term tracking of every pixel," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6837–6847, 2024.

[11] T. Jelínek, J. Šerých, and J. Matas, "Dense matchers for dense tracking," in *Proceedings of the 27th Computer Vision Winter Workshop (CVWW 2024)*, 2024.

[12] J. Šerých, M. Neoral, and J. Matas, "MFTIQ: Multi-flow tracker with independent matching quality estimation," *under review*, 2025.

[13] J. Šerých and J. Matas, "Coin-tracking – double-sided tracking of flat objects," Master's thesis, Czech Technical University in Prague, Jan 2018.

[14] J. Šerých and J. Matas, "Visual coin-tracking: Tracking of planar double-sided objects," in *German Conference on Pattern Recognition*, pp. 317–330, Springer, 2019.

[15] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[16] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2*, pp. 674–679, 1981.

[17] Q. Dong and Y. Fu, "MemFlow: Optical flow estimation and prediction with memory," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19068–19078, 2024.

[18] X. Shi, Z. Huang, W. Bian, D. Li, M. Zhang, K. C. Cheung, S. See, H. Qin, J. Dai, and H. Li, "Videoflow: Exploiting temporal cues for multi-frame optical flow estimation," *arXiv preprint arXiv:2303.08340*, 2023.

[19] S. Jiang, D. Campbell, Y. Lu, H. Li, and R. Hartley, "Learning to estimate hidden motions with global motion aggregation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9772–9781, 2021.

[20] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

[21] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8934–8943, 2018.

[22] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*, pp. 402–419, Springer, 2020.

[23] M. Neoral, J. Šochman, and J. Matas, "Continual occlusion and optical flow estimation," in *Asian Conference on Computer Vision*, pp. 159–174, Springer, 2018.

[24] M. Luz, R. Mohan, A. R. Sekkat, O. Sawade, E. Matthes, T. Brox, and A. Valada, "Amodal optical flow," *arXiv preprint arXiv:2311.07761*, 2023.

[25] R. Ranftl, K. Bredies, and T. Pock, "Non-local total generalized variation for optical flow estimation," in *European conference on computer vision*, pp. 439–454, Springer, 2014.

[26] J. Hur and S. Roth, "Iterative residual refinement for joint optical flow and occlusion estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5754–5763, 2019.

[27] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "Flowformer: A transformer architecture for optical flow," *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pp. 668–685, 2022.

[28] X. Shi, Z. Huang, D. Li, M. Zhang, K. C. Cheung, S. See, H. Qin, J. Dai, and H. Li, "FlowFormer++: Masked cost volume autoencoding for pretraining optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1599–1610, 2023.

[29] A. Jahedi, M. Luz, L. Mehl, M. Rivinius, and A. Bruhn, "High resolution multi-scale RAFT (robust vision challenge 2022)," *arXiv preprint arXiv:2210.16900*, 2022.

[30] A. Jahedi, L. Mehl, M. Rivinius, and A. Bruhn, "Multi-scale RAFT: Combining hierarchical concepts for learning-based optical flow estimation," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 1236–1240, IEEE, 2022.

[31] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4040–4048, 2016.

[32] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black, "Lessons and insights from creating a synthetic optical flow benchmark," in *Computer Vision–ECCV 2012. Workshops and Demonstrations: Florence, Italy, October 7-13, 2012, Proceedings, Part II 12*, pp. 168–177, Springer, 2012.

[33] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pp. 611–625, Springer, 2012.

[34] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[35] L. Mehl, J. Schmalfuss, A. Jahedi, Y. Nalivayko, and A. Bruhn, "Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo," *arXiv preprint arXiv:2303.01943*, 2023.

[36] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Gussefeld, M. Rahimimoghaddam, S. Hofmann, C. Brenner, *et al.*, "The HCI benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 19–28, 2016.

[37] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3061–3070, 2015.

[38] S. Zhao, Y. Sheng, Y. Dong, E. I. Chang, Y. Xu, *et al.*, "Maskflownet: Asymmetric feature matching with learnable occlusion mask," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6278–6287, 2020.

