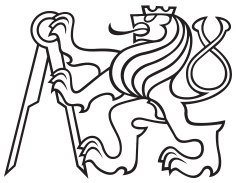
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Coin-Tracking - Double-Sided Tracking of Flat Objects

Jonáš Šerých

Supervisor: Prof. Ing. Jiří Matas, Ph.D.
Field of study: Computer Vision and Image Processing
January 2018

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Šerých Jonáš** Personal ID number: **406478**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer Vision and Image Processing**

II. Master's thesis details

Master's thesis title in English:

Coin-Tracking - Double-Sided Tracking of Flat Objects

Master's thesis title in Czech:

Coin-Tracking - Oboustranné sledování plochých objektů

Guidelines:

1. Consider tracking of objects that are approximately flat in sequences where both sides of such objects are visible in turn. The "approximately flat" means that it is possible to assume that the boundary of the two sides is visible, barring occlusion by another object, irrespective of the pose, together with one of the sides. The other side is fully self-occluded.
2. Propose and implement a coin-tracking algorithm. The algorithm must output not only the position and segmentation of the object in an image, but also identify the visible side.
3. Consider two variants of the problem:
a) both sides of the object are known before tracking start,
b) only one side of the object is known, e.g. the one visible in frame 1.
4. Address the following problems:
a) representation of the appearance of the object, i.e. its two sides and the occluding contour,
b) representation of the pose of the object,
c) modelling of the 3D orientation of the object.
5. Collect a set of cointracking sequences.
6. Evaluate the performance of the proposed algorithm.

Bibliography / sources:

[1] S. Caelles, K.K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, L. Van Gool - One-Shot Video Object Segmentation, Computer Vision and Pattern Recognition (CVPR), 2017
[2] P. Voigtlaender and B. Leibe - Online adaptation of convolutional neural networks for video object segmentation - BMVC, 2017
[3] A. Khoreva, R. Benenson, E. Ilg, T. Brox and B. Schiele - Lucid Data Dreaming for Object Tracking - The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops

Name and workplace of master's thesis supervisor:

prof. Ing. Jiří Matas, Ph.D., Visual Recognition Group, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **25.09.2017** Deadline for master's thesis submission: **09.01.2018**

Assignment valid until: **17.02.2019**

prof. Ing. Jiří Matas, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____ _____

Date of assignment receipt | Student's signature

Acknowledgements

I would like to express my gratitude to my supervisor prof. Ing Jiří Matas, Ph.D., for his valuable advice and guidance. My thanks also go to my family and my friends, who supported me the whole time. Lastly, I cannot thank my wife Anička enough for all the support she gave me and for her great patience.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 9th January 2018

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 9. ledna 2018

Abstract

The thesis introduces a novel type of visual tracking problem, coin-tracking, in which the objects being tracked are approximately flat, meaning that only one of the object's two sides can be visible at any given time, as the other side is fully self-occluded. It also holds, that the boundary between the object's two sides is always visible, except for occlusions by other objects. Because of the inherent properties of the coin-tracking sequences, the standard visual tracking algorithms are not suitable.

We analyse the problem and propose an coin-tracking algorithm that combines appearance-based deep neural network with a classical shape-based object classification approach and an estimator of the object pose in order to provide a segmentation mask and a visible side indicator for each frame of the input video sequence.

The presented algorithm is evaluated on a coin-tracking dataset, that we have collected and annotated with bounding boxes and visible side labels.

Keywords: tracking, segmentation, deep neural network, coin-tracking

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

Abstrakt

Diplomová práce prezentuje nový typ problému sledování, nazvaný coin-tracking, ve kterém jsou sledované objekty přibližně ploché, což znamená, že je v každém okamžiku vidět pouze jedna z jejich dvou stran, jelikož druhá strana je zcela zakryta tou viditelnou. Pro takovéto objekty platí, že je hranice mezi jejich stranami vždy viditelná, až na případné překryvy jinými objekty. Použití standardních algoritmů pro sledování objektů ve videu není za těchto podmínek vhodné.

V práci analyzujeme coin-tracking problém a navrhujeme pro jeho řešení algoritmus, který kombinuje hlubokou neuronovou síť pro segmentaci a odhad viditelné strany podle vzhledu objektu s klasickou metodou založenou na rozpoznávání objektů podle jeho tvaru a s algoritmem odhadujícím 3D pózu objektu. Tento algoritmus poskytuje pro každý snímek vstupního videa na výstupu segmentační masku objektu společně s identifikátorem viditelné strany.

Kvalitu navrženého algoritmu jsme otestovali na datové sadě videí pro coin-tracking, kterou jsme sestavili a oanotovali bounding boxy a indikátorem viditelné strany objektu.

Klíčová slova: tracking, segmentace, hluboké neuronové sítě, coin-tracking

Překlad názvu: Coin-Tracking - Oboustranné sledování plochých objektů

Contents

Figures

Tables

Chapter 1

Introduction

Visual tracking is a fundamental problem in the field of computer vision, with various applications in autonomous driving, human-computer interaction, surveillance, sport match analysis, video post-production and other. Given a video sequence and some object marked on the first frame, the task of a tracking algorithm is to output the pose of the object in all the subsequent frames of the video. Both the annotation of the first frame and the nature of the object pose representation may vary from a simple (x, y) coordinate of the object center, an axis-aligned or arbitrarily rotated object bounding box, up to a full pixel-dense object segmentation.

Depending on the available knowledge about the objects being tracked, the problem can be divided into two classes. Commonly a *model-free* variant is considered, which has the goal of tracking arbitrary objects, without any prior knowledge of their type, 3D shape or other properties. Model-free trackers usually operate directly on the image, without having an internal representation that would capture the object 3D spatial properties. This problem is extensively studied and standard benchmarks ([5, 6]) exist to evaluate the trackers' performance on challenging video sequences.

Alternatively, a tracking with *model* can be studied, where a 3D model of the tracked object is either known in advance (e.g. a CAD model of a manufactured part) , or reconstructed from the video. Such problem can be formulated in terms of *structure from motion* - SfM ([1], which is another classical computer vision problem.

1.1 Thesis contributions

We propose a novel tracking problem called "coin-tracking", which lies on the boundary of the two mentioned classes of the visual tracking problem. In coin-tracking, the objects of interest are known to be *approximately flat* and *two-sided*. Many real-world objects have these two properties, including items of everyday use like smartphones, credit cards, books, magazines, plates and coins (hence the problem name), various sports equipment such as surfboards, ice-hockey sticks and table tennis rackets, tools like knives, hand saws, scissors, and pliers and other objects, including doors, window shutters, sails, wings, propeller blades, playing cards, poker chips and more.

Coin-like objects have interesting properties. Their occluding contour is always visible if not for occlusion by other object, and the object images are related by homographies and thus no 3D model is necessary to represent the object. often they undergo out-of-the-plane rotation, which the objects present in the currently available tracking datasets do not. Consider a general 3D object rotating in a video, the profile views and the view of the back side of the object are revealed continuously when it rotates, which is not the case for a coin-like object, where only one side is visible at any single moment. When a coin-like object rotates out-of-the-plane, only it's front face is visible for some time, then a flip happens at which moment only the boundary between the sides is visible and then the second side appears suddenly. The difference between these two cases is demonstrated in figure 1.2. More detailed introduction into the problem and its properties is given in chapter 2.

Based on the problem analysis, we propose a coin-tracking algorithm, which consists of three components - segmentation and visible side classification deep neural network, a classical moment-based shape classification method and an a object pose estimator - all combined together in a meaningful way. Example of the output of the proposed algorithm is shown in figure 1.1.

We have collected a coin-tracking dataset consisting of 22 video sequences and corresponding ground truth annotations. The performance of the proposed coin-tracking algorithm and its components was evaluated on the dataset and the results are summarized in chapter 7. The dataset and the outputs of our method are provided on the CD accompanying this thesis.

