Robust Algorithm for Planar Objects Tracking in Real Life Cases

(Praktyczny algorytm
do śledzenia płaskich powierzchni
w rzeczywistych zastosowaniach)

Dominik Samorek

Praca magisterska

Promotor: dr Rafał Nowak

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

17 lutego 2022

Abstract

Tracking planar structures is a more specific subproblem of general homography estimation which is an old and basic task in computer vision. While classic approaches developed over the years tried to achieve better performance and robustness, they still often fail in hard, real-life cases. On the other hand some new, deep learning based solutions obtained better performance when dealing with practical challenging factors like significant lighting changes or dynamic scenes, they rather tackle the problem of general image alignment on synthetic datasets and cannot be used for planar tracking easily. In this thesis we propose a combined algorithm for planar structures tracking along with its implementation. It is based on classic homography estimation approaches but also utilizes novel deep learning techniques for relocalization in hard cases. Our solution achieves significantly better precision that previous state-of-the-art approaches and low computational cost being able to run in real-time on ordinary desktop machine.

Śledzenie płaskich powierzchni jest podproblemem bardziej ogólnego i wręcz podstawowego w wizji komputerowej zadania, jakim jest estymacja homografii. Podczas gdy tradycyjne metody ulepszane z biegiem lat osiągały coraz lepszą skuteczność i odporność, wciąż często zawodziły w trudnych, rzeczywistych przypadkach. Z drugiej strony nowe podejścia oparte na głębokich sieciach neuronowych okazały się bardziej skuteczne w starciu z praktycznymi trudnościami jak znaczące zmiany oświetlenia i dynamiczne sceny. Próbują one jednak zwykle rozwiązać problem ogólnego dopasowania zdjęć na syntetycznych zbiorach danych i nie mogą być łatwo zaaplikowane do zadania śledzenia płaskich powierzchni. W tej pracy przedstawiamy złożony algorytm do śledzenia płaskich powierzchni wraz z implementacją. Jest on oparty na klasycznych metodach do estymacji homografii, ale korzysta również z nowych osiągnięć głębokich sieci neuronowych do relokalizacji w trudnych przypadkach. Nasze rozwiązanie osiąga zdecydowanie lepszą dokładność niż wcześniejsze metody, jak również wymaga niedużych zasobów obliczeniowych, mogąc działać w czasie rzeczywistym na przeciętnym komputerze.

Contents

Chapter 1

Introduction

Planar object tracking task comes from classic homography estimation problem, which is one of the fundamentals of computer vision. The aim is to obtain an accurate mapping from template image to its projection on a video. Despite being old, it is still an important and active area of research, because of its applications in augmented reality and robotics. Algorithms developed over time need to be constantly improved to handle more challenges that current applications bring. While in past the algorithms needed to work only in artificial scenarios, nowadays there are much more real life applications, for which more robustness is needed. In the same time, during fast technological development of hardware, we also expect better accuracy in novel approaches. Finally, because there is a constantly increasing number of applications on IoT and mobile devices, low computational cost is very important. Therefore, these three targets should be a common area of focus: robustness, accuracy and computational cost.

In this thesis, we propose a combined algorithm based on classic homography estimation approaches but also utilizing novel deep learning techniques for relocalization in hard cases. Our solution significantly outperforms other state-of-the-art approaches in accuracy on the recent benchmark [13], while being very efficient in terms of computational cost and running in real-time. Together with the thesis we provide the source code of our implementation, on which all the tests have been made.

The thesis consists of the following parts. In Chapter 2 we introduce the theory needed to understand and properly define the problems of homography estimation and planar object tracking. In Chapter 3 we go through classic approaches and algorithms from the field. In Chapter 4 we introduce the benchmark on which we evaluate our solution. In Chapter 5 we present our algorithm with a special attention over the important steps and, in Chapter 6, we show the evaluation results of our solution in comparison with other algorithms applied on the common benchmark. Finally, in Chapter 7 we give a summary of our findings like also possible future improvements to our approach.

Chapter 2

Theoretical Background and Problem Description

To understand the problem of planar objects tracking we need a set of mathematical tools to properly describe it. First, we need to define the transformation taking place in a camera when taking a photo. Next we will derive the formal definition of homography, as a main objective of our approach, with its most important properties. Finally, having all the necessary tools, we will formally define the problem of planar object tracking.

2.1 Pinhole Camera Model

To formally describe the transformation taking place when we take a photo of a scene, we first need to define the model of a camera. The most popular one is so called a *pinhole camera* model. It assumes that between the projective plane and the scene is a small hole, that only one photon from each direction can pass through it. Therefore each point on the image comes from exactly one ray which starts in single 3D point in the scene, see Fig. 2.1.

Because each point is transported on exactly one ray, there is no blur and the image is perfectly well focused. We also assume that each ray has enough amount of light for each point to be visible. These assumptions are of course not satisfied in real world cameras. Such small hole would not be enough to pass sufficient amount of light in reasonable time, that is why, we use lenses. Nevertheless, such assumptions give us a well defined conditions that we can further utilize for mathematical tools.

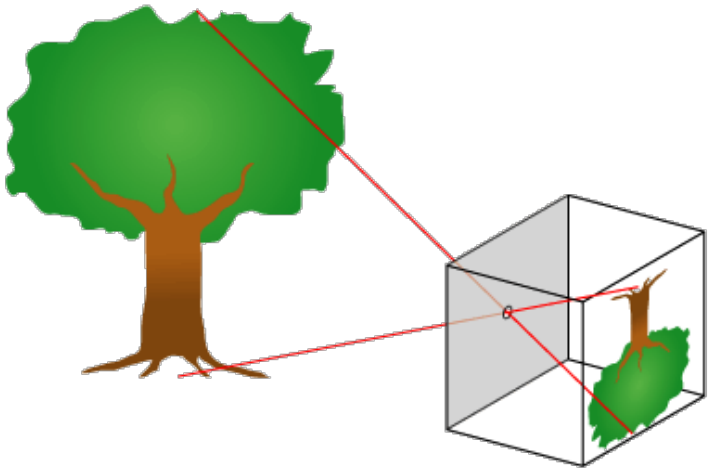


Figure 2.1: The tree from the scene is perfectly mapped on the image plane with usage of pinhole camera. Because each point is transported on exactly one ray, there is no blur and the image is perfectly well focused. We also assume that each ray has enough amount of light for each point to be visible. Source: `https://commons.wikimedia.org/wiki/File:Pinhole-camera.png` (last access: 11.11.2021).

2.2 Projective Geometry

Transformation that we introduced in previous section can be explained mathematically. But first we need to show the space in which it will be well-defined. For that purpose, a field called projective geometry was invented, which can connect the 3D world and its projections on the images.

Homogeneous Coordinates

We want to get both the 3D world and its projections on images in one space called projective space. As we said, projected 2D point on the image can be equated to a ray coming from corresponding 3D point through the center of projection. Since, from the image point of view, we do not know which particular 3D point on this ray produced the projection, we can represent it as a set of all possible 3D points using the following homogeneous coordinates:

$$\begin{pmatrix} x \\ y \end{pmatrix} \backsimeq \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \backsimeq \begin{pmatrix} wx \\ wy \\ w \end{pmatrix}. \tag{2.1}$$

As we can see in Equation (2.1), point $(x, y)^T$ on the image plane is equivalent with point $(x, y, 1)^T$ and $(wx, wy, w)^T$ for every $w \neq 0$. Such representation uses an assumption that the center of projection C in 3D world is located in $(0, 0, 0)^T$,

and the projective plane **p** in the coordinate $Z = 1$, see Fig. 2.2. Then the point $(x, y)^T$ from the image is a $(x, y, 1)^T$ point in the 3D world.

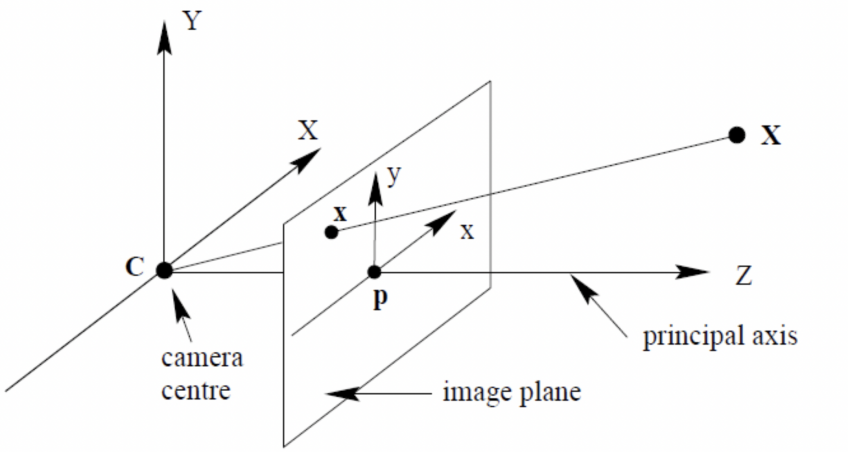


Figure 2.2: Projective space with included projective plane. Often for simplicity we assume that the projective plane is located in coordinate $Z = 1$ and that the center of the projection is in the center of the coordinate system. Axis Z is perpendicular to the projective plane. Source: [9].