[39] C. Harris, M. Stephens, *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[40] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[41] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 224–236, 2018.

[42] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[43] J. Edstedt, Q. Sun, G. Bökman, M. Wadenbäck, and M. Felsberg, "RoMa: Revisiting robust losses for dense feature matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19790–19800, 2024.

[44] J. Edstedt, I. Athanasiadis, M. Wadenbäck, and M. Felsberg, "DKM: Dense kernelized feature matching for geometry estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17765–17775, 2023.

[45] D. Brüggemann, C. Sakaridis, P. Truong, and L. Van Gool, "Refign: Align and refine for adaptation of semantic segmentation to adverse conditions," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3174–3184, 2023.

[46] P. Truong, M. Danelljan, F. Yu, and L. Van Gool, "Probabilistic warp consistency for weakly-supervised semantic correspondences," *arXiv preprint arXiv:2203.04279*, 2022.

[47] P. Truong, M. Danelljan, R. Timofte, and L. Van Gool, "PDC-Net+: Enhanced probabilistic dense correspondence network," *arXiv preprint arXiv:2109.13912*, 2021.

[48] W. Jiang, E. Trulls, J. Hosang, A. Tagliasacchi, and K. M. Yi, "COTR: Correspondence transformer for matching across images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6207–6217, 2021.

[49] Z. Li and N. Snavely, "MegaDepth: Learning single-view depth prediction from internet photos," in *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[50] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pp. 501–518, Springer, 2016.

[51] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113, 2016.

[52] P. Truong, M. Danelljan, F. Yu, and L. Van Gool, "Warp consistency for unsupervised learning of dense correspondences," *arXiv preprint arXiv:2104.03308*, 2021.

[53] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 2544–2550, IEEE, 2010.

[54] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[55] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "Atom: Accurate tracking by overlap maximization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4660–4669, 2019.

[56] A. Lukezic, J. Matas, and M. Kristan, "D3S – a discriminative single shot segmentation tracker," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7133–7142, 2020.

[57] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-shot video object segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 221–230, 2017.

[58] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1328–1338, 2019.

[59] Z. Yang and Y. Yang, "Decoupling features in hierarchical propagation for video object segmentation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36324–36336, 2022.

[60] X. Chen, H. Peng, D. Wang, H. Lu, and H. Hu, "SeqTrack: Sequence to sequence learning for visual object tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14572–14581, 2023.

[61] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, H. J. Chang, M. Danelljan, L. Cehovin, A. Lukežič, O. Drbohlav, J. Käpylä, G. Häger, S. Yan, J. Yang, Z. Zhang, and G. Fernández, "The ninth visual object tracking vot2021 challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 2711–2738, October 2021.

[62] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[63] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," *arXiv preprint arXiv:1902.03604*, 2019.

[64] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2137–2155, Nov 2016.

[65] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "Trackingnet: A large-scale dataset and benchmark for object tracking in the wild," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 300–317, 2018.

[66] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling, "Lasot: A high-quality benchmark for large-scale single object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5374–5383, 2019.

[67] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Computer Vision and Pattern Recognition*, 2016.

[68] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang, "Youtube-vos: A large-scale video object segmentation benchmark," *arXiv preprint arXiv:1809.03327*, 2018.

[69] H. Ding, C. Liu, S. He, X. Jiang, P. H. Torr, and S. Bai, "Mose: A new dataset for video object segmentation in complex scenes," *arXiv preprint arXiv:2302.01872*, 2023.

[70] K. Zhang, J. Chen, and B. Jia, "Asymptotic moving object tracking with trajectory tracking extension: A homography-based approach," *International Journal of Robust and Nonlinear Control*, vol. 27, no. 18, pp. 4664–4685, 2017.

[71] S. Benhimane and E. Malis, "Homography-based 2d visual tracking and servoing," *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.

[72] F. Sun, X. Sun, B. Guan, T. Li, C. Sun, and Y. Liu, "Planar homography based monocular slam initialization method," in *Proceedings of the 2019 2nd International Conference on Service Robotics Technologies*, pp. 48–52, 2019.