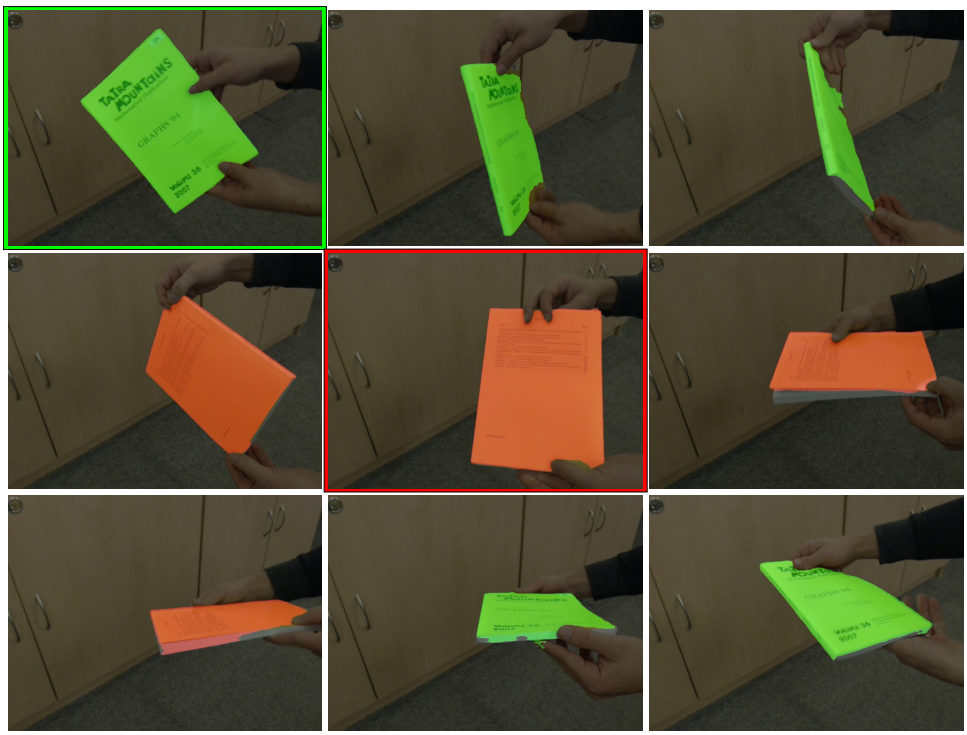


Figure 1.1: An example output of our algorithm. The algorithm was trained from two annotated frames (images in the green, respectively red frame). It outputs the object segmentation as well as the visible side identification (visualised by the segmentation overlay color). Note that the object is originally yellow from both sides.

(a) : 0° **(b) :** 45° **(c) :** 90° **(d) :** 135° **(e) :** 180°

Figure 1.2: Comparison of out-of-the-plane rotation of a general object (top) and an ideal coin-type object (bottom). Notice the abrupt change of the object appearance between b and d as well as the coin disappearing completely in c.

Chapter 2

Coin-tracking

We define cointracking 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. Moreover, the boundary between these 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 the tracked object is thus fully characterized by a visible side indication and a homography transformation to a canonical frame as discussed in section 2.1.1. However, because of possible object symmetries, the transformation might not be unique and thus we characterize the object state by a visible side indication and a segmentation mask instead.

Task definition. The inputs of a coin-tracking algorithm are the following: a video sequence, a segmentation of the **obverse** side of the tracked object on some frame and possibly a segmentation of the **reverse** side of the object on some other frame. Given this information, the algorithm is to output a segmentation mask (a binary image with the size of the frame, one indicates the object, zero the background) on each frame of the sequence. Moreover, the algorithm has to output a side indicator (**obverse** or **reverse**) for each frame.

2.1 Problem analysis

The object side can be classified based on several complementary clues. First, the two sides can be distinguished purely by the *appearance* as the color and texture characteristics can be different on each one. The second source of information is the object *dynamics* in the video sequence. If the side flips can be detected, it is possible to predict the currently visible side solely by counting the number of flips. Even if the flip detection is not reliable, the history of the object motion is an important piece of information. Finally, the object *shape* can be used, as the coin-like objects are rigid and the boundary between the two sides is visible at all times, up to occlusion. The occluding boundary undergoes a mirroring transformation between the **obverse** and the **reverse** side view and the presence of such mirroring with respect to the prototype **obverse** annotation can be detected, yielding a visible side classifier.

With these possibilities in mind, we choose the tracking by segmentation approach. In contrast to the bounding box output by the commonly used state-of-the art tracking by detection algorithms, a per-pixel segmentation makes it possible to exploit all of the three described feature types - appearance, dynamics and shape.

Figure 2.1: Two views of a poker chip, in fact each one of a different side, the **obverse** side on the left and the **reverse** side on the right.

The use of each of these features have limitations as discussed more in detail in the next sections. For example, the appearance fails to provide any usefull information if the object sides have nearly indistinguishable color and texture, which is the case with e.g. poker chips as shown in figure 2.1. Even if the **reverse** side is clearly distinguishable from the **obverse** by their

appearances, the shape can fail to be a good side discriminator, because of the object symmetries as shown in 2.2. In particular, it is impossible to deduce the currently visible object side solely from its shape if the shape is reflection symmetric along any axis. That is because the shape of any coin-like object **reverse** side is just a mirroring of the **obverse** side shape. It follows, that in case of symmetric object, the **reverse** shape is exactly the **obverse** shape and thus no decision can be made based on it.

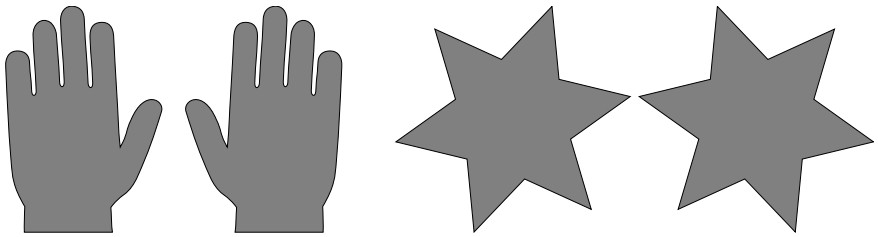


Figure 2.2: The currently visible object side can be recognized from its occluding contour for non-symmetric objects. Note that the only way to transform the first hand into the second one is mirroring. It is impossible to tell if the second star is a mirroring of the first one or its in-plane rotation.

Taking these failure modes into account, we can now analyse the possible situations arising in coin-tracking. From the appearance point of view, the object sides can range anywhere from being indistinguishable to being completely different, the object shape can be either symmetric or non-symmetric and on top of that two different scenarios have to be considered based on the available algorithm inputs - either both **obverse** and **reverse** training examples are available, or only the **obverse** one.

As our algorithm rely on the tracked objects segmentation, the **obverse**-only variant of the coin-tracking problem depends on the ability of the segmentation method to segment the **reverse** side without being trained on it. From this point of view, the objects with sides of nearly indistinguishable appearance are optimal, because the segmentator has no issues segmenting both sides.

2.1.1 Coin-tracking image formation

When the pin-hole camera image acquisition model is used, images of planes in the scene are related by a particular kind of transformation – a homography. More specifically, images of points on a plane can be transformed by a homography, to get the images in the other view as described in [1], chapter 13 and visualized in figure 2.3.

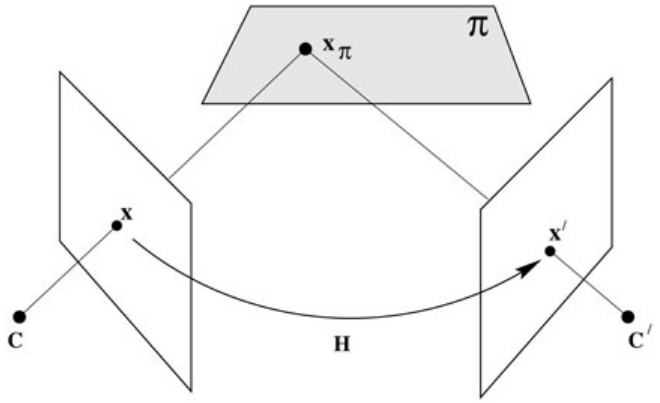


Figure 2.3: Homography between images of the plane π. [1]

An ideal coin-like object is a subset of some plane in space and therefore its images in a video sequence are related by homographies. With the prototype shape of the object obverse side S, the object shape can be modeled as

$$S' \simeq \mathbf{H}S$$

in each frame of the sequence. The homography is not a linear transformation, but it can be linearly approximated by affine transformation, yielding

$$S' \approx \mathbf{A}S$$

. The approximation is reasonable, if the object is small compared to its distance from the camera. In coin-tracking, the transformation $\mathbf{A}$ is often not close to a similarity transformation (translation, rotation and scale) and performs strongly anisotropic scaling, caused by out-of-plane rotations. This causes issues for current state-of-the art trackers, which are usually correlation filter based, giving us another motivation for use of a segmentation based approach.