However, it is not always the case that the center of the coordinate system and the center of projection overlap. We would like to model also applications where we have several cameras with different positions and orientations taking photos of one scene from different viewpoints.

2.2.1 Camera Matrices

Camera matrices are the component that can model real camera's position in the world and the transformation taking place when taking a photo. It can also express the physical parameters of the camera such as focal length or the resolution in a mathematical way. Camera matrices are essential for the following *Projection Equation*:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \, [R \mid T] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.2}$$

Left side of the Equation (2.2) is the $(u, v)^T$ image point represented in homogeneous coordinates along with a scale s. On the right side we have intrinsic camera matrix K, extrinsic matrix $[R \mid T]$ and 3D point from the scene $(X, Y, Z)^T$ which is being projected. This 3D point is also represented in homogeneous coordinates with additional fourth coordinate equal 1. We use it to simplify the notation rather than to give additional interpretation, what can be seen during further camera matrices

analysis. Equation (2.2) is a full description of the transformation from the 3D world into 2D projection plane.

Extrinsic Camera Matrix

Extrinsic matrix allows to move between the coordinate system of the world/scene and each particular camera. It is therefore the component that enables having the center of projection in different place than the center of the world coordinate system. Its notation from Equation (2.2) we can unfold in the following way:

$$[R \mid T] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix}.$$

The matrix consists of two parts, R and T stacked together. R is a rotation matrix between two coordinate systems and T is a translation vector between these systems. We can see that thanks to the addition fourth coordinate of the point $(X, Y, Z)^T$ equal to 1 we can express the translation as a matrix multiplications instead of addition what simplifies the notation, especially when combining few of such transformations. In applications of 3D reconstruction or localisation, we often refer to the extrinsic camera matrix as a camera pose, because it describes the camera's position and orientation in world coordinate system.

Going back to the projection equation (2.2), after applying extrinsic matrix to the point $(X, Y, Z)^T$ we get a 3D point in a camera's coordinate system with the center of projection in a point $(0, 0, 0)^T$ and the projection plane on coordinate $Z = 1$. Now we can use the intrinsic camera matrix to transform it into a 2D projection on an image.

Intrinsic Camera Matrix

In a similar way let us unfold what is inside the intrinsic matrix K in the Equation (2.2):

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.3}$$

We can now take a look on each particular element of the matrix K, according to the Equation (2.3). On the diagonal we have values f_x and f_y which are responsible for scaling world coordinates into the pixels. They are directly related to the focal length of the camera. Often K is simplified with an assumption that $f_x = f_y$, which should be true in ideal pinhole camera, but in real world there might be some inaccuracies because of some lens deformations. In the third column of the matrix

we have once more a translation component, namely the c_x and c_y elements. It is needed because on images we commonly place the origin in the corner, not in the center. c_x and c_y are therefore values equal to half of the image resolution, because they translate the center of the image into its corner.

Scale

By analysing the left side of the projection equation (2.2) we can see an isolated scale component. It comes from the fact that we already observed introducing homogeneous coordinates. For particular point on the image we are not able to tell how far on the incoming ray the corresponding 3D point is located. When looking at the object on the image we do not know if it is big and located far away, or small and really near, everything is defined up to a scale in homogeneous coordinates. Thus, the s coefficient is a byproduct of the projection and we usually do not have any use of it.

2.3 Homography

After the introduction we now should understand each component of the projection equation (2.2) and see how by using it, transform 3D points on the scene into 2D projections on the image. Now we can proceed to define a homography and explain how it can be used. For that we will need two cameras, because the homography is a transformation between two images i.e. it transforms points from one image into the other as shown in the Figure 2.3.

With a homography we can represent two types of transformations. First, when we have two projections of the same plane, for example a building wall, billboard or a part of the floor. Second occurs when the movement of the camera between the first and the second photo is a pure rotation without translation. Then it is no longer important if the scene is a plane. Most common example of such case is when we take a series of photos to combine them into one panorama of a far away landscape.

We will go back to both this cases after analysing the relation between the homography and camera matrices.

2.3.1 Homography and Camera Matrices Relation

Let K_a, K_b, $[R_a \mid T_a]$ and $[R_b \mid T_b]$ be respectively intrinsic and extrinsic matrices of cameras a and b, with which we took a photos of a plane P. We are looking for a homography H, that can transform points from an image from camera b into the points of the image of camera a. Additionally, let n be the normal vector of a plane

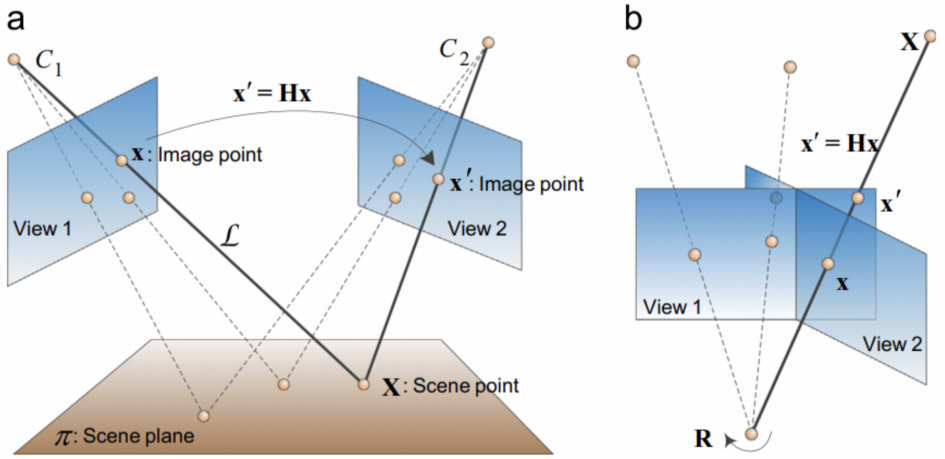


Figure 2.3: Homography can describe two types of transformations. First (a), between two projections of the same plane in a 3D world. And the second (b), between two projections of unconstrained 3D scene, not necessarily a plane, but the transformation between these two cameras needs to be a pure rotation without translation. Source: [10].

P and d be the distance from camera b to the plane P. Then we can express homography H with an equation:

$$H = s\, K_a \left(R_a R_b^T + \frac{(R_a R_b^T T_b - T_a) n^T}{d} \right) K_b^{-1}. \tag{2.4}$$

First thing that we can note is once again the scale coefficient s. It means, that the homography is also defined up to a scale i.e. H and sH represent the same transformation for all $s \neq 0$. When analysing further the Equation (2.4) we can notice the intrinsic matrices K_a and K_b^{-1}. They transform the point from image b into the projective plane in 3D world and then from the projection plane into the image a. We can think about them as a unit conversion. The factor in the parentheses is more complicated. But we can see that $R_a R_b^T$ is in fact a relative rotation from camera b to the camera a. It means that $(R_a R_b^T T_b - T_a)$ is in fact a relative translation from a to b. We can simplify the notation by introducing $R_{ba} \coloneqq R_a R_b^T$ and $T_{ab} = (R_a R_b^T T_b - T_a)$. Then we have

$$H = s\, K_a \left(R_{ba} + \frac{T_{ab}\, n^T}{d} \right) K_b^{-1}. \tag{2.5}$$

Let us see what happens to the point $\mathbf{p}$ being transformed by the homography H. For simplicity, let assume that $\mathbf{p}$ is already in the projective plane and we can forget about the intrinsic matrices. From Equation (2.5) we can write:

$$\begin{aligned}\tilde{H}\mathbf{p} &= \left(R_{ba} + \frac{T_{ab}\, n^T}{d}\right)\mathbf{p} \\ &= R_{ba}\mathbf{p} + \frac{T_{ab}\, n^T}{d}\mathbf{p} \\ &= R_{ba}\mathbf{p} + \frac{T_{ab}\, n^T\mathbf{p}}{d} \\ &= R_{ba}\mathbf{p} + \frac{(n^T\mathbf{p})T_{ab}}{d}.\end{aligned}$$

The component with R_{ba} is quite intuitive, we transform the point $\mathbf{p}$ from the coordinate system of the camera b to the system of camera a according to the relative rotation. Translation component is more complicated - instead of just adding the translation we weight it by the $\frac{(n^T\mathbf{p}}{d}$ coefficient. d is constant for all points, but $n^T\mathbf{p}$ takes most significant value when $\mathbf{p}$ is parallel to the normal vector of a plane and equals 0 when it is perpendicular.

For deriving these formulas we made an assumption that the scene is a plane with a distance d from a camera. That was the first case of the homography. We can see that in the second case the translation $T_{ab} = 0$, so the whole second component disappears and we have just a normal rotation transformed by the intrinsic cameras matrices. d also disappears so the assumption about a plane scene is no longer needed.