[73] J. Valognes, N. S. Dastjerdi, and M. Amer, "Augmenting reality of tracked video objects using homography and keypoints," in *International Conference on Image Analysis and Recognition*, pp. 237–245, Springer, 2019.

[74] C. Pirchheim and G. Reitmayr, "Homography-based planar mapping and tracking for mobile phones," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 27–36, IEEE, 2011.

[75] G. Simon, A. W. Fitzgibbon, and A. Zisserman, "Markerless tracking using planar structures in the scene," in *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*, pp. 120–128, IEEE, 2000.

[76] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[77] K. Lebeda, J. Matas, and O. Chum, "Fixing the locally optimized ransac–full experimental evaluation," in *British machine vision conference*, vol. 2, Citeseer, 2012.

[78] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Joint Pattern Recognition Symposium*, pp. 236–243, Springer, 2003.

[79] O. Chum and J. Matas, "Matching with prosac-progressive sample consensus," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 220–226, IEEE, 2005.

[80] D. Barath, J. Noskova, M. Ivashechkin, and J. Matas, "MAGSAC++, a fast, reliable and accurate robust estimator," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1304–1312, 2020.

[81] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, J.-K. Kämäräinen, M. Danelljan, L. Č. Zajc, A. Lukežič, O. Drbohlav, *et al.*, "The eighth visual object tracking vot2020 challenge results," in *European Conference on Computer Vision*, pp. 547–601, Springer, 2020.

[82] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[83] S. Hare, A. Saffari, and P. H. Torr, "Efficient online structured output learning for keypoint-based object tracking," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1894–1901, IEEE, 2012.

[84] T. Wang and H. Ling, "Gracker: A graph-based planar object tracker," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1494–1501, 2017.

[85] D. Matveichev and D.-T. Lin, "Mobile augmented reality: Fast, precise, and smooth planar object tracking," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 6406–6412, IEEE, 2021.

[86] Y. Liu, Z. Shen, Z. Lin, S. Peng, H. Bao, and X. Zhou, "Gift: Learning transformation-invariant dense visual descriptors via group cnns," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[87] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3279–3286, 2015.

[88] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, "SOSNet: Second order similarity regularization for local descriptor learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11016–11025, 2019.

[89] R. Pautrat, V. Larsson, M. R. Oswald, and M. Pollefeys, "Online invariance selection for local feature descriptors," in *European Conference on Computer Vision*, pp. 707–724, Springer, 2020.

[90] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[91] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1, pp. 943–948, IEEE, 2004.

[92] L. Chen, F. Zhou, Y. Shen, X. Tian, H. Ling, and Y. Chen, "Illumination insensitive efficient second-order minimization for planar object tracking," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4429–4436, IEEE, 2017.

[93] L. Chen, Y. Chen, H. Ling, X. Tian, and Y. Tian, "Learning robust features for planar object tracking," *IEEE Access*, vol. 7, pp. 90398–90411, 2019.

[94] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[95] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[96] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.

[97] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar, "Unsupervised deep homography: A fast and robust homography estimation model," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2346–2353, 2018.

[98] I. Rocco, R. Arandjelovic, and J. Sivic, "Convolutional neural network architecture for geometric matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6148–6157, 2017.

[99] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[100] X. Zhan, Y. Liu, J. Zhu, and Y. Li, "Homography decomposition networks for planar object tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 3234–3242, 2022.

[101] R. Zeng, S. Denman, S. Sridharan, and C. Fookes, "Rethinking planar homography estimation using perspective fields," in *Asian Conference on Computer Vision*, pp. 571–586, Springer, 2018.

[102] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[103] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

[104] T.-W. Hui, X. Tang, and C. C. Loy, "A lightweight optical flow cnn—revisiting data fidelity and regularization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 8, pp. 2555–2569, 2020.

[105] M. Contributors, "MMFlow: Openmmlab optical flow toolbox and benchmark." https://github.com/open-mmlab/mmflow, 2021.

[106] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4938–4947, 2020.