The effect of out-of-the-plane rotation is particularly hard to deal with when the tracked object is flat. While some parts of the tracked object might be still visible when it is rotating out-of-the-plane by 90, it is not the case with flat objects, specifically, a perfectly flat object can be rotated in such a way, that none of it sides is visible as illustrated in figure 1.2.

Chapter 3

Segmentation

As discussed in 2.1, we have chosen to approach the coin-tracking problem by tracking by segmentation. Recently, convolutional neural networks (CNNs) have scored a great success in computer vision, overperforming the previous state-of-the art methods by large margin. Since the 2012 paper [7], deep neural networks became the main choice for many computer vision tasks, including image classification, object detection, semantic segmentation, human pose estimation and others.

In this chapter we will briefly introduce convolution neural networks, review the current state-of-the-art of tracking by segmentation and propose a segmentation method for our coin-tracking algorithm.

3.1 Convolutional neural networks

Artificial neural networks are machine learning systems inspired by the biological networks of neurons inside animal brains. They represent a mapping $f : X \mapsto Y$ between some input $X \subseteq \mathbb{R}^N$ and output $Y \subseteq \mathbb{R}^M$ real-numbered spaces.

The mapping is performed by a series of L interconnected *layers*, each performing a function $l_i : \mathbb{R}^{n_i} \mapsto \mathbb{R}^{m_i}$. The output y of the whole neural network applied on input x can then be written in terms of these layers as

$$y = l_L(l_{L-1}(\dots l_1(x)))$$

The layers functions are parametrized by parameters θ_i, which are learnt from training data. The training is formulated as an optimization problem

$$\min_{\theta} \sum_{(x,y)\in D} L(f_\theta(x), y)$$

For this optimization a training set $D = (x_i, y_i)_{i\in\{0,\cdots,T\}}$ has to be provided as well as some appropriate *loss function* $L : Y \mapsto \mathbb{R}$

This optimization problem can be solved by *backpropagation* method. The method is based on gradient descent and works in a two-phase *forward-backward* cycle. In the forward phase a network output $\bar{y} = f(x_i)$ is computed for a training sample, followed by the computation of the loss $L(\bar{y}, y)$. In the backward pass, the gradient $\frac{\partial L}{\partial \theta_i}$ of the loss is computed with respect to parameters θ of each layer. Thanks to the chain-rule, these derivatives can be computed as

$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial l_L}\frac{\partial l_L}{\partial l_{L-1}}\cdots\frac{\partial l_i}{\partial \theta_i} \tag{3.1}$$

The parameters are then updated by stochastic gradient descent (SGD) algorithm or its variant.

3.1.1 Layer types

Several types of layer functions are commonly combined inside a neural network. A *fully-connected layer* of input dimension m and output dimension n is a function $l_{\mathrm{FC}} : \mathbb{R}^m \mapsto \mathbb{R}^n$, which has a form of

$$l_{\mathrm{FC}}(x) = \mathbf{W}x + b$$

where $\mathbf{W}$ is a $n \times m$ weight matrix and b is a $\mathbb{R}^m$ bias vector.

A non-linear *activation function* is usually placed behind such layer, because a network composed only of the linear fully-connected layers would not be able to represent any non-linear functions. Several activation functions are commonly used, such as logistic function 3.3, or a Rectified Linear Unit - ReLU 3.4.

$$l_{\mathrm{logistic}}(x) = \frac{1}{1+e^{-x}} \tag{3.3}$$

$$l_{\mathrm{ReLU}}(x) = \max(0, x) \tag{3.4}$$

In the past, simple neural networks composed from fully-connected layers and nonlinearities were successfully used to solve relatively simple tasks. However, such networks are not usable for modern computer vision tasks. With image sizes of millions pixels, the number of parameters of a fully-connected layer, $(n+1)m$, would be too big to be practically usable. Instead, a *convolution* layer can be used, which reduces the number of necessary parameters significantly, enabling use of very deep (i.e. having large number of layers) so-called *convolutional neural networks* (CNNs).

When dealing with images, the inputs of the convolution layer are representing $H \times W \times D_{\text{in}}$ array, which can be also viewed as a $H \times W$ image with D_{in} channels. The layer outputs are computed by convoluting the inputs with D_{out} convolution kernels of size $K \times K \times D_{\text{in}}$ and stacking all of the $D_{textout}$ $H \times W$ outputs along the third dimension (Note that the output spacial dimension may be slightly different depending on the convolution method). The filter coefficients are the learnable parameters of the layer.

As we have stated previously, the fully-connected layers need large number of parameters, moreover, as opposed to the convolution layers, they do not directly utilize the fact, that low-level information in images is highly local. The convolution layers, on the other hand, were designed to capture the local relations as inspired by human vision. When many of these layers are stacked on top of each other, the resulting field-of-view (input image area having effect on the layers output) of the deeper layer is large enough to capture less localized image information.

In order to further increase the field-of-view of the deeper layers and to get robust to translations, the *pooling* layers, which downsample the signal, are commonly put after a block of convolution layers. As with the other layer types, there are many options of a pooling layer design, the most popular being *max-pooling*. It works by reducing the size of the input feature map by a strided pass of $K \times K$ filter, outputting the maximal value in each sampled $K \times K$ region. With a commonly used stride choice of 2, the output feature map has dimensions $\frac{H}{2} \times \frac{W}{2} \times D$, greatly reducing the computation complexity of the layers following the pooling layer. The localization accuracy of the consequent layers is reduced as well, but that is in agreement with the goals of image classification, where the network output should be invariant to shifts (e.g. an image of a cat should be classified as a cat regardless of the cat's position in the photo).

A simple CNN "LeNet" designed by Lecun *et al.*[2] in 1998 achieved an error rate under one percent on handwritten digit recognition task, with input images of 32×32 pixels. However, the convolutional neural networks gained on popularity for other more complicated computer vision tasks 14

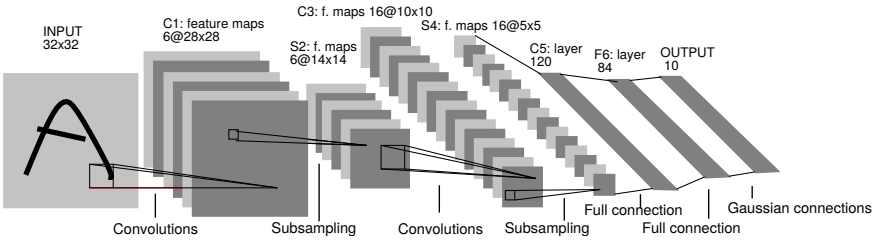


Figure 3.1: LeNet, an example of a CNN for handwritten digit recognition [2]

years later, with the success of AlexNet [7] on ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [8]. Since then, the neural networks could be made much deeper thanks to the available hardware computational power and thanks to clever design decisions. The main principles, however are very similar to LeNet. LeNet architecture is shown on figure 3.1.

3.2 DNN Segmentation literature review

In the previous section, we have provided an overview of the convolutional neural networks. In recent years, most of the state-of-the-art methods in computer vision were based on them. However, the success of supervised deep learning is conditioned on availability of large annotated datasets, which are difficult to obtain. Datasets related to the topic of this thesis can be split into three main categories, increasing in difficulty of annotation: those for image classification, object detection and semantic segmentation.

Image classification. The task of image classification is to assign one (or several) of K known classes to an input image, meaning that a dataset for training an image classification neural network must contain images paired with their annotated classes. The introduction of the ImageNet dataset[8] in 2009 enabled the first significant advances of deep learning in image classification[7].

Object detection. Unlike the image classification task, in object detection there might be multiple objects present in the image, each of them from different class. For each image all of the objects of each known class have to be annotated by a bounding box. Such annotation is much more time consuming and expensive. Su et al.[9] has measured the average time needed for annotating a single bounding box on ImageNet images to be 50.8 seconds. Moreover this does not include the time needed for quality verification. Since then efforts were made to speed the annotation process up, such as recently

proposed method[10], which achieves 7 seconds per bounding box, while keeping the annotation quality.