2.3.2 Homography Properties

From Equation (2.4) follows the fact, that we can represent homography as a 3×3 matrix. However, it does not mean that this transformation is linear. It is not in general, what can be easily seen when taking a look at how it transforms a 2D point $(x, y)^T$ on an image:

$$H\begin{pmatrix}x\\y\\1\end{pmatrix} = \begin{pmatrix}h_{11} & h_{12} & h_{13}\\ h_{21} & h_{22} & h_{23}\\ h_{31} & h_{32} & h_{33}\end{pmatrix}\begin{pmatrix}x\\y\\1\end{pmatrix} = \begin{pmatrix}h_{11}x + h_{12}y + h_{13}\\ h_{21}x + h_{22}y + h_{23}\\ h_{31}x + h_{32}y + h_{33}\end{pmatrix} = \begin{pmatrix}\frac{h_{11}x+h_{12}y+h_{13}}{h_{31}x+h_{32}y+h_{33}}\\ \frac{h_{21}x+h_{22}y+h_{23}}{h_{31}x+h_{32}y+h_{33}}\\ 1\end{pmatrix}, \tag{2.6}$$

thus

$$\begin{pmatrix}x\\y\end{pmatrix} \overset{H}{\longmapsto} \begin{pmatrix}\frac{h_{11}x+h_{12}y+h_{13}}{h_{31}x+h_{32}y+h_{33}}\\ \frac{h_{21}x+h_{22}y+h_{23}}{h_{31}x+h_{32}y+h_{33}}\end{pmatrix}.$$

From Equation (2.6) it follows, that the homography transforms point $(x, y)^T$ into the point $(\frac{h_{11}x+h_{12}y+h_{13}}{h_{31}x+h_{32}y+h_{33}}\, \frac{h_{21}x+h_{22}y+h_{23}}{h_{31}x+h_{32}y+h_{33}})^T$, what in general is not a linear transformation.

We said, that the homography is defined up to the scale, therefore most often homography matrix is normalized to have 1 in the bottom right corner. The rest 8 parameters of the matrix can be changed freely, therefore it has 8 degrees of freedom. Hence, we should be able to determine the homography matrix from 4 pairs of corresponding 2D points on the two images, because each such pair would give us two constraints on the matrix. From that we can derive the system of equations having a unique solution if and only if for both these images none 3 of 4 points would be collinear, as shown on the Figure 2.4.

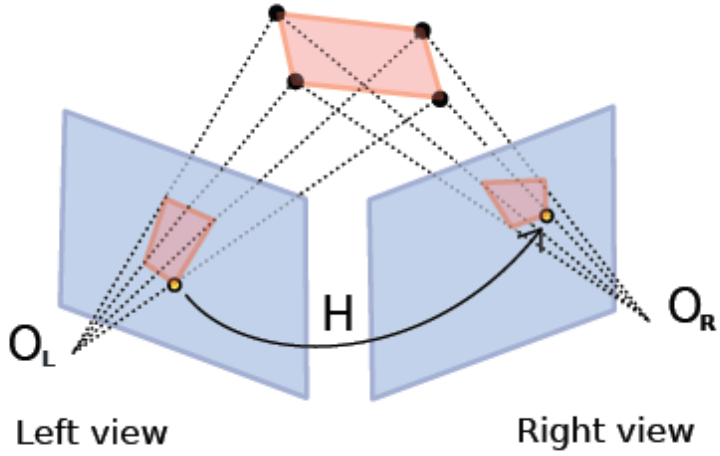


Figure 2.4: Two views of the same red quadrilateral. Sample pair of corresponding points is showed in yellow. Source: `https://alicevision.readthedocs.io/en/latest/openMVG/multiview/multiview.html#homographyfigure` (last access: 8.09.2021).

Let $(\mathbf{p_1}, \mathbf{p'_1}), (\mathbf{p_2}, \mathbf{p'_2}), (\mathbf{p_3}, \mathbf{p'_3}), (\mathbf{p_4}, \mathbf{p'_1})$ be corresponding points pairs. For each pair $(\mathbf{p_i}, \mathbf{p'_i})$ we can derive a system of two equations using the Equation (2.6):

$$\begin{cases} x'_i = \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}, \\ y'_i = \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}. \end{cases}$$

After simple transformation we get to the system:

$$\begin{cases} -h_{11}x_i - h_{12}y_i - h_{13} + h_{31}x_i x'_i + h_{32}y_i x'_i + h_{33}x'_i = 0, \\ -h_{21}x_i - h_{22}y_i - h_{23} + h_{31}x_i y'_i + h_{32}y_i y'_i + h_{33}y'_i = 0. \end{cases}$$

By combining all these systems into one big system of equations and writing it in a matrix form we get:

$$PH = \begin{pmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y'_1 & y_1y'_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y'_2 & y_2y'_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x'_3 & y_3x'_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y'_3 & y_3y'_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x'_4 & y_4x'_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y'_4 & y_4y'_4 & y'_4 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = 0.$$

Such system of equations, with additional constraint $||H|| = 1$, we can solve using SVD decomposition. By finding the decomposition $P = U\Sigma V^T$, as a solution we can take the right singular vector corresponding to singular value equal to 0. Additional constraint $||H|| = 1$ is insignificant, because H is defined up to the scale.

One of the most important property of the homography is the fact that it is an invertible transformation, we can find the inverse by simply inverting the matrix H. In practice, also one additional property is important - homography, despite being nonlinear, maps straight lines to straight lines. However, it does not necessarily preserve angles and proportions.

Homographies are also defined for higher dimensional spaces, however they do not have applications related to images.

2.4 Planar Object Tracking

Now we have all necessary definitions and tools to properly describe our main task. Planar object tracking is a problem that consists of an object that we want to track and a video on which this object is recorded. In our case we will assume that we have a template image of the tracked object $\mathbf{T}$, e.g. undistorted picture of a road sign that we are tracking.

Having the proper template $\mathbf{T}$ we can now estimate the homography between $\mathbf{T}$ and current video frame, so that we know how to map the template onto the frame. In tracking task we need to estimate the homography on each frame in a sequential manner, we start from the first frame and we can use the information from the past for the next estimation. In this setting, as a result we have a homography H_i for each video frame $\mathbf{F}_i$ that maps the template on that particular frame.

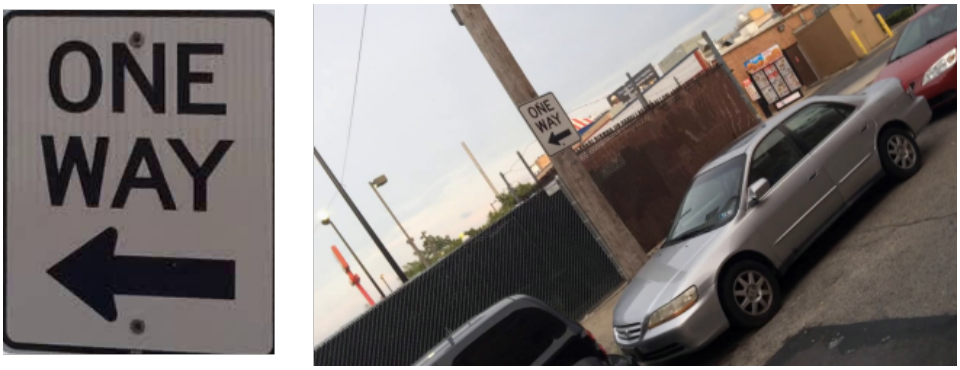


Figure 2.5: Sample template image of a planar object that could be tracked alongside with a frame from a video on which it is recorded. The example is taken from [13].

Chapter 3

Related Work

Most popular algorithms for robust homography estimation and planar object tracking are typically representatives of one of two groups: keypoint-based and region-based methods. We will try to describe the general idea of both of these classes of methods in this chapter, as also introduce some of particular algorithms in the next one.

3.1 Keypoint-based Methods

As mentioned in Section 2.3 the easiest way to establish the homography is to have 4 points correspondences and based on that compute the matrix from the direct formula. Keypoint-based approaches follow the same way, but with a more robust solution, which adapts to much more than just 4 correspondences.

The general algorithm looks like this:

Algorithm 1 General keypoint-based algorithm

1: Detect the sets of keypoints on both images P_1, P_2.
2: Find the matching between P_1 and P_2 getting the set of correspondences $C \subseteq P_1 \times P_2$.
3: Estimate the homography H based on C.

Figure 3.1 shows keypoint-based homography estimation example. Each step of Algorithm 1 can be seen there, but we will describe them in more detail.