[107] H. Alismail, B. Browning, and S. Lucey, "Robust tracking in low light and sudden illumination changes," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 389–398, IEEE, 2016.

[108] T. Crivelli, P.-H. Conze, P. Robert, and P. Pérez, "From optical flow to dense long term correspondences," in *2012 19th IEEE International Conference on Image Processing*, pp. 61–64, IEEE, 2012.

[109] T. Crivelli, M. Fradet, P.-H. Conze, P. Robert, and P. Pérez, "Robust optical flow integration," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 484–498, 2014.

[110] P.-H. Conze, P. Robert, T. Crivelli, and L. Morin, "Dense long-term motion estimation via statistical multi-step flow," in *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 3, pp. 545–554, IEEE, 2014.

[111] P.-H. Conze, P. Robert, T. Crivelli, and L. Morin, "Multi-reference combinatorial strategy towards longer long-term dense motion estimation," *Computer Vision and Image Understanding*, vol. 150, pp. 66–80, 2016.

[112] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman, "TAPIR: Tracking any point with per-frame initialization and temporal refinement," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10061–10072, October 2023.

[113] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht, "CoTracker: It is better to track together," *arXiv preprint arXiv:2307.07635*, 2023.

[114] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 500–513, 2010.

[115] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International journal of computer vision*, vol. 103, pp. 60–79, 2013.

[116] Y. Liu, J. Shen, W. Wang, H. Sun, and L. Shao, "Better dense trajectories by motion in videos," *IEEE transactions on cybernetics*, vol. 49, no. 1, pp. 159–170, 2017.

[117] Z. Ren, O. Gallo, D. Sun, M.-H. Yang, E. B. Sudderth, and J. Kautz, "A fusion approach for multi-frame optical flow estimation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2077–2086, IEEE, 2019.

[118] T. Crivelli, P.-H. Conze, P. Robert, M. Fradet, and P. Pérez, "Multi-step flow fusion: Towards accurate and dense correspondences in long video shots," in *British Machine Vision Conference*, 2012.

[119] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, "Uncertainty estimates and multi-hypotheses networks for optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 652–667, 2018.

[120] S. Zhao, Y. Sheng, Y. Dong, E. I. Chang, Y. Xu, *et al.*, "MaskFlownet: Asymmetric feature matching with learnable occlusion mask," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6278–6287, 2020.

[121] S. Liu, K. Luo, N. Ye, C. Wang, J. Wang, and B. Zeng, "OIFlow: Occlusion-inpainting optical flow estimation by unsupervised learning," *IEEE Transactions on Image Processing*, vol. 30, pp. 6420–6433, 2021.

[122] C. Zhang, C. Feng, Z. Chen, W. Hu, and M. Li, "Parallel multiscale context-based edge-preserving optical flow estimation with occlusion detection," *Signal Processing: Image Communication*, vol. 101, p. 116560, 2022.

[123] A. S. Wannenwetsch, M. Keuper, and S. Roth, "ProbFlow: Joint optical flow and uncertainty estimation," in *Proceedings of the IEEE international conference on computer vision*, pp. 1173–1182, 2017.

[124] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, "Uncertainty estimates and multi-hypotheses networks for optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 652–667, 2018.

[125] G. Yang and D. Ramanan, "Volumetric correspondence networks for optical flow," in *Advances in neural information processing systems*, pp. 794–805, 2019.

[126] C. Doersch, A. Gupta, L. Markeeva, A. R. Continente, L. Smaira, Y. Aytar, J. Carreira, A. Zisserman, and Y. Yang, "TAP-Vid: A benchmark for tracking any point in a video," *Advances in Neural Information Processing Systems*, 2022.

[127] A. W. Harley, Z. Fang, and K. Fragkiadaki, "Particle video revisited: Tracking through occlusions using point trajectories," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pp. 59–75, Springer, 2022.

[128] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," *International Journal of Computer Vision*, vol. 80, pp. 72–91, 2008.

[129] C. Doersch, Y. Yang, D. Gokay, P. Luc, S. Koppula, A. Gupta, J. Heyward, R. Goroshin, J. Carreira, and A. Zisserman, "BootsTAP: Bootstrapped training for tracking-any-point," 2024.