Segmentation. Annotating a pixel level ground-truth is even more difficult than bounding boxes, causing much smaller datasets to be available. Although there are big datasets like COCO[11] with 91 categories in 328000 images and ADE20K [12] with 20000 images and 2693 object classes, the number of annotated classes is still small and it is very expensive to obtain ground truth for task specific type of images.

Recently, Perazzi et al.[13] have published DAVIS - a benchmark dataset and evaluation methodology for video object segmentation containing fifty high quality video sequences and per-frame ground truth segmentations. This enabled new boom in tracking by segmentation based on convolutional networks. The performance achieved by the state-of-the-art methods has enabled us to use them for our cointracking problem.

Long *et al.* [14] introduced fully convolutional neural networks (FCN) for semantic segmentation. They proposed to modify the pretrained convolutional classification network by replacing the fully connected layers by convolutions with kernel size 1×1. This allows the FCN to output dense predictions as opposed to the single global class predicted by the original classification neural network.

The network architectures used in current state-of-the-art methods on video object segmentation [3, 15, 4, 16] are based on this idea and similarly use pretrained image classification networks such as VGG16 [17] or ResNet[18] and modify them for the segmentation task. They differ in several key design choices as discussed in the following sections.

3.2.1 Segmentation resolution

The resolution of the deep layers in image classification networks is small (7×7 in VGG16). As a consequence, the network has to be further modified in order to be useful for per-pixel segmentation task. The segmentation network architectures can be grouped by the mechanisms used for achieving full resolution outputs.

Skip connections

The methods used by[4, 16, 14] rely on "skip connections", which combine the outputs of high-resolution shallow layers providing the details with the outputs of the deep layers that have bigger receptive field, but low resolution. In particular, [4] implements the skip connections by upscaling the outputs of the last convolutional layer before each pooling layer to the input image resolution by means of bilinear interpolation. All of these skip connections are later concatenated together and passed through a 1×1 convolution linear classifier to get the output per-pixel class predictions.

Dilated convolutions

Another way of achieving higher output resolution is dilated convolution (also called atrous convolution from the french "à trous" [a tʁu] meaning "with holes") proposed by [19]. They replace the standard convolutions in deeper CNN layers by a new type of convolution, which has larger kernel size, but keeps the number of learned parameters intact. More specifically, the atrous convolution introduces zeros between the consecutive values of the convolution filter, increasing the kernel size from $k \times k$ to $k_e \times k_e$, where $k_e = k + (k-1)(r)$, with r being the number of additional zeros in between the filter coefficients. Varying the r allows to control the field-of-view of the layer.

The higher output resolution is then achieved by combining atrous convolution with not downsampling via pooling in the deeper layers (4 and 5 in VGG). However, the output resolution of the network proposed in [19] is still only 1/8 of the input image resolution. To get to the full resolution, they further upscale the output segmentation by bilinear interpolation and employ a fully connected conditional random field (CRF) [20] to improve the segmentation accuracy. CRF uses a logarithm of the network outputs as unary potentials, while the pairwise potentials are combination of two terms. The first one penalizes the pixels with similar color and position but different labels and the second one penalizes pixels with different labels based only on their spatial distance.

3.2.2 Loss function

In classification, the cross-entropy loss function is commonly used. The outputs of the last network layer are first converted to represent class probabilities by a non-linear activation sigmoid function, usually the logistic function:

$$q(x) = \frac{e^x}{e^x + 1}$$

With known probability labels $p(x)$, the cross-entropy

$$H(p, q) = -\sum_x p(x) \log(q(x)) \tag{3.5}$$

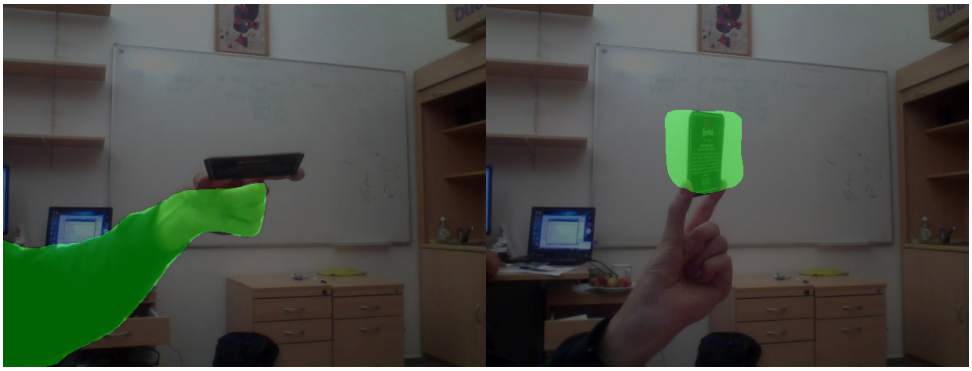
characterizes the difference between the distributions p and q. As the segmentation task is in fact a per-pixel classification, the cross-entropy loss and its modifications are commonly used as well.

Xie and Tu [21] proposed an edge detection method, formulated as per-pixel edge/background classification. They argue that the cross-entropy loss function needs to be modified for that task, because of the high inbalance in the data. In the case of edge detection as much as 90% of the pixels are non-edge, leading to a strong bias towards the background class. Consequently, they have proposed to use a balanced cross-entropy loss - a simple modification of the cross-entropy loss which fights the imbalance of the classes by reweighting the terms corresponding to each of them.

$$\begin{aligned} L_{\text{bal}}(\mathbf{W}, X) = &- \beta \sum_{j \in Y_+} \log \Pr\left(y_j = 1 | X; \mathbf{W}\right) \\ &- (1 - \beta) \sum_{j \in Y_-} \log \Pr\left(y_j = 0 | X; \mathbf{W}\right) \end{aligned} \tag{3.6}$$

where $\beta = |Y_-| \,/\, |Y|$ denotes the fraction of pixels in the background class. This loss function or its variation has then been adopted by other segmentation methods, including OSVOS [4] and several of the top-scoring entries [22, 23, 24] in the DAVIS 2017 challenge[25].

OnAVOS [16] addresses the class inbalance by using the online bootstrapping method proposed in [26]. The cross-entropy is computed per-pixel according to equation (3.5), but only the worst k of the pixels are then considered when computing the mean loss of the segmentation. This leads to class-balancing, because as the network learns to classify the pixels in the dominant class correctly, the pixels from this class are more often skipped in the loss computation, leading to bigger importance of the pixels from the weaker class. The number k of pixels is updated dynamically based on the performance on the current image.



(a) : The segmentation drifted to the hand holding the tracked object because of the use of optical flow

(b) : The segmentation did not react to the object rotation because of incorrectly big influence of the propagated mask

Figure 3.2: Examples of LucidTrack [3] failures on our data.

3.2.3 Input modality

Apart from the RGB image of the current frame, other inputs may be used. Two such input types most relevant to our problem are discussed in this section.

Optical flow

The optical flow captures an important information about the motion in the video. It is probable, that a big discrepancy in optical flow corresponds to some object boundary. It is thus reasonable to use optical flow as an input, however it does not provide any help when the tracked object is not moving or when it is moving in a similar way to another object in the scene. In cointracking, this happens often, because the tracked object is likely to be held by a hand that moves in the same way. In such case, the flow can cause the segmentation to drift as shown in figure 3.2a.

Mask

MaskTrack [15] and LucidTrack [3] both use the computed segmentation mask from the previous frame as an additional input channel. The network's task is then to refine a segmentation mask - given an imperfect mask, compute the

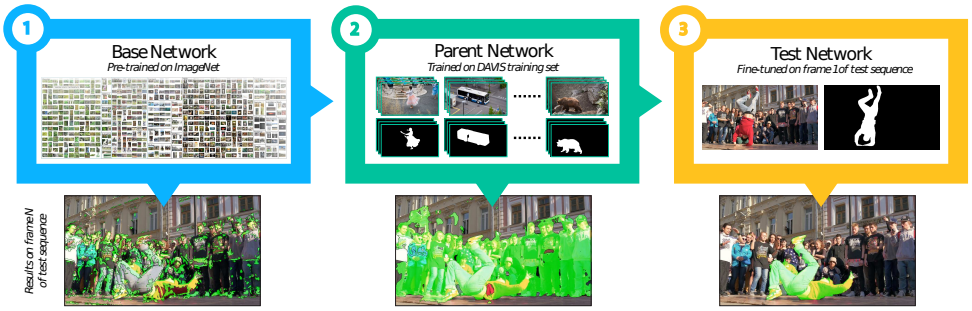


Figure 3.3: Three stage training proposed in [4]

correct one. Under the assumption of slow camera/object motion the network output on the previous frame is a good estimate of the correct segmentation of the current one. The previous frame mask is morphologically dilated in [15] and then added as additional channel to the current frame's RGB. In [3] instead of dilating the mask, they warp it with the optical flow between the two frames.