3.1.1 Detecting the Keypoints

By analyzing the Algorithm 1 we can name the desired properties of detected keypoints as following:

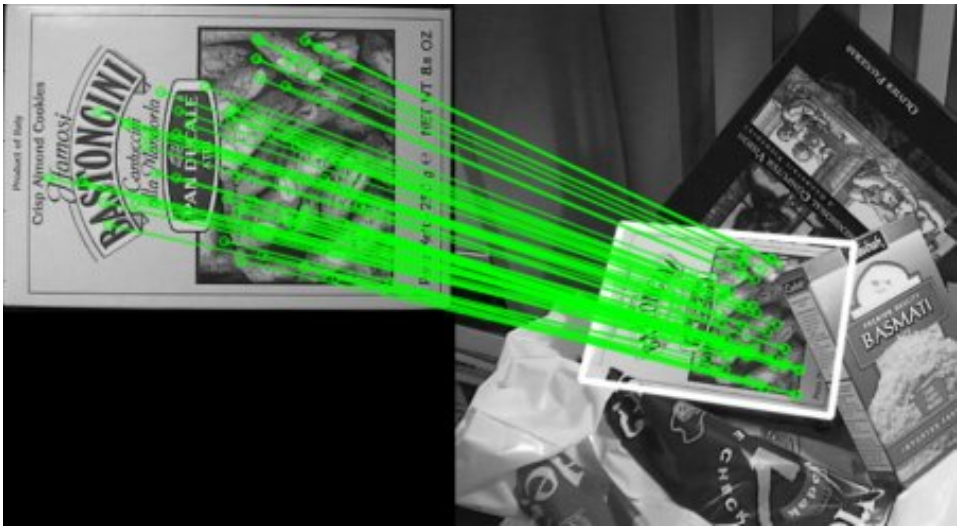


Figure 3.1: Keypoint-based method in practice. On the left there is a reference template image, on the right the scene having the transformed template. On both images detected keypoints were drawn with small green circles. Corresponding pairs were connected with green lines. Computed transformation was used to draw the white quadrilateral around estimated position of transformed template image. Source: `https://docs.opencv.org/4.5.2/dc/dc3/tutorial_py_matcher.html` (last access: 8.08.2021).

1. Covariant to spatial transformations like, rotations, translations, perspective, shear and mirroring.
2. Invariant to illumination changes.
3. Easily distinguishable.
4. Sparse - the number of keypoints must be much less than the number of pixels.
5. Local - each keypoint occupies only small area.

Items 1° and 2° are fundamental, i.e. to estimate the homography based on keypoints, it must be possible to detect the same keypoints on two images transformed in different way, e.g. taken with different distances, light conditions or with different viewpoints. Item 3° should make the matching step a solvable problem. Item 4° is important from computational resources point of view - the smaller number of keypoints necessary to find a proper homography the less computations needed to be performed. Similarly item 5°, points matching requires the analysis of the important keypoint area, so the smaller it is the faster the computations. Also, lower is the chance for a keypoint to be occluded, what often ends up with a wrong homography.

In practice, keypoints are most often defined as some kind of corners, so points that are local maximas of variously defined "cornerness" functions on the image domain. Different functions have different approaches to satisfy all the requirements

to keypoints properties. The most classical ones are Harris [8] and Harris-Laplace [15] corner detectors.

3.1.2 Descriptors

After detecting the keypoints, according to Algorithm 1, it is important to establish a way to compare those keypoints to find the matching. Depending on the priors that we have for matching problem, different approaches can be formulated. If we know that two images are transformed by only small translation, we can search for correspondences in the small neighbourhood. In general, however, we do not have such assumptions and our matching technique needs to handle every possible configuration of correspondences. In such case global matching techniques are applied and descriptors are the way to do it.

As the name suggests, descriptor is an object able to describe the keypoint. In most cases it is a vector of values, that should in most unique way, but also invariant to spatial and illumination transformations, characterize particular points and its neighbourhood. Having that, it is possible to for each keypoint from one image, find the closest one in the descriptors space on the second image, and in this way establish the matching.

Many descriptors, such as SIFT [14] or ORB [20] are designed for particular keypoints detectors, with which they work best, because of common assumptions of possible transformations and number of computations constraints, therefore keypoints detector and descriptor are sometimes jointly known by same name.

3.1.3 Estimating the Homography

On this stage of the Algorithm 1, we have detected keypoints on both images paired into correspondences. We know from Section 2.3 that homography can be computed from system of equations using 4 correspondences. Such system could be easily extended to more points and final homography would minimize the mean squared error on correspondences distance. In practice, such approach would not be robust enough. Previous steps of Algorithm 1 were heuristic approaches to really hard problems, so we have no guaranties that the set of correspondences is free from errors. Such mismatch would totally destroy our solution based on minimizing the mean squared error, therefore more robust approach must be proposed.

RANSAC

Random Sample Consensus [6], is a simple and general algorithm able to fit a model to data even in the presence of many outliers. It can be applied to many possible problems, but here we describe its version in homography estimation.

Algorithm 2 RANSAC

1: $C_i \leftarrow \emptyset$
2: **while** Some STOP condition **do**
3: From set of correspondences C sample 4 pairs S
4: Compute homography H_s based on S
5: $C_s \leftarrow$ subset of C consistent with H_s
6: **if** $|C_s| > |C_i|$ **then**
7: $C_i \leftarrow C_s$
8: **end if**
9: **end while**
10: Estimate and return homography H based on C_i

Algorithm 2 contains few steps requiring some clarification. First of all, it has undefined stop condition. There are a lot of possibilities there, the basic one with given number of repetitions. Most popular one is, based on number of correspondences and estimated noise factor, number of steps necessary to find good subset of points with given probability. In each such iteration we sample 4 correspondences, to compute the homography. Sampling techniques can also be modified, to give particular points higher probability, because of additional information. Then we find all correspondences (p, p') for which the distance between Hp and p' does not exceed a fixed threshold value. All such correspondences are considered inliers. After finishing the loop we should have the set of correspondences without outliers for which we can compute the final homography minimizing the mean squared error.

In practice, usage of this or a similar robust estimator is crucial to achieve good results, because in real life examples the ratio of outliers can be huge.

3.2 Region-based Methods

In keypoint-based methods we use only the information contained in few hundreds points out of thousands or even millions of pixels. Also, apart from the number of inliers, we have no other way of checking if the homography is good. In region-based approaches, also called direct methods, these issues are being addressed. Instead of separate steps of detecting the keypoints, matching and estimating the homography, we formulate an optimization problem *directly* on the given images and solve it using iterative minimization methods. Such algorithms can be especially effective in planar object tracking applications, because we can use the assumption, that homography computed for previous video frame is close to the next one, so only small number of optimization iterations is needed.

3.2.1 Optimization Problem

Let $\mathbf{x_c}$ be parameters of homography we are looking for, $\mathbf{e}$ parameters of identity homography, and $s(\mathbf{x})$ be an image warped with a homography with parameters $\mathbf{x}$. Them, the error function f we can write as follows:

$$f(\mathbf{x}) = \frac{1}{2}\|s(\mathbf{x} \circ \mathbf{x_c}) - s(\mathbf{e})\|^2 = \frac{1}{2}\sum_{k=1}^{q}\left(s_k(\mathbf{x} \circ \mathbf{x_c}) - s_k(\mathbf{e})\right).$$

$\mathbf{x}$ is a current estimation of the inverse of $\mathbf{x_c}$ homography. The sum from $k = 1$ to q runs through all pixels of the image. The aim is to find the argument that minimizes the value of a function f:

$$\arg\min(f(\mathbf{x})) = \mathbf{x_c}^{-1}.$$

For this task we can use the general minimization strategies, which rely on iterative minimum approximation, by taking a step $\Delta\mathbf{x}$ in each iteration. The way in which we choose the direction and length of this step is the main difference between all the approaches. Different strategies we can depict by appropriately chosen matrix $\mathbf{S}$ in the following formula:

$$\Delta\mathbf{x} = -\mathbf{S}^{-1}\mathbf{g}.$$

We distinguish methods of first and second order. First, because they use only information of the first derivatives of the error function, $\mathbf{S}$ can take a form $\mathbf{S} = \alpha\mathbf{I}$. Second-order methods use also the second derivatives, then $\mathbf{S}$ becomes a Hessian. In practice, because of high dimensionality of the problem, computing the exact values of second derivatives might be inefficient, therefore often some kind of approximations are used instead.

Because we minimize the mean squared error, we can use the Gauss-Newton method, that approximates the Hessian with a usage of a Jacobian, containing only first derivatives:

$$\mathbf{S} = \mathbf{J}^T\mathbf{J}.$$

Often employed is its modification called Levenberg-Marquandt method:

$$\mathbf{S} = \mathbf{J}^T\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{J}).$$

There exist also a minimization strategies designed directly for planar object tracking, for example ESM [4], GO-ESM [5], IC [1] and SCV [18], which try to further utilize special properties of this problem and model it so that a lot of computations are shared between iterations.

Beside the minimization algorithm, also the error function can be modified. Often additional component based on image gradient is used, to reduce the impact of illumination differences. Having other assumptions about the problem, they also can be easily taken into account, by for example constraining the areas on which we search for the minimum.

Chapter 4

Benchmark Introduction

To efficiently and reliably compare many algorithms in the same task, it is very useful to have a common benchmark. By trying different approaches in the same conditions, we can draw true conclusions. For planar object tracking, there exist a few benchmarks, but we will focus on the one that has the properties that we are looking for. Planar Object Tracking in the Wild: A Benchmark [13], as the name suggests, focuses on real life examples. It provides the data with the most often and significant for this task challenging factors, both isolated and combined. The paper itself also contains the results of the most important and classic algorithms, to which new approaches can be compared.

4.1 Dataset

The dataset consists of 210 videos of 30 different planar objects, 501 frames each, taken in the natural environment: on the streets, parks and inside buildings. Each object is recorded on 7 videos, one from each of the following challenging factor categories:

1. scale change (SC),
2. rotation (RT),
3. perspective distortion (PD),
4. motion blur (MB),
5. occlusion (OC),
6. out-of-view (OV),
7. unconstrained (UC).