[130] Y. Xiao, Q. Wang, S. Zhang, N. Xue, S. Peng, Y. Shen, and X. Zhou, "SpatialTracker: Tracking any 2d pixels in 3d space," *arXiv preprint arXiv:2404.04319*, 2024.

[131] G. Le Moing, J. Ponce, and C. Schmid, "Dense optical tracking: connecting the dots," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19187–19197, 2024.

[132] Q. Wang, Y.-Y. Chang, R. Cai, Z. Li, B. Hariharan, A. Holynski, and N. Snavely, "Tracking everything everywhere all at once," *arXiv:2306.05422*, 2023.

[133] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[134] Y. Song, J. Lei, Z. Wang, L. Liu, and K. Daniilidis, "Track everything everywhere fast and robustly," *arXiv preprint arXiv:2403.17931*, 2024.

[135] N. Tumanyan, A. Singer, S. Bagon, and T. Dekel, "DINO-Tracker: Taming DINO for self-supervised point tracking in a single video," *arXiv preprint arXiv:2403.14548*, 2024.

[136] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, *et al.*, "Kubric: A scalable dataset generator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3749–3761, 2022.

[137] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang, "Bounding box regression with uncertainty for accurate object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2888–2897, 2019.

[138] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518, 1992.

[139] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4040–4048, 2016.

[140] Z. Zhang, H. Jiang, and H. Singh, "NeuFlow: Real-time, high-accuracy optical flow estimation on robots using edge devices," *arXiv preprint arXiv:2403.10425v1*, 2024.

[141] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "DINOv2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[142] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009.

[143] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial intelligence and machine learning for multi-domain operations applications*, vol. 11006, pp. 369–386, SPIE, 2019.

[144] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[145] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," *arXiv preprint arXiv:1907.06987*, 2019.

[146] M. Vecerik, C. Doersch, Y. Yang, T. Davchev, Y. Aytar, G. Zhou, R. Hadsell, L. Agapito, and J. Scholz, "RoboTAP: Tracking arbitrary points for few-shot visual imitation," *arXiv preprint arXiv:2308.15975*, 2023.

[147] B. Biggs, T. Roddick, A. Fitzgibbon, and R. Cipolla, "Creatures great and smal: Recovering the shape and motion of animals from video," in *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part V 14*, pp. 3–19, Springer, 2019.

[148] H. Li, H. Zhang, S. Liu, Z. Zeng, T. Ren, F. Li, and L. Zhang, "TAPTR: Tracking any point with transformers as detection," *arXiv preprint arXiv:2403.13042*, 2024.

[149] K. Li, H. Liu, and T. Wang, "Centroid-based graph matching networks for planar object tracking," *Machine Vision and Applications*, vol. 34, no. 2, p. 31, 2023.

[150] H. Zhang and Y. Ling, "HVC-Net: Unifying homography, visibility, and confidence learning for planar object tracking," in *Computer Vision and Pattern Regognition (CVPR)*, 2022.

[151] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint*, 2024.

[152] D. Rozumnyi, J. Matas, M. Pollefeys, V. Ferrari, and M. R. Oswald, "Tracking by 3d model estimation of unknown objects in videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14086–14096, 2023.

[153] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2015 challenge results," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 777–823, 2016.

[154] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Cehovin Zajc, T. Vojir, G. Bhat, A. Lukezic, A. Eldesokey, *et al.*, "The sixth visual object tracking vot2018 challenge results," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[155] M. Kristan, J. Matas, M. Danelljan, M. Felsberg, H. J. Chang, L. Č. Zajc, A. Lukežič, O. Drbohlav, Z. Zhang, K.-T. Tran, *et al.*, "The first visual object tracking segmentation vots2023 challenge results," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1796–1818, 2023.

[156] L. Lin, H. Fan, Y. Xu, and H. Ling, "Swintrack: A simple and strong baseline for transformer tracking," *arXiv preprint arXiv:2112.00995*, 2021.