Using the segmentation mask from the previous frame adds some temporal information, but there are situations where it cannot be used, such as the tracked object being fully occluded (or going out of the camera view) for. Furthermore it is not obvious how the network balances between the information coming from the image and the mask. Figure 3.2b shows an incorrect segmentation computed by LucidTrack caused by the network putting more importance to the previous mask than to the current image.

3.2.4 Fine-tuning augmentation

As we have discussed in the beginning of this chapter, deep learning relies on large datasets. In [4], they have proposed a three stage training strategy visualized in figure 3.3. After pretraining on ImageNet and transfer learning of the object segmentation task on DAVIS dataset, the network is fine-tuned to segment the particular object of interest. The motivation for the individual training phases is the following. The first ImageNet training stage provides a reasonable weight initialization. It is a de facto standard to transfer the semantic representations learned on the image classification to other tasks. In the second phase, the fully connected layers are dropped and the network is trained to a video object segmentation task. Both the first two training steps should teach the network how a general object looks.

While the first two training stages provide a good initialization, it is still not enough to fine-tune using only the frame 1 and the ground truth. To get

additional training data, different data augmentation strategies have been used. OSVOS [4] augments the training data by adding mirror reflection and scaling. Although this helps preventing overfitting, this particular choice of augmentations has little real-world justification.

In contrast to the simple augmentation, LucidTracker[3] employs much more complex procedure in order to expand the available training dataset. First, the object is separated from the background, then the resulting hole in the background is filled with an image inpainting algorithm [27]. Both object and background are then augmented separately by applying small affine transformation and a thin-plate-spline transform on top of that. The object is then composed on top of the background by means of Poisson matting [28] and the corresponding ground truth segmentation is generated. Furthermore, the image is modified by randomly altering it's saturation and value in HSV space in a non-linear way. The procedure is meant to simulate both object and camera motion and various global illumination changes. Although this is a very rough approximation of the situations that may arise during the video, the performance achieved on the DAVIS challenge is impressive.

3.3 Proposed segmentation DNN

Taking into account all of the previously discussed variants of segmentation networks, we propose a coin-tracking segmentation algorithm of a construction similar to to [4] - the backbone of our network is an ImageNet pretrained VGG16 with the fully connected layers cut off. Before `pool2`, `pool3`, `pool4` and after the last convolutional layer (`conv5_3`), skip connections are made. On each of these skip connection branches, a 3×3 convolution with 16 output channels is applied and the results are upscaled to the input image size using bilinear interpolation. These branches are then concatenated, resulting in $H \times W \times (16 \times 4)$ feature map. Three linear classifiers in form of a single 1×1 kernel convolution with 3 output channels are appended. Finally a sigmoid activation is used to get the soft segmentation output, corresponding to background, **obverse** and **reverse** side respectively.

In order to get the final segmentation a post-processing procedure is proposed. First, the object region is extracted by selecting the largest 4-connected component of the binary image formed by computing $(1 - \text{background}) > \theta_{\text{bg}}$, then any holes inside this mask are filled.

In addition to the object segmentation, the outputs of this network represent

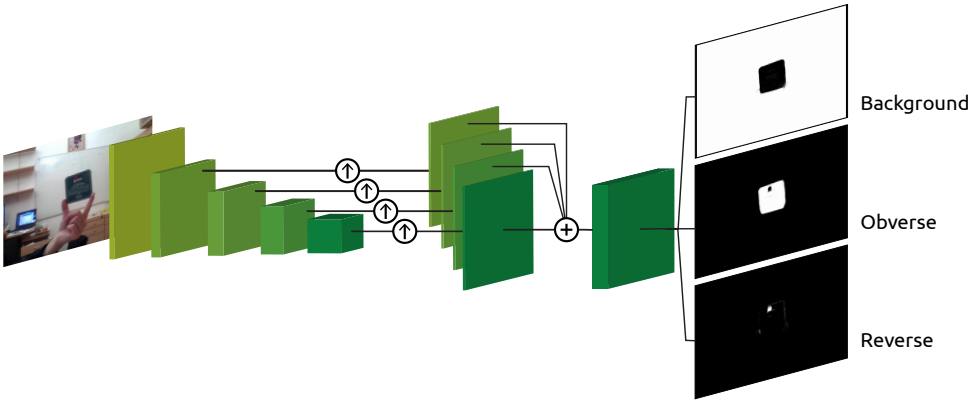


Figure 3.4: Proposed segmentation network architecture

the appearance part of our coin-tracking algorithm. In order to get the predicted **obverse** side probability, the later two outputs of the segmentation network corresponding to the **obverse** and the **reverse** side are summed over the area corresponding to the object, producing two quantities N_{OBV} and N_{REV} respectively. The **obverse** side probability given the observed image I is then estimated as:

$$P(side = \mathrm{OBV} \mid I) = \frac{N_{\mathrm{OBV}}}{N_{\mathrm{OBV}} + N_{\mathrm{REV}}} \tag{3.7}$$

3.3.1 Training

Similarly to [4], backbone VGG network is pretrained on ImageNet classification. After changing the architecture as described in the previous section, the parent network is fine-tuned for segmentation on the DAVIS16 dataset. However, our network has three output channels corresponding to background and the two coin-like object sides as opposed to the single output channel representing the object probability in [4], thus a different loss function has to be applied. As discussed in 3.2.2, many other methods use some kind of class balancing in the loss function. Our data is similar to the DAVIS data, in the sense of the typical object sizes as compared to the image size. The background class usually occupies most of the image and the object class is now divided into two classes corresponding to each of the object sides, leading to further increase of the background class dominance. In contrast to the previous methods, we argue that the class inbalance on the training data represents the inbalance on the test data reasonably well and thus using a class-balancing loss function is counterproductive. The balancing alters the class (object/background) prior probability, i.e. the size of the object compared to the size of the image and consequently should be avoided.

With this in mind, we choose to use a simple cross-entropy loss as defined in equation 3.5. When training the parent network the objects from DAVIS dataset are not coin-like and there is no notion of obverse and reverse side, thus we simply label the objects as both obverse and reverse.

Augmentation

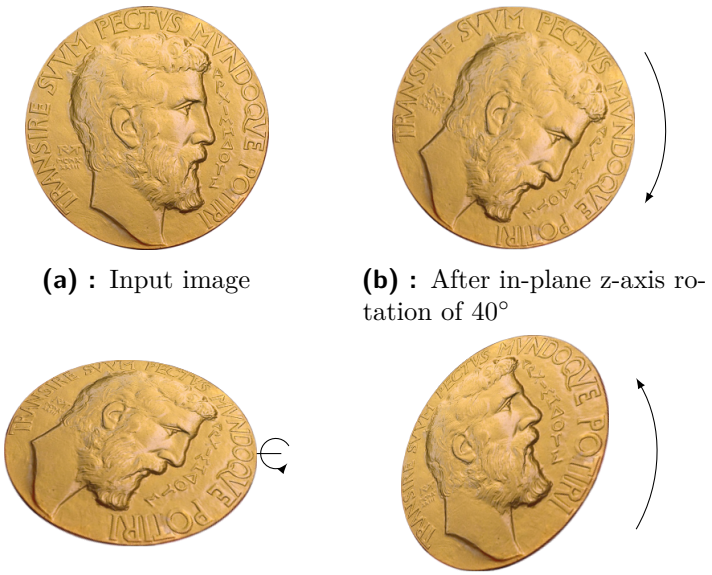
At test time, the pretrained parent network is further fine-tuned on augmented images of the input annotated frames. The properties of the coin-like objects discussed in section 2.1.1 permit to augment the images in a matter similar to [3], but better-founded, because in contrast to their augmentation applied on general 3D objects, our augmentations are direct simulations of possible future object poses.