The **UC** factor consists of all factors combined together.

4.2 Annotations

Additionally each video is provided with annotations of ground truth object position, i.e. positions of four corners on the frame and relative homographies between first and subsequent frames. For each video first frame is annotated as an initialization input for evaluated algorithms, and later every second frame as a ground truth for results comparison. For all videos frames with motion blur that is too strong for accurate annotation or such where recorded object is visible in less than 50%, are additionally labeled, to not be taken into account for the evaluation.

4.3 Metrics

To establish reliable quantitative comparison between the algorithms authors proposed two metrics for the evaluation:

Alignment error based on the four reference points (four corners of the object), and is defined as a square root of the mean square distances between the estimated positions of the points and their ground truth

$$e_{AL} = \left(\frac{1}{4} \sum_{i=1}^{4} \|\mathbf{x}_i - \mathbf{x}_i^*\|_2^2 \right)^{1/2},$$

where $\mathbf{x}_i$ is the position of a reference point and $\mathbf{x}_i^*$ is its ground truth position.

Homography discrepancy measures the difference between the ground truth homography T^* and the predicted one T, and it is defined as

$$S(T^*, T) = \frac{1}{4} \sum_{i=1}^{4} \left\| \mathbf{c}_i - (T^* T^{-1})(\mathbf{c}_i) \right\|_2,$$

where $\{\mathbf{c}_i\}_{i=1..4} = \{(-1,-1)^T, (1,-1)^T, (-1,1)^T, (1,1)^T\}$ are the corners of a square.

4.4 Selected Algorithms

In a similar manner as we did in the Chapter 3, the authors of [13] divided the algorithms selected for their comparison into two groups: keypoint-based and region-based.

4.4.1 Keypoint-based Algorithms

All the keypoint-based algorithms follow the same framework that we introduced in the Section 3.1, but with different approaches for keypoints detection and matching. SIFT [14] and SURF [3] are well known keypoints descriptors for many years

considered as a state-of-the-art. FERNS [16] and SOL [7] additionally use the fact that the template is known beforehand and are trying to learn the descriptor and matching offline.

4.4.2 Region-based Algorithms

Selected region-based algorithms are the same that we already mentioned in Section 3.2. ESM [4] proposes special optimization algorithm designed for planar objects tracking. GO-ESM [5] and SCV [18] are its extensions by modifying the error function - GO-ESM incorporates the information from the image gradients, SCV uses sum of conditional variance to ensure invariance to non-linear illumination changes. IC [1] approach modifies classic Lucas-Kanade image alignment algorithm by switching the role of the template and the image, what brings a constant Hessian and accelerates the optimization process.

Additionally, three generic object tracking algorithms were included in the benchmark: GPF [11], IVT [19] and L1APG [2]. However, as they are not designed for planar object tracking and homography estimation, their results in this problem are rather weak.

Chapter 5

Algorithm

Both classes of homography estimation algorithms have their strong and weak sides. Keypoint-based approaches rely heavily on the assumption that mostly the same keypoints are being detected on all the frames of the video. Detectors and descriptors try to ensure the number of invariances, but in really challenging conditions they can fail. Region-based methods can be really time efficient in tracking task, by skipping the computational heavy steps of keypoints detection and descriptors evaluation, but are really sensitive to occlusions and rely on the starting point of the optimization. When tracking is lost on one frame, then it has really small chances to get back on track.

Our method tries to combine the strong sides of both of these classes and avoid the weak ones. The general approach is presented in Algorithm 3.

We will now walk through all the steps of the algorithm to get the general idea and later focus on the technical details.

As a first step of the algorithm, we detect the keypoints on the provided template $\mathbf{T}$. We do it only once for the whole video. Next, in the same way as in benchmark [13], we assume the initialization on the first frame with a proper homography. Then, for all subsequent frames we perform the actual tracking. First we are searching for an initial homography estimate. Therefore we use the homography estimated for previous iteration, to project all detected keypoints from $\mathbf{T}$ on the previous frame. Then, with optical flow technique, we estimate the motion of these keypoints between previous and current frame. After that we try to refine position of each keypoint, based on image patches cross correlation. If these steps fail, we use additional relocalization technique based on generic Siamese Neural Network tracker [12], that performs more global but quite imprecise search. For successful search, we again try to refine the positions of the projected keypoints to acquire more accurate homography. If any of the two refinement steps is successful, then we estimate new homography based on such refined keypoints positions. If not, we keep the previous homography and start next iteration.

Algorithm 3 Our Algorithm

```
1: K ← keypoints detected on template T
2: H ← initialization on first frame
3: for frame F in all subsequent frames do
4:     Project all the keypoints from K on previous frame using H from previous iteration
5:     Using F and previous frame estimate the movement of the projected keypoints
6:     Refine positions of all projected keypoints
7:     if Tracking is lost then
8:         Try to relocalize
9:         if Relocalization successful then
10:            Refine positions of all keypoints from K
11:            Estimate homography H
12:        else
13:            Keep previous H
14:        end if
15:    else
16:        Estimate homography H
17:    end if
18: end for
```

Such approach gives us a lot of advantages. First, because we detect keypoints only on the template, the algorithm saves time and can run in over real-time on the ordinary laptop. By incorporating knowledge from previous frame it is possible to do only local search for keypoints refinement instead of costly matching step. But, to avoid the drawback most region-based methods suffer from, we introduce a relocalization technique, that is able to re-initialize the tracking even after being lost for few frames.

Now we will go through technical details of the most complicated steps of the algorithm.

5.1 Keypoints

For our approach we needed to choose a keypoints detector that would perform well with our tracking technique. For this reason, we follow [21], called after its authors, *Shi-Tomasi detector*.

As we mentioned in Section 3.1.1, best keypoints are often defined as corners and are found by looking for a maxima of a specially designed "cornerness" function. Such approach is used also in [21]. Informally, corner is a point on the image, which neighbourhood, when looking on it through a small window, changes significantly

when we shift this window in any possible direction. We will now try to formalize it. Let $I(x, y)$ be the image value in the point (x, y). Let W be a set of pixels that form a window around analyzed point. Let $(\Delta x, \Delta y)$ be considering shift. $E(\Delta x, \Delta y)$ is a value of the difference, that happens inside the window after the shift:

$$E(\Delta x, \Delta y) = \sum_{x_k, y_k \in W} \left(I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y)\right)^2. \tag{5.1}$$

We can approximate $I(x_k + \Delta x, y_k + \Delta y)$ using a Taylor's expansion in (x_k, y_k). Let I_x, I_y be the partial derivatives of the function I. Then:

$$I(x_k + \Delta x, y_k + \Delta y) \approx I(x_k, y_k) + I_x(x_k, y_k)\Delta x + I_y(x_k, y_k)\Delta y. \tag{5.2}$$

By using (5.2) inside (5.1) we get:

$$E(\Delta x, \Delta y) \approx \sum_{x_k, y_k \in W} \left(I_x(x_k, y_k)\Delta x + I_y(x_k, y_k)\Delta y\right)^2,$$

what in a matrix form can be viewed as:

$$E(\Delta x, \Delta y) \approx \begin{pmatrix} \Delta x & \Delta y \end{pmatrix} M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix},$$

where M is a second moments matrix:

$$M = \sum_{x_k, y_k \in W} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} = \begin{pmatrix} \sum_{x_k, y_k \in W} I_x^2 & \sum_{x_k, y_k \in W} I_x I_y \\ \sum_{x_k, y_k \in W} I_x I_y & \sum_{x_k, y_k \in W} I_y^2 \end{pmatrix}.$$

We can see that E is entirely defined by the matrix M, therefore we would like to have some simple condition on M that would ensure the desired property on E - that for every shift the difference in E is significant. Authors of [21] proposed a condition on the eigenvalues of M. The difference in every direction is big if both the eigenvalues of M exceed the given threshold. Therefore their "cornerness" function is defined as follows:

$$R = \min(\lambda_1, \lambda_2),$$

where λ_1 and λ_2 are the eigenvalues of M. The point is chosen to be a keypoint if it is a local maximum of R that exceeds the threshold.

5.2 Optical Flow

To utilize the information from previous frame, we could use the previous homography as an initial guess of the current solution. However, in faster movements it

would not be enough. For that reason, we additionally estimate the movement of the projected template's keypoints between previous frame and current. For that task we use an optical flow technique.

Optical flow is a motion of the objects through the frames of a video. We can think of it as vector field, that for each pixel of the frame, gives us the movement that this pixel takes up to the next frame. To estimate it, we need two assumptions:

- pixel intensity does not change between frames,
- movement in local neighbourhood is constant.

As these assumptions might not be true for the whole video, they should rather hold between two consecutive frames. Let x and y be image pixel coordinates and t be the time. Let $I(x, y, t)$ be the image intensity value. Let $\Delta x, \Delta y, \Delta t$ be the analyzed movement in space and time. Thanks to the first assumption we can write:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \tag{5.3}$$

By assuming the movement to be small, we can use the Taylor's expansion to get:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y \frac{\partial I}{\partial t}\Delta t. \tag{5.4}$$

By combining (5.3) and (5.4) and dividing by Δt we get:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y = -\frac{\partial I}{\partial t}, \tag{5.5}$$

where $V_x = \frac{\Delta x}{\Delta t}$ and $V_y = \frac{\Delta y}{\Delta t}$ are two unknowns being the searched optical flow values.