[157] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[158] S. Bai, Z. He, T.-B. Xu, Z. Zhu, Y. Dong, and H. Bai, "Multi-hierarchical independent correlation filters for visual tracking," *arXiv preprint arXiv:1811.10302*, 2018.

[159] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg, "Unveiling the power of deep tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 483–498, 2018.

[160] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. Čehovin, T. Vojíř, G. Häger, A. Lukežič, *et al.*, *The Visual Object Tracking VOT2016 Challenge Results*, pp. 777–823. Cham: Springer International Publishing, 2016.

[161] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele, "Lucid data dreaming for object tracking," in *The DAVIS Challenge on Video Object Segmentation*, 2017.

[162] P. Voigtlaender and B. Leibe, "Online adaptation of convolutional neural networks for video object segmentation," *British Machine Vision Conference (BMVC)*, 2017.

[163] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool, "Blazingly fast video object segmentation with pixel-wise metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1189–1198, 2018.

[164] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

[165] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[166] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results." http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[167] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *International Conference on Computer Vision (ICCV)*, 2011.

[168] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[169] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[170] Y. Zhang, D. Wang, L. Wang, J. Qi, and H. Lu, "Learning regression and verification networks for long-term visual tracking," *arXiv preprint arXiv:1809.04320*, 2018.

[171] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu, "Distractor-aware siamese networks for visual object tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 101–117, 2018.

[172] R. Xu, C. Wang, S. Xu, W. Meng, Y. Zhang, B. Fan, and X. Zhang, "DomainFeat: Learning local features with domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 1, pp. 46–59, 2023.

[173] A. Schmidt, O. Mohareri, S. DiMaio, M. C. Yip, and S. E. Salcudean, "Tracking and mapping in medical computer vision: A review," *Medical Image Analysis*, p. 103131, 2024.

[174] B. Wang, Y. Zhang, J. Li, Y. Yu, Z. Sun, L. Liu, and D. Hu, "SplatFlow: Learning multi-frame optical flow via splatting," 2023.

[175] A. Schmidt, O. Mohareri, S. DiMaio, and S. E. Salcudean, "Surgical tattoos in infrared: A dataset for quantifying tissue tracking and mapping," *IEEE Transactions on Medical Imaging*, 2024.

[176] S. Cho, J. Huang, S. Kim, and J.-Y. Lee, "FlowTrack: Revisiting optical flow for long-range dense tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19268–19277, 2024.

[177] Y.-T. Sun, Y.-H. Huang, L. Ma, X. Lyu, Y.-P. Cao, and X. Qi, "Splatter a video: Video gaussian representation for versatile processing," *arXiv preprint arXiv:2406.13870*, 2024.

[178] Z. Song, Y. Tang, R. Luo, L. Ma, J. Yu, Y.-P. P. Chen, and W. Yang, "Autogenic language embedding for coherent point tracking," in *ACM Multimedia 2024*.

[179] H. Li, H. Zhang, S. Liu, Z. Zeng, F. Li, T. Ren, B. Li, and L. Zhang, "TAPTRv2: Attention-based position update improves tracking any point," *arXiv preprint arXiv:2407.16291*, 2024.

[180] P. Kumar, N. Padmanabhan, L. Luo, S. S. Rambhatla, and A. Shrivastava, "Trajectory-aligned space-time tokens for few-shot action recognition," *arXiv preprint arXiv:2407.18249*, 2024.

[181] S. Cho, J. Huang, J. Nam, H. An, S. Kim, and J.-Y. Lee, "Local all-pair correspondence for point tracking," *arXiv preprint arXiv:2407.15420*, 2024.

[182] B. Wang, J. Li, Y. Yu, L. Liu, Z. Sun, and D. Hu, "Scenetracker: Long-term scene flow estimation network," *arXiv preprint arXiv:2403.19924*, 2024.

[183] R. Li and D. Liu, "Decomposition betters tracking everything everywhere," *arXiv preprint arXiv:2407.06531*, 2024.

[184] J. Guo, J. Wang, Z. Li, T. Jia, Q. Dou, and Y.-H. Liu, "Ada-tracker: Soft tissue tracking via inter-frame and adaptive-template matching," 2024.