Following [3], we first augment the Saturation and Value channels of the HSV image representation by computing $I' = aI^b + c$, where a is drawn uniformly from $[1 - 0.05, 1 + 0.05]$, b from $[1 - 0.3, 1 + 0.3]$ and c from $[-0.07, +0.07]$. Next, we split the training image into the object and the background using the provided segmentation mask.

The object image is then randomly resized with scaling factor drawn uniformly from $[0.6, 2]$ and transformed by a homography, which is constructed to represent a realistic 3D rotation of the object, giving us almost a perfect simulation of the possible appearances of the object during the video sequence. The 3D rotation is composed from three random rotations, first one being in-plane rotation around the z-axis, second one out-of-plane rotation around the x-axis and the third one again around the z-axis with the object image centered in origin and lying in the $z = 0$ plane. The angle of each of the rotations around z-axis are uniformly drawn from the full interval $[0°, 360°]$. The out-of-plane rotation (around the x-axis) has angle drawn from $[0°, 85°]$. The process is illustrated in figure 3.5 After the rotation, the brightness of the object image is modified by multiplying the Value channel of its HSV representation by a number drawn randomly from normal distribution with $\mu = 0.2$ and $\sigma = 1$, in order to simulate the brightness changes caused by the object rotation with respect to light source.

In order to compose the augmented object with a meaningful background, we fill the hole in the background image using the open-source OpenCV[1] implementation of the image inpainting method by Telea [29]. The resulting image is then distorted by a thin-plate spline deformation [30] with five

[1] https://opencv.org/



(a) : Input image

(b) : After in-plane z-axis rotation of 40°

(c) : After out-of-plane x-axis rotation of -45°

(d) : After z-axis rotation of -60°

Figure 3.5: The process of generating the 3D rotation augmentation.

control points each shifted uniformly by 25px in each coordinate. See the figure 3.6 for an example of such transformation. Finally, the augmented object is placed randomly on the augmented background and a corresponding segmentation mask is created to form the augmented training example.

When training the segmentation network with both sides known in advance, we observed that the network sometimes learned to differentiate the obverse and the reverse side only from the background similarity to the training examples. Therefore, we changed the augmentation procedure to uniformly sample the background from all the ground truth image-segmentation pairs.

We generate 300 such augmentations for each of the provided image-segmentation pair.

Single side fine-tuning

In the case of only the **obverse** side of the object known in advance, it is not clear, how to perform the fine-tuning. We propose three different methods. First, the fine-tuning is the same as in case of two-sided fine-tuning, setting all the fine-tuning reverse side labels to zero. However, this *zero reverse* strategy should not be used when both of the object sides look similar to

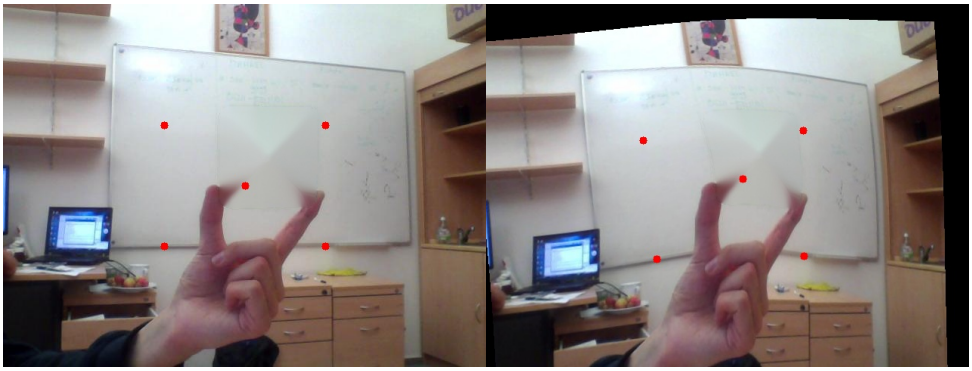


Figure 3.6: An example of a thin-plate spline deformation of an inpainted background. The TPS control points are shown in red.

each other, because in that case we would incorrectly teach the network that the **reverse** side does not look like the **obverse** one. To address this issue, we propose an *ignore reverse* strategy, where the reverse side is labeled zero on the background and special ignore label on the object. The third strategy (*fake reverse*) is again inspired by the Lucid dreaming [3]. Instead of not providing any training samples of the reverse side, we propose to use a random crop from the DAVIS dataset shaped as the mirrored obverse side of the object in order to hallucinate some possible reverse side appearances. While this does not result in real-looking objects, the goal is mostly to provide an object with realistic shape and texture different from background.

(a) : Original image **(b) :** Augmented image

Figure 3.7: Examples of the augmentations. Notice the fake **reverse** side on the last row.

Chapter 4

Shape

As introduced in section 2.1, shape is one of the features useful for **obverse/reverse** side discrimination. In this chapter, we will describe a simple method of side classification from shape, based on Afinne Moment Invariants.

For non-symmetric objects, the visible side can be distinguished just by looking at the shape of the object occluding contour. A simple flip detector can be designed based on Affine Moment Invariants (AMIs), which are functions of image moments invariant with respect to affine transformations. Flusser et al. [31] show, that it is impossible to construct a projective invariant from finite number of moments, leaving AMIs as necessary approximation.

A mirror reflection is element of affine transformation, so true affine invariants would not help us to discriminate the two sides of the tracked object. Fortunately, affine moment pseudoinvariants can be constructed, which are invariant with respect to affine transformations up to the sign, which represents the presence of mirroring in the transformation, yielding a simple way of of flip detection. We use two independent affine moment pseudoinvariants I_5 and I_{10}, listed in [31].

In order to get the pseudoinvariants, first the segmentation mask central moments μ_{ij} have to be computed up to fourth order ($i + j \leq 4$).

$$\mu_{ij} = \sum_{x,y} (x - \bar{x})^i (y - \bar{y})^j \tag{4.1}$$

with $\bar{x}$ and $\bar{y}$ being the mask centroid coordinates defined as:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}} \tag{4.2}$$

$$m_{ij} = \sum_{x,y} x^i y^j \tag{4.3}$$

The x and y are coordinates at which the segmentation mask is non-zero.

The two used independent pseudoinvariants are defined as follows:

$$\begin{aligned} I_5 =&(\mu_{20}^3\mu_{30}\mu_{03}^3 - 3\mu_{20}^3\mu_{21}\mu_{12}\mu_{03}^2 + 2\mu_{20}^3\mu_{12}^3\mu_{03} - 6\mu_{20}^2\mu_{11}\mu_{30}\mu_{12}\mu_{03}^2 \\ &+ 6\mu_{20}^2\mu_{11}\mu_{21}^2\mu_{03}^2 + 6\mu_{20}^2\mu_{11}\mu_{21}\mu_{12}^2\mu_{03} - 6\mu_{20}^2\mu_{11}\mu_{12}^4 \\ &+ 3\mu_{20}^2\mu_{02}\mu_{30}\mu_{12}^2\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}^2\mu_{12}\mu_{03} + 3\mu_{20}^2\mu_{02}\mu_{21}\mu_{12}^3 \\ &+ 12\mu_{20}\mu_{11}^2\mu_{30}\mu_{12}^2\mu_{03} - 24\mu_{20}\mu_{11}^2\mu_{21}^2\mu_{12}\mu_{03} + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{12}^3 \\ &- 12\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{12}^3 + 12\mu_{20}\mu_{11}\mu_{02}\mu_{21}^3\mu_{03} - 3\mu_{20}\mu_{02}^2\mu_{30}\mu_{21}^2\mu_{03} \\ &+ 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{21}\mu_{12}^2 - 3\mu_{20}\mu_{02}^2\mu_{21}^3\mu_{12} - 8\mu_{11}^3\mu_{30}\mu_{12}^3 + 8\mu_{11}^3\mu_{21}^3\mu_{03} \\ &- 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{21}^2\mu_{03} + 24\mu_{11}2\mu_{02}\mu_{30}\mu_{21}\mu_{12}^2 - 12\mu_{11}^2\mu_{02}\mu_{21}^3\mu_{12} \\ &+ 6\mu_{11}\mu_{02}^2\mu_{30}^2\mu_{21}\mu_{03} - 6\mu_{11}\mu_{02}^2\mu_{30}^2\mu_{12}^2 - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21}^2\mu_{12} \\ &+ 6\mu_{11}\mu_{02}^2\mu_{21}^4 - \mu_{02}^3\mu_{30}^3\mu_{03} + 3\mu_{02}^3\mu_{30}^2\mu_{21}\mu_{12} - 2\mu_{02}^3\mu_{30}\mu_{21}^3)/\mu_{00}^{16} \end{aligned} \tag{4.4}$$