From one equation we cannot estimate two unknowns, therefore we need to use the second assumption, about the constant flow in local neighbourhood. Let $q_1, \ldots, q_n$ be the local neighbourhood of the pixel $(x, y)^T$. From that we can get the following system of equations:

$$\begin{cases} \frac{\partial I}{\partial x}(q_1)V_x + \frac{\partial I}{\partial y}(q1)V_y &= -\frac{\partial I}{\partial t}(q1), \\ \frac{\partial I}{\partial x}(q_2)V_x + \frac{\partial I}{\partial y}(q_2)V_y &= -\frac{\partial I}{\partial t}(q_2), \\ & \vdots \\ \frac{\partial I}{\partial x}(q_n)V_x + \frac{\partial I}{\partial y}(q_n)V_y &= -\frac{\partial I}{\partial t}(q_n). \end{cases}$$

We can write it in a matrix $Av = b$ form as:

$$\begin{pmatrix} \frac{\partial I}{\partial x}(q_1) & \frac{\partial I}{\partial y}(q_1) \\ \frac{\partial I}{\partial x}(q_2) & \frac{\partial I}{\partial y}(q_2) \\ \vdots & \\ \frac{\partial I}{\partial x}(q_n) & \frac{\partial I}{\partial y}(q_n) \end{pmatrix} \begin{pmatrix} V_x \\ V_y \end{pmatrix} = \begin{pmatrix} -\frac{\partial I}{\partial t}(q_1) \\ -\frac{\partial I}{\partial t}(q_2) \\ \vdots \\ -\frac{\partial I}{\partial t}(q_n) \end{pmatrix}.$$

The least squares solution can be obtained by:

$$\begin{pmatrix} V_x \\ V_y \end{pmatrix} = \begin{pmatrix} \sum_i \left(\frac{\partial I}{\partial x}(q_i)\right)^2 & \sum_i \frac{\partial I}{\partial x}(q_i)\frac{\partial I}{\partial y}(q_i) \\ \sum_i \frac{\partial I}{\partial x}(q_i)\frac{\partial I}{\partial y}(q_i) & \sum_i \left(\frac{\partial I}{\partial y}(q_i)\right)^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum_i \frac{\partial I}{\partial x}(q_i)\frac{\partial I}{\partial t}(q_i) \\ -\sum_i \frac{\partial I}{\partial y}(q_i)\frac{\partial I}{\partial t}(q_i) \end{pmatrix}. \quad (5.6)$$

It is important to see that the matrix being inverted in Equation (5.6), is the same second moments matrix, that we had in Shi-Tomasi keypoints detector, Equation (5.1). It is therefore well visible, that the solution to the optical flow equation (5.5) is well determined for points chosen to be the keypoints of the Shi-Tomasi detector.

5.3 Refinement Procedure

Optical flow can give us a coarse estimate of where the template's keypoints are located on the frame, but to get really accurate result, we need additional refinement step. The idea is shown in the Algorithm 4. To obtain keypoint's true position on

Algorithm 4 Refinement Procedure

1: Estimate coarse homography H_c from keypoints position after optical flow tracking
2: Warp the template $\mathbf{T}$ according to the coarse homography H_c
3: **for** all keypoints **do**
4: Obtain a window patch W_T around the keypoint on the warped template
5: Obtain a window patch W_F around the keypoint projected on the frame according to H_c
6: Obtain the optimal shift of W_T inside W_F so that patches have best overlapping
7: **end for**

the frame we try to match the small patch around this keypoint on the warped template with some patch in the neighbourhood of the keypoint projected on the frame. We assume, that our first homography estimate is close enough, so that it is sufficient to consider only translation of the template patch, without additional rotations and perspective transformations. It makes the solution space much smaller.

As a matching evaluation function, we used normalized cross-correlation, because it brings invariance to illumination changes, which might be crucial, because template is unchanged for the whole video and the light conditions on the scene may vary. We measure the similarity of the template's and frame's patches for the shift

$(\Delta x, \Delta y)$ according to the following formula:

$$R(\Delta x, \Delta y) = \frac{\sum_{x,y} W_T(x,y) \cdot W_F(x+\Delta x, y+\Delta y)}{\sqrt{\sum_{x,y} W_T(x,y)^2 \cdot \sum_{x,y} W_F(x+\Delta x, y+\Delta y)^2}}. \tag{5.7}$$

By considering only pure translation shifts and having a formula from Equation (5.7), we can very efficiently find the optimal shift by checking matching results for all possible shifts and taking the one giving the maximal value.

Such approach can give us the keypoint's position with accuracy of one pixel. However, sometimes it is not enough, especially in augmented reality applications without sub-pixel accuracy, the resulting homography might be trembling what makes it unreliable for a human eye. Therefore, to overcome this issue, we apply one last step of finding the sub-pixel maximum of patches matching. To achieve that, we fit a two-dimensional quadratic function in the 3×3 neighbourhood of the found shift, and obtain its maximum from the following formula:

$$(\Delta x_{opt}, \Delta y_{opt}) = -H^{-1}(\Delta x_{max}, \Delta y_{max}) \cdot \nabla R(\Delta x_{max}, \Delta y_{max}),$$

where $(\Delta x_{max}, \Delta y_{max})$ is the shift giving the maximal value obtained through patches matching, and ∇R and H are R function gradient and Hessian.

Refinement procedure gives us not only better positions of the points, but also a hint about which points might be localized incorrectly. If the value of the normalized cross-correlation for particular point does not exceed given threshold, the point is treated as an outlier, so it does not participate in the next homography estimation step.

5.4 Relocalization Procedure

Our approach to homography estimation relies heavily on the homography found for previous frame. While this helps us to find the accurate solution really fast, it becomes a serious drawback, when the tracking gets lost and our previous homography is far from the next one. To overcome this, we propose an additional relocalization procedure, that can give us a hint of current homography when being lost.

5.4.1 Lost State

However, the first thing we need is the ability to recognize, that we are in a lost state. For that case, the procedure is quite simple, we combine to conditions:

1. geometrically reliable homography

2. high overlapping score

The first condition checks if obtained homography makes geometrical sense, i.e. if it does not change the order of template's corners and if the perspective coefficients are not unlikely big. In such cases it is most probable that the keypoints' positions were wrongly estimated and the whole homography is incorrect. The second condition uses the same measure that we used for refinement, namely normalized cross-correlation (NCC) (5.7). We project the frame on the template with the inverse of obtained homography, and check the value of NCC measure. If it is low, it is likely that the homography is wrongly estimated.

When we know, we are in a lost state, we need to use the actual relocalization, to give us an idea where the searched planar object is. For that task we use a Siamese tracker.

5.4.2 Siamese Neural Networks Trackers

Siamese Neural Networks Trackers are a class o methods for solving a single object tracking problem. It is a more general task, than planar object tracking. The aim is for each frame of the video, to provide a bounding box in which tracked object is located. The result for planar object tracking is therefore much less accurate, but in exchange for that, the tracker should be much more robust to occlusions and motion blur - the two factors because of which the main tracking can most often fail.

The idea behind Siamese trackers is that they are pairs of twin convolutional neural networks, sharing parameters, able to compare their inputs looking for similarities, as shown on Figure 5.1. The first network takes the image of the target being tracked and produces a map of features of it, while the second analyzes the video frame and also produces the map of features. Two maps are later compared to find the area with biggest similarity. This region is the resulting bounding box of the tracker.

For our algorithm, we used a pretrained SiamRPN++ model [12] available in OpenCV GitHub repository[1]. The SiamRPN++ model follows the general idea of Siamese trackers mentioned before, with two upgrades. First is the usage of Region Proposal Network modules [17]. Instead of getting the bounding box from the raw cross-correlation of two feature maps, there are additional modules that analyze the feature maps and "propose regions" by assigning a probability of containing an object in a set of predefined bounding boxes. These modules are also capable of small adjustments of the corners of these predefined bounding boxes. The final result may be therefore much more accurate. The second modification is an introduction of very deep network as a shared model for twin networks. Earlier approaches

[1]OpenCV GitHub repo available at link: `https://github.com/opencv/opencv` (last access: 15.05.2021).