[185] A. F. Sevilla, J. M. Lahoz-Bengoechea, and A. Díaz, "Automated extraction of prosodic structure from unannotated sign language video," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 1808–1816, 2024.

[186] J. J. Losada-del Olmo, Á. L. Perales Gómez, A. Ruiz, and P. E. López de Teruel, "A few-shot learning methodology for improving safety in industrial scenarios through universal self-supervised visual features and dense optical flow," *Available at SSRN 4777359*.

APPENDIX

A

LIST OF PUBLICATIONS

A.1 CONFERENCE PAPERS IN WOS

Publication:
J. Šerých and J. Matas, "Visual coin-tracking: Tracking of planar double-sided objects," in *German Conference on Pattern Recognition*, pp. 317–330, Springer, 2019

Authorship statement:
Jonáš Šerých: methodology, software, validation, formal analysis, investigation, data curation, writing - original draft, visualization.
Jiří Matas: conceptualization, methodology, writing - review & editing, supervision, project administration, funding acquisition

Citations:

- D. Rozumnyi, J. Matas, M. Pollefeys, V. Ferrari, and M. R. Oswald, "Tracking by 3d model estimation of unknown objects in videos," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14086–14096, 2023

Publication:
J. Šerých and J. Matas, "Planar object tracking via weighted optical flow," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1593–1602, 2023
Authorship statement:
Jonáš Šerých: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing - original draft, visualization.
Jiří Matas: conceptualization, methodology, writing - review & editing, supervision, project administration, funding acquisition
Citations:

- R. Xu, C. Wang, S. Xu, W. Meng, Y. Zhang, B. Fan, and X. Zhang, "DomainFeat: Learning local features with domain adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 1, pp. 46–59, 2023
- X. Liu, X. Liu, Z. Yi, X. Zhou, T. Le, L. Zhang, Y. Huang, Q. Yang, and H. Fan, "PlanarTrack: A large-scale challenging benchmark for planar object tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20449–20458, October 2023

Publication:
M. Neoral, J. Šerých, and J. Matas, "MFT: Long-term tracking of every pixel," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6837–6847, 2024
Authorship statement:
Michal Neoral: conceptualization, methodology, software, validation, formal analysis, investigation, writing - original draft, visualization.
Jonáš Šerých: conceptualization, methodology, software, validation, formal

analysis, investigation, data curation, writing - original draft, visualization. Jiří Matas: conceptualization, methodology, writing - review & editing, supervision, project administration, funding acquisition
Citations:

- Q. Wang, Y.-Y. Chang, R. Cai, Z. Li, B. Hariharan, A. Holynski, and N. Snavely, "Tracking everything everywhere all at once," *arXiv:2306.05422*, 2023
- A. Schmidt, O. Mohareri, S. DiMaio, M. C. Yip, and S. E. Salcudean, "Tracking and mapping in medical computer vision: A review," *Medical Image Analysis*, p. 103131, 2024
- G. Le Moing, J. Ponce, and C. Schmid, "Dense optical tracking: connecting the dots," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19187–19197, 2024
- Y. Xiao, Q. Wang, S. Zhang, N. Xue, S. Peng, Y. Shen, and X. Zhou, "SpatialTracker: Tracking any 2d pixels in 3d space," *arXiv preprint arXiv:2404.04319*, 2024
- C. Doersch, Y. Yang, D. Gokay, P. Luc, S. Koppula, A. Gupta, J. Heyward, R. Goroshin, J. Carreira, and A. Zisserman, "BootsTAP: Bootstrapped training for tracking-any-point," 2024
- B. Wang, Y. Zhang, J. Li, Y. Yu, Z. Sun, L. Liu, and D. Hu, "SplatFlow: Learning multi-frame optical flow via splatting," 2023
- N. Tumanyan, A. Singer, S. Bagon, and T. Dekel, "DINO-Tracker: Taming DINO for self-supervised point tracking in a single video," *arXiv preprint arXiv:2403.14548*, 2024
- A. Schmidt, O. Mohareri, S. DiMaio, and S. E. Salcudean, "Surgical tattoos in infrared: A dataset for quantifying tissue tracking and mapping," *IEEE Transactions on Medical Imaging*, 2024
- H. Li, H. Zhang, S. Liu, Z. Zeng, T. Ren, F. Li, and L. Zhang, "TAPTR: Tracking any point with transformers as detection," *arXiv preprint arXiv:2403.13042*, 2024
- S. Cho, J. Huang, S. Kim, and J.-Y. Lee, "FlowTrack: Revisiting optical flow for long-range dense tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19268–19277, 2024
- Y.-T. Sun, Y.-H. Huang, L. Ma, X. Lyu, Y.-P. Cao, and X. Qi, "Splatter a video: Video gaussian representation for versatile processing," *arXiv preprint arXiv:2406.13870*, 2024
- Z. Song, Y. Tang, R. Luo, L. Ma, J. Yu, Y.-P. P. Chen, and W. Yang, "Autogenic language embedding for coherent point tracking," in *ACM Multimedia 2024*
- H. Li, H. Zhang, S. Liu, Z. Zeng, F. Li, T. Ren, B. Li, and L. Zhang, "TAPTRv2: Attention-based position update improves tracking any point," *arXiv preprint arXiv:2407.16291*, 2024
- Y. Song, J. Lei, Z. Wang, L. Liu, and K. Daniilidis, "Track everything everywhere fast and robustly," *arXiv preprint arXiv:2403.17931*, 2024