$$\begin{aligned} I_{10} =&(\mu_{20}^3\mu_{31}\mu_{04}^2 - 3\mu_{20}^3\mu_{22}\mu_{13}\mu_{04} + 2\mu_{20}^3\mu_{13}^3 - \mu_{20}^2\mu_{11}\mu_{40}\mu_{04}^2 \\ &- 2\mu_{20}^2\mu_{11}\mu_{31}\mu_{13}\mu_{04} + 9\mu_{20}^2\mu_{11}\mu_{22}^2\mu_{04} - 6\mu_{20}^2\mu_{11}\mu_{22}\mu_{13}^2 \\ &+ \mu_{20}^2\mu_{02}\mu_{40}\mu_{13}\mu_{04} - 3\mu_{20}^2\mu_{02}\mu_{31}\mu_{22}\mu_{04} + 2\mu_{20}^2\mu_{02}\mu_{31}\mu_{13}^2 \\ &+ 4\mu_{20}\mu_{11}^2\mu_{40}\mu_{13}\mu_{04} - 12\mu_{20}\mu_{11}^2\mu_{31}\mu_{22}\mu_{04} + 8\mu_{20}\mu_{11}^2\mu_{31}\mu_{13}^2 \\ &- 6\mu_{20}\mu_{11}\mu_{02}\mu_{40}\mu_{13}^2 + 6\mu_{20}\mu_{11}\mu_{02}\mu_{31}^2\mu_{04} - \mu_{20}\mu_{02}^2\mu_{40}\mu_{31}\mu_{04} \\ &+ 3\mu_{20}\mu_{02}^2\mu_{40}\mu_{22}\mu_{13} - 2\mu_{20}\mu_{02}^2\mu_{31}^2\mu_{13} - 4\mu_{11}^3\mu_{40}\mu_{13}^2 + 4\mu_{11}^3\mu_{31}^2\mu_{04} \\ &- 4\mu_{11}^2\mu_{02}\mu_{40}\mu_{31}\mu_{04} + 12\mu_{11}^2\mu_{02}\mu_{40}\mu_{22}\mu_{13} - 8\mu_{11}^2\mu_{02}\mu_{31}^2\mu_{13} \\ &+ \mu_{11}\mu_{02}^2\mu_{40}^2\mu_{04} - +2\mu_{11}\mu_{02}^2\mu_{40}\mu_{31}\mu_{13} - 9\mu_{11}\mu_{02}^2\mu_{40}\mu_{22}^2 \\ &+ 6\mu_{11}\mu_{02}^2\mu_{31}^2\mu_{22} - \mu_{02}^3\mu_{40}^2\mu_{13} + 3\mu_{02}^3\mu_{40}\mu_{31}\mu_{22} - 2\mu_{02}^3\mu_{31}^3)/\mu_{00}^{15} \end{aligned} \tag{4.5}$$

Experimental evaluation of the affine moment invariant method can be found in chapter 7.

Chapter 5

Dynamics

As discussed in section 2.1, tracked object dynamics contain lot of information about the currently visible side. In this section, we propose two ways of measuring the object out-of-the-plane rotation. The changes of the measured quantity can then be used to predict a possible side flip occurence or, equally importantly, to detect parts of the video sequence, during which only one of the object sides is visible as illustrated in figure 5.1.

5.1 Segmentation area

The first method is a very simple baseline. Instead of measuring the angle itself, we propose to measure the area of the object image. This measurement is trivial to obtain from the segmentation algorithm output mask for each frame of the input video sequence. The apparent object area is a sensible quantity to track, because it is closely related to the out-of-plane rotation of the object. More specificaly, let us assume the following.

Assumption 5.1. *The object distance to the camera is far larger than the object size.*

Assumption 5.2. *The object distance to the camera does not change significantly.*

Then the assumption 5.1 ensures, that the perspective effects on the rotating object are negligible and the object image transformation is well approximated by an affine transform. When the object is rotated directly out-of-the-plane from a frontoparallel position by an angle α, the object image area shrinks as $A' = A \cos \alpha$. With the assumption 5.2 ensuring that the object image area

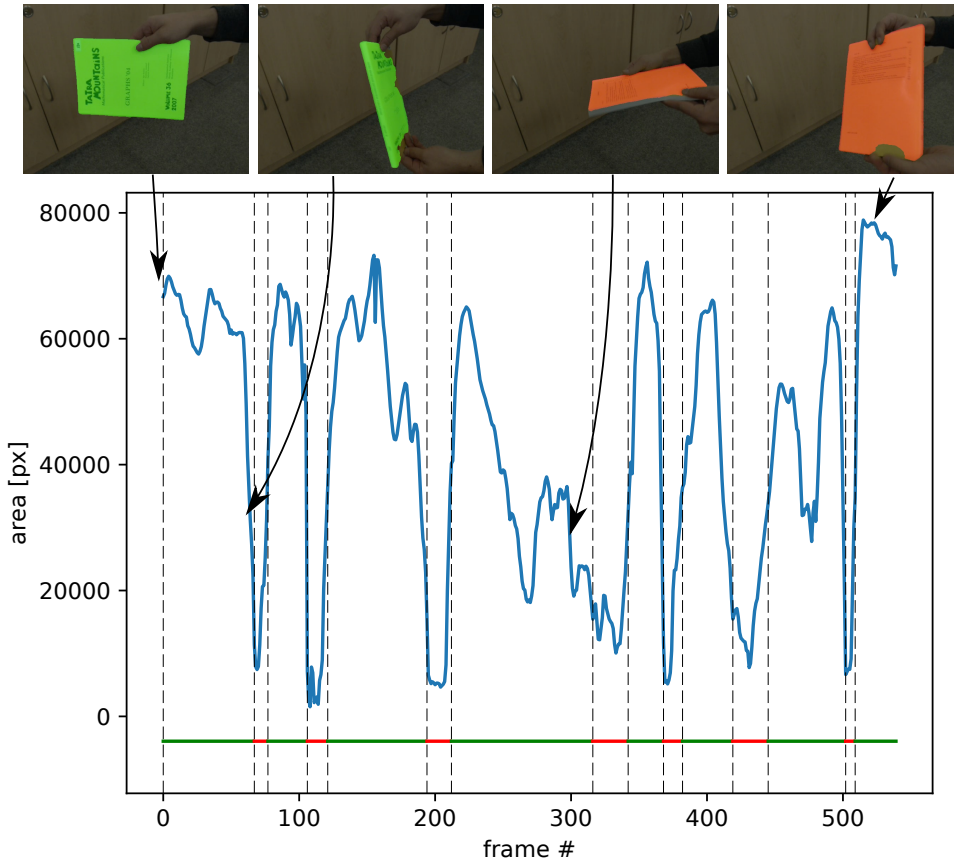


Figure 5.1: Example of dynamics-based flip prediction. Parts of sequence not suspected to contain any flips are shown in green, parts with possible flips in red.

does not change significantly due to the 3D translations, the area can be used instead of the out-of-the-plane rotation angle.

5.2 Out-of-the-plane rotation angle regression

The second method we propose is direct regression of the out-of-the-plane rotation angle using a deep neural network designed specially for the coin-tracking task. The network consists of two identical feature extractors with shared parameters, the first one is fed with a prototype frontoparallel image of the tracked object, while the second one uses the rotated object from the current frame of the video sequence as segmented by the segmentation network. The outputs of the last convolutional layer from both branches are then concatenated and fed through three fully-connected layers. The output of the resulting network represents the out-of-the-plane angle.

Architecture

As the feature extractor, we use MobileNet 1.0 [32], which is an adaptation of the VGG16 [17] network with much smaller number of parameters, that was designed to be used in mobile devices with limited computational resources. The parameter reduction is accomplished by using *depth-separable* convolutional layers instead of the standard ones. Instead of having a convolution filter with $K \times K \times D_{in}$ parameters per each output channel as discussed in section 3.1.1, [32] have proposed to have only one $K \times K \times 1$ *depthwise* convolution filter for each layer input channel. A linear combination in form of $1 \times 1 \times D_{out}$ *point-wise* convolution is then applied on the resulting D_{in} channel feature map. This depth-wise separation of the convolution filters have been shown to achieve performance only slightly worse than the same model using the standard convolutions, while having over 7 times less parameters. Please refer to [32] for details of the MobileNet network architecture. While we have chosen to use MobileNet as our backbone model, our proposed architecture does not rely on the exact feature extractor architecture and thus a more heavy-weight network, like [18, 33] can be plugged in instead.