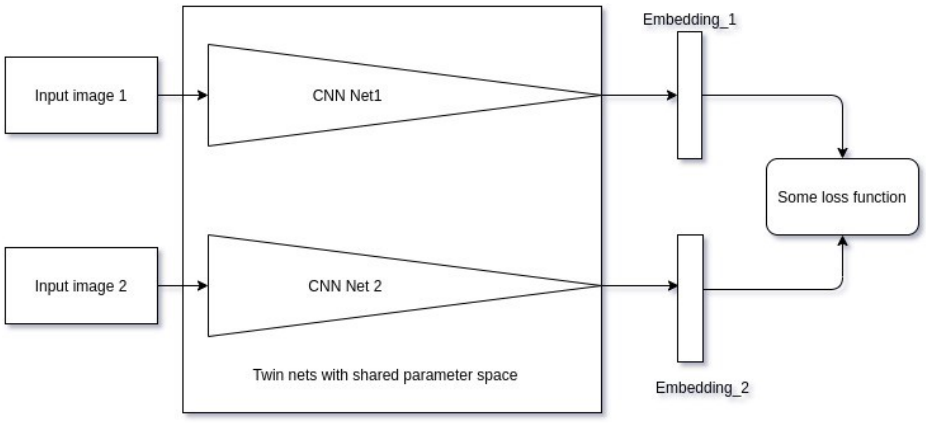


Figure 5.1: Siamese tracker consists of two twin convolutional neural networks sharing the parameters. The first network produces a map of features of the target, while the second analyzes the video frame and also produces the map of features. Two maps are later compared to find the area with biggest similarity. This region is the resulting bounding box of the tracker. Source: `https://medium.com/@reachraktim/object-tracking-with-siamese-networks-and-detectron2-572e04dac547` (last access: 11.11.2021).

used only a shallow networks, because of the restriction of translation invariance for a tracker to work. Authors of [12] propose a spatial aware sampling strategy for network training, and achieve strong enough invariance. Using deeper network for features extraction allows the whole model to achieve much better results than previous approaches.

When our algorithm finds itself in a lost state, the relocalization with SiamRPN++ tracker is performed. Resulting bounding box is used to compute the initial homography, that maps the template image to the bounding box. Based on this homography, refinement procedure is used to establish better positions of the keypoints and better resulting homography.

5.5 Time Performance

To measure the time performance of our algorithm we should split it into two parts: tracking and relocalization procedures. Our tests were performed on a Dell XPS 15 7590 laptop, with an Intel i7-9750H CPU and Nvidia GeForce GTX 1050 Ti Max-Q 4GB GPU devices.

When it comes to tracking, our algorithms shows very good time performance, able to run with 35-50 FPS on 1280×720 resolution videos on CPU only, depending on the template image size. With relocalization switched on and neural network run on the GPU, the algorithm still was able to run in real-time, achieving on

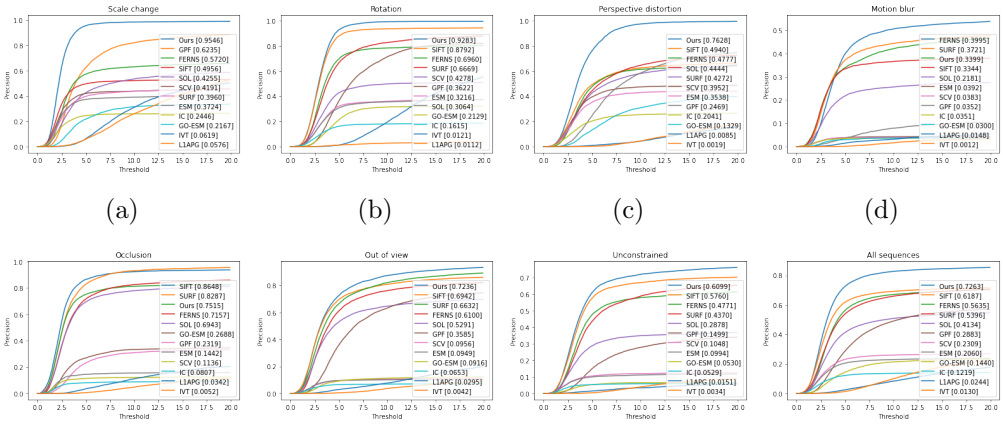
average 30-40 FPS. However, with the lack of a GPU, one frame processing time with relocalization increases from 20-30 ms to over 100 ms, achieving less than 10 FPS, what might be to low for some applications. In such scenarios it is still possible to run the algorithm in real-time, if the need for relocalization is not too frequent. When the tracking is lost, the relocalization procedure can be run in the background and incoming frames saved in a queue. After successful relocalization, tracking procedure is fast enough to catch up by processing all the frames from the queue. In such setup, for all the frames in the queue except last, the result is not returned on time, so the whole application would need to be in the "lost" state for that time.

Chapter 6

Results

We evaluated our algorithm's performance on all the videos from the Planar Object Tracking benchmark introduced in Chapter 4, following the same rules as for other algorithms in [13].

To establish reliable quantitative comparison with other methods we used the same **alignment error** and **homography discrepancy** metrics. Additionally we draw precision and success plots which correspond to the ratio of frames tracked with, respectively alignment and homography errors, smaller than the threshold t. As a representative precision and success scores $t_p = 5$ and $t_s = 10$ are taken respectively.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6.1: Comparison of our and all trackers from [13] using precision plots. The precision at the threshold $t_p = 5$ is used as a representative score. It can be seen that our approach outperforms all other trackers by a significant margin in most of the cases, especially in scale change and perspective distortion. Only in occlusion and motion blur our algorithm takes third position, however the difference to the two best is not significant.

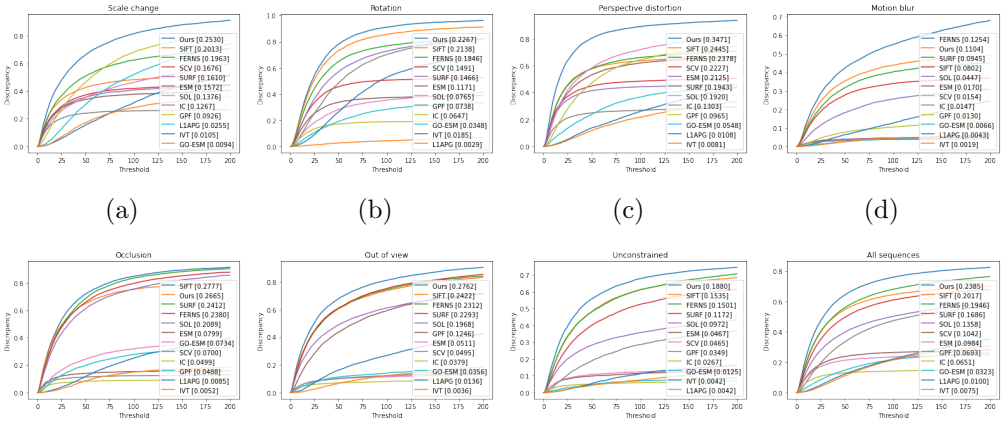
As can be seen on Figures 6.1 and 6.2 and what is also summarized in Tables 6.1

Algorithm	SC	RT	PD	MB	OC	OV	UC	All
SIFT	0.4956	0.8792	0.4940	0.3344	**0.8648**	0.6942	0.5760	0.6187
SURF	0.3960	0.6669	0.4272	0.3721	0.8287	0.6632	0.4370	0.5396
FERNS	0.5720	0.6960	0.4777	**0.3995**	0.7157	0.6100	0.4771	0.5635
SOL	0.4255	0.3064	0.4444	0.2181	0.6943	0.5291	0.2878	0.4134
ESM	0.3724	0.3216	0.3538	0.0392	0.1442	0.0949	0.0994	0.2060
GO-ESM	0.2167	0.2129	0.1329	0.0300	0.2688	0.0916	0.0530	0.1440
SCV	0.4191	0.4278	0.3952	0.0383	0.1136	0.0956	0.1048	0.2309
IC	0.2446	0.1615	0.2041	0.0351	0.0807	0.0653	0.0529	0.1219
GPF	0.6235	0.3622	0.2469	0.0352	0.2319	0.3585	0.1499	0.2883
IVT	0.0619	0.0121	0.0019	0.0012	0.0052	0.0042	0.0034	0.0130
L1APG	0.0576	0.0112	0.0085	0.0148	0.0342	0.0295	0.0151	0.0244
Our	**0.9546**	**0.9283**	**0.7628**	0.3399	0.7515	**0.7236**	**0.6099**	**0.7263**

Table 6.1: Precision scores of all the algorithms achieved on videos with particular challenging factor taken with precision threshold $t_p = 5$. In most of the cases our method outperforms the others by a significant margin.

Algorithm	SC	RT	PD	MB	OC	OV	UC	All
SIFT	0.2013	0.2138	0.2445	0.0802	**0.2777**	0.2422	0.1535	0.2017
SURF	0.1610	0.1466	0.1943	0.0945	0.2412	0.2293	0.1172	0.1686
FERNS	0.1963	0.1846	0.2378	**0.1254**	0.2380	0.2312	0.1501	0.1946
SOL	0.1376	0.0765	0.1920	0.0447	0.2089	0.1968	0.0972	0.1358
ESM	0.1572	0.1171	0.2125	0.0170	0.0799	0.0511	0.0467	0.0984
GO-ESM	0.0094	0.0348	0.0548	0.0066	0.0734	0.0356	0.0125	0.0323
SCV	0.1676	0.1491	0.2227	0.0154	0.0700	0.0495	0.0465	0.1042
IC	0.1267	0.0647	0.1303	0.0147	0.0499	0.0379	0.0267	0.0651
GPF	0.0926	0.0738	0.0965	0.0130	0.0488	0.1246	0.0349	0.0693
IVT	0.0105	0.0185	0.0081	0.0019	0.0052	0.0036	0.0042	0.0075
L1APG	0.0255	0.0029	0.0108	0.0043	0.0085	0.0136	0.0042	0.0100
Our	**0.2530**	**0.2267**	**0.3471**	0.1104	0.2665	**0.2762**	**0.1880**	**0.2385**

Table 6.2: Success scores of all the algorithms achieved on videos with particular challenging factor taken with homography discrepancy threshold $t_s = 10$. In most of the cases our method outperforms the others by a significant margin.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6.2: Comparison of our and all trackers from [13] using success (homography discrepancy error) plots. The error at the threshold $t_s = 10$ is used as a representative score. Similarly as with precision measure, it can be seen that our approach outperforms all other trackers by a significant margin in most of the cases, especially in scale change and perspective distortion. Here, for motion blur and occlusion factors, our algorithm takes second position, being slightly worse than the best one.

and 6.2, our approach significantly outperforms other trackers on the average of all sequences but also in most of the separate cases, especially in scale change and perspective distortion. In these cases our algorithm heavily relies on good estimates from previous frame and initially tracked points. In such conditions our keypoints matching technique is highly robust to even extreme scale and perspective distortions. Few examples of our method's results can be seen on the Figures 6.3, 6.4 and 6.5.