- P. Kumar, N. Padmanabhan, L. Luo, S. S. Rambhatla, and A. Shrivastava, "Trajectory-aligned space-time tokens for few-shot action recognition," *arXiv preprint arXiv:2407.18249*, 2024
- S. Cho, J. Huang, J. Nam, H. An, S. Kim, and J.-Y. Lee, "Local all-pair correspondence for point tracking," *arXiv preprint arXiv:2407.15420*, 2024
- B. Wang, J. Li, Y. Yu, L. Liu, Z. Sun, and D. Hu, "Scenetracker: Long-term scene flow estimation network," *arXiv preprint arXiv:2403.19924*, 2024
- R. Li and D. Liu, "Decomposition betters tracking everything everywhere," *arXiv preprint arXiv:2407.06531*, 2024
- J. Guo, J. Wang, Z. Li, T. Jia, Q. Dou, and Y.-H. Liu, "Ada-tracker: Soft tissue tracking via inter-frame and adaptive-template matching," 2024
- A. F. Sevilla, J. M. Lahoz-Bengoechea, and A. Díaz, "Automated extraction of prosodic structure from unannotated sign language video," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 1808–1816, 2024
- J. J. Losada-del Olmo, Á. L. Perales Gómez, A. Ruiz, and P. E. López de Teruel, "A few-shot learning methodology for improving safety in industrial scenarios through universal self-supervised visual features and dense optical flow," *Available at SSRN 4777359*

A.2 WORKSHOP PAPERS

Publication:
T. Jelínek, J. Šerých, and J. Matas, "Dense matchers for dense tracking," in *Proceedings of the 27th Computer Vision Winter Workshop (CVWW 2024)*, 2024
Authorship statement:
Tomáš Jelínek: methodology, software, validation, formal analysis, investigation, writing - original draft, visualization.
Jonáš Šerých: conceptualization, methodology, software, validation, formal analysis, investigation, writing - original draft, visualization.
Jiří Matas: conceptualization, methodology, writing - review & editing, supervision, project administration, funding acquisition

A.3 UNDER REVIEW IN WOS-EXCERPTED CONFERENCE

Publication:
J. Šerých, M. Neoral, and J. Matas, "MFTIQ: Multi-flow tracker with independent matching quality estimation," *under review*, 2025
Authorship statement:
Jonáš Šerých: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing - original draft, visualization.
Michal Neoral: conceptualization, methodology, software, validation, formal analysis, investigation, writing - original draft, visualization.
Jiří Matas: conceptualization, methodology, writing - review & editing, supervision, project administration, funding acquisition