When it comes to motion blur it appears that Ferns [16] and SURF [3] achieve better precision than our approach. It can be explained by the fact that our algorithm relies on optical flow computed between subsequent frames and such flow

(a) First frame (b) Frame with rotation and motion blur

Figure 6.3: Comparison of first frame of video V01_7 from [13] with a frame with a significant rotation and motion blur present. Estimated object position is marked with red quadrilateral.

(a) First frame (b) Frame with perspective distortion

Figure 6.4: Comparison of first frame of video V04_3 from [13] with a frame with present projective distortion and some reflecting artifacts. Estimated object position is marked with red quadrilateral.

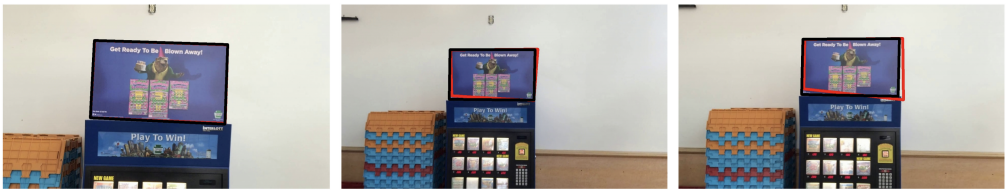
(a) First frame (b) Frame with scale change

Figure 6.5: Comparison of first frame of video V21_1 from [13] with a frame with present remarkable scale change. Estimated object position is marked with red quadrilateral.

might be much inaccurate in the presence of significant blur. It could be overcompensated by the keypoints patches matching step if we additionally estimated the blur kernel of the motion and applied it to patches to try to match their burred versions.

Second challenging factor with which our algorithm is not the best one is the occlusion. Similarly to motion blur, the optical flow estimation is one of the weak points here, because it may be easily misled by moving object in front of the tracked surface. Additionally, our procedure, responsible for noticing the lost state, can faultily decide that the tracking is lost because of smaller NCC score in the presence of occluding object despite the correctly estimated homography.

Our algorithm performed relatively better in success ranking than in precision ranking. Homography discrepancy with threshold $t_s = 10$ is a very tight measure, as pointed out by the authors of [13] and shown on the Figure 6.6. It means, that in not very hard conditions, when precise homography is possible to estimate, our solution is more accurate than other approaches and, thanks to that, makes up the difference from inaccurate estimations in the presence of significant blur and occlusion. This is a very important factor, when choosing the algorithm to apply in practice, when visual reliability is highly required.



(a) 8.05 (b) 85.75 (c) 315.75

Figure 6.6: Examples of discrepancy scores (shown under subfigures) in different cases. It can be seen, that almost unnoticeable difference in example (a) has the error equal to 8.05. For slight, but noticeable difference in example (b) the error takes the value of 85.75, which is high above the threshold $t_s = 10$.

As authors showed in [13], and what can be seen on Figure 6.1, keypoints based approaches achieved vastly better results than region based trackers. The best on the average from this group, GPF [11], obtained precision score equal to 0.2883 for $t_p = 5$, what puts it far behind the top trackers. One of the most significant reasons of this weakness is the fact, that region based trackers relies on transformation estimated on preceding frame and are unable to perform more global search. That results in large difficulties in recovery after lost tracking. In comparison, our approach, which also makes the usage of preceding transformation estimates, is resistant to such behavior because of a robust relocalization technique and thus, can save much time without performing the global transformation search.

What makes our algorithm the best candidate to be used in practice is its performance in the unconstrained scenarios and the average result on all sequences.

One can expect it to behave most robustly in the common environment of the real life applications. As stated in Section 5.5, our algorithm can be run in real-time on mid-range hardware, in opposite to the top alternatives like SIFT.

Chapter 7

Conclusions and Further Work

In this thesis we introduce the planar object tracking problem, as well as a theoretical background for this task. We also discuss existing state-of-the-art field solutions with a benchmark on which it is possible to evaluate them. Finally, we propose our approach together with a comparison to previously described algorithms.

Our solution draws from both keypoint-based and region-based methods, combining their strong sides. On the discussed benchmark, which tries to provide scenarios from real life cases, our approach outperforms other state-of-the-art methods, showing best robustness and accuracy. What is more, it is also very efficient, being able to run in real-time, what is not always the case with the other algorithms. Presented results prove that our solution is the best candidate to be used in practice.

Besides the algorithm, we also propose a relocalization technique, that, when applied to region-based approaches, can fill the gap between them and keypoint-based methods, which is very significant as stated by the authors of [13]. Further analysis of how much this procedure can improve the results is however beyond the scope of this work.

Despite being the best among the compared approaches, our algorithm still can be improved, as shown by the results presented in Chapter 6. The hardest challenging factor for our, but also all the other algorithms, is motion blur. However, non of the presented solution, tries to address this problem directly. One could imagine an additional mechanism able to estimate the blur kernel and utilizing it by deblurring the video frame or blurring the template image, so that it is more similar to this on the scene. Such approach could even further strengthen the results of our semi-region-based method in comparison to the keypoint-based ones, because on blurred images it is be much harder to detect any keypoints, while our approach would just compare patches of similarly blurred areas.

Proposed relocalization procedure can be also improved. First of all, instead of generic object neural network tracker one could use an architecture specialized for planar objects and trained on such examples. Secondly, convolutional neural

networks are not covariant to the rotations by definition. In practice one usually tries to achieve some kind of invariance by data augmentation. In our case however, it would be possible to provide to the network appropriately rotated video frame, so that the template recorded on the scene would have similar orientation to the reference template image. Such approach could increase the number of successful relocalizations.

Bibliography

[1] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework Part 1: The Quantity Approximated, the Warp Update Rule, and the Gradient Descent Approximation". In: *International Journal of Computer Vision - IJCV* (Jan. 2004).

[2] Chenglong Bao et al. "Real time robust L1 tracker using accelerated proximal gradient approach". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 1830–1837.

[3] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142.

[4] 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. 2004, 943–948 vol.1.

[5] Lin Chen et al. "Illumination insensitive efficient second-order minimization for planar object tracking". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 4429–4436.

[6] M. Fischler and R. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[7] Sam Hare, Amir Saffari, and Philip H. S. Torr. "Efficient online structured output learning for keypoint-based object tracking". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 1894–1901.

[8] C. G. Harris and M. Stephens. "A Combined Corner and Edge Detector". In: *Alvey Vision Conference*. 1988.

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

[10] Lai Kang et al. "A highly accurate dense approach for homography estimation using modified differential evolution". In: *Engineering Applications of Artificial Intelligence* 31 (2014). Special Issue: Advances in Evolutionary Optimization Based Image Processing, pp. 68–77. ISSN: 0952-1976.

[11] Junghyun Kwon et al. “A Geometric Particle Filter for Template-Based Visual Tracking”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.4 (2014), pp. 625–643.

[12] Bo Li et al. *SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks*. 2018.

[13] Pengpeng Liang et al. *Planar Object Tracking in the Wild: A Benchmark*. 2018.

[14] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691.

[15] Krystian Mikolajczyk and Cordelia Schmid. “Scale and Affine Invariant Interest Point Detectors”. In: *International Journal of Computer Vision* 60 (Oct. 2004), pp. 63–86.

[16] Mustafa Özuysal et al. “Fast Keypoint Recognition Using Random Ferns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2010), pp. 448–461.

[17] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016.

[18] Rogério Richa et al. “Visual tracking using the sum of conditional variance”. English (US). In: *IROS’11 - 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE International Conference on Intelligent Robots and Systems. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems: Celebrating 50 Years of Robotics, IROS’11 ; Conference date: 25-09-2011 Through 30-09-2011. Dec. 2011, pp. 2953–2958. ISBN: 9781612844541.

[19] David A. Ross et al. “Incremental Learning for Robust Visual Tracking.” In: *International Journal of Computer Vision* 77.1-3 (2008), pp. 125–141.

[20] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571.

[21] Jianbo Shi and Tomasi. “Good features to track”. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 1994, pp. 593–600.