The last three fully connected layers are inspired by [34], with output dimensions 1024, 1024 and 1 respectively. Note, that because of the fully connected layers, the input image size have to be fixed, in this case to 224×224 pixels. The object image, cropped by the segmentation mask bounding box, is thus first rescaled to this size, while keeping the aspect ratio intact by padding with zeros when necessary. Note that it is essential to do the rescaling this way, otherwise the information about the object 3D orientation would be lost.

Training

Similarly to the segmentation network, our angle regression CNN feature extractor is pre-trained on ImageNet and the whole network is then finetuned on augmentations of the known object sides. The object is cut out of the image and randomly rotated as described in section 3.3.1. This time, though, the object does not have to be composed on top of an augmented background, because the network operates on the cropped object images only.

Thanks to the chosen sequence of augmentation 3D rotations, the out-of-the-plane angle of the augmented image is uniquely defined by the angle of the second rotation (around the x-axis), thus it is simple to generate training pairs in form of a prototype image, rotated image, angle triplets. To make

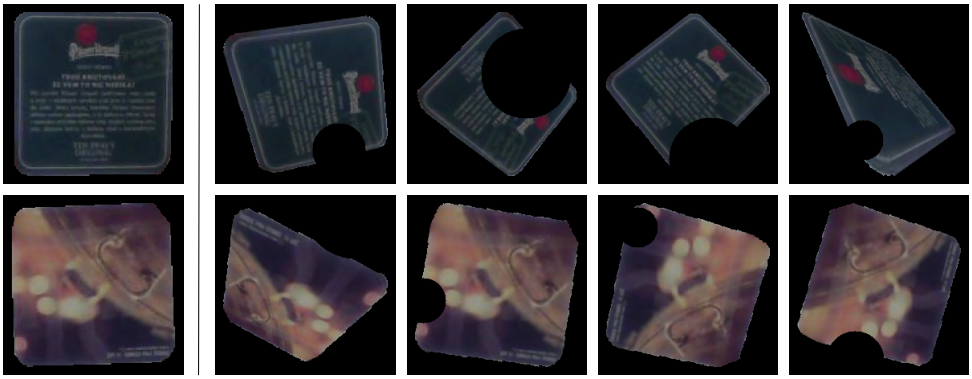


Figure 5.2: Example augmentations of the **obverse** (top) and the **reverse** (bottom) sides from the *beermat* sequence.

the system more robust to the quality of the segmentation, we additionally augment the rotated image by making a random circular "hole" it. After the augmented dataset is generated, the network is trained by Adam optimizer with the loss function chosen as the mean squared difference between the ground-truth and the predicted angle.

Testing

At test time, the object is cropped from the ground-truth side image to form the prototype (for each side if both **obverse** and **reverse** sides are provided). Then, the network is finetuned on the augmentations of the prototypes and evaluated on each frame, once for each available side prototype.

5.3 Sequence partitioning

As discussed in the introduction of this chapter, the estimated object out-of-the-plane angle can be used to identify the flip moments in the sequence. In reality, however, the flip may not be represented by a single frame and the object can stay rotated almost perpendicular to the image plane for prolonged periods of time. As the reliability of all the proposed methods depends heavily on the quality of the segmentation, which drops significantly with big out-of-the-plane rotation angles, we propose to partition the video sequence into "calm" parts without any side flips, where the side classification is more likely to be correct and "flippy" parts, where an unknown number of flips is suspected to happen.

In order to split the sequence into the two mentioned types of subsequences, we threshold the out-of-the-plane angle (or the object image area) signal. When the CNN estimated angle is used, the parts with the out-of-the-plane angle smaller than 45° are considered *calm*. In the case of area based partitioning, the procedure is different, because the object distance from the camera is not fixed exactly and consequently the object area does not have a fixed range, disabling use of a single threshold. Instead, we do the thresholding adaptively, taking into account the maximal observed area in the previous *calm* part.

In particular, the sequence is traversed from the first frame, keeping track of the maximum object area A_{max}. As the first frame is known to be the annotated frame of the **obverse** side, thus most likely to be close to a frontoparallel view, the first part is considered to be a calm one. It's end is reached, when the currently measured area falls bellow $\theta_A \cdot A_{max}$, at which moment a new *flippy* part is started. The thresholding algorithm then continues traversing the sequence, until the observed area rises above $h \cdot \theta_A \cdot A_{max}$, where $h \geq 1$ is a hysteresis coefficient. Our choice of two thresholds θ_A and $h \cdot \theta_A$ is motivated by possible segmentation area instabilities around the flip moments. We empirically set $\theta_A = 0.25$ and $h = 2$.

After the end of the *flippy* part is found, the whole process is repeated until the end of the video sequence, while resetting A_{max} before the start of each *calm* part.

Chapter 6

The coin-tracking algorithm

In this chapter, we propose a coin-tracking algorithm built from the blocks described in the previous chapters. All the three types of information available (appearance, dynamics and shape) will be combined together in one algorithm. As discussed before, we have chosen the tracking by segmentation approach to the coin-tracking problem and introduced a segmentation DNN, which classifies each pixel into belonging to background, **obverse** or **reverse** side of the object. The background output channel of the network is thresholded to get the segmentation mask of the object, which is then post-processed as described in section 3.3. Then, the affine moment invariants are computed, together with the estimated out-of-the-plane angle or segmentation mask areas.

In order to marge all the information together, we propose to formulate the task of visible side classification in terms of classification of the *calm* parts of the sequence followed by propagation of the side labels into the *flippy* parts. We model this part-wise classification in a probabilistic way.

For a given *calm* part $k \in \{1, \ldots, K\}$ identified by its starting frame number s_k and its ending frame number e_k, the outputs of the three measurements – segmentation DNN, the first and the second affine moment invariant are realizations of random variables X_k, Y_k and Z_k respectively. All three random variables share a common mean μ_k, which is the probability, that the **obverse** side of the object is visible in the part k. The random variable X_k is continuous in $[0, 1]$ interval, while the variables Y_k and Z_k have Bernoulli distribution.

The realizations $x_k^{(t)}$ are directly the outputs of the per-frame **obverse** side probabilities of the segmentation DNN.

The realizations $y_k^{(t)}$ are computed from the first invariant $I_1^{(t)}$ by comparing its sign to the sign of the invariant of the **obverse** side prototype I_{OBV}.

$$y_k^{(t)} = \begin{cases} 1, & \text{if } I_1^{(t)} \cdot I_{\mathrm{OBV}} > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

The second invariants are processed in the same way to get $z_k^{(t)}$ and the mean μ of the random variables is then estimated by computing

$$\begin{aligned} \mu_k =& \frac{1}{3}\left(\left(\frac{1}{e_k - s_k}\sum_{t=s_k}^{e_k} x_k^{(t)}\right) + \left(\frac{1}{e_k - s_k}\sum_{t=s_k}^{e_k} y_k^{(t)}\right) + \left(\frac{1}{e_k - s_k}\sum_{t=s_k}^{e_k} z_k^{(t)}\right)\right) \\ =& \frac{1}{3}\frac{1}{e_k - s_k}\left(\sum_{i=s_k}^{e_k} x_k^{(t)} + \sum_{i=s_k}^{e_k} y_k^{(t)} + \sum_{i=s_k}^{e_k} z_k^{(t)}\right) \end{aligned} \tag{6.2}$$

The part k classification is then

$$C_k = \begin{cases} \text{obverse}, & \text{if } \mu_k > 0.5 \\ \text{reverse}, & \text{otherwise.} \end{cases} \tag{6.3}$$

After all of the *calm* are classified as either the **obverse**, or the **reverse** side, the labels are propagated to get final per-frame side classification. In the case of *calm* parts, the frames simply inherit the side label from the part classification. The *flippy* parts are split into two sub-parts by finding the frame $f_{\min}$ with the largest out-of-the-plane angle or the smallest segmentation area, respectively. The frames before $f_{\min}$) get the side label of the previous calm part, while the frames after it are assigned the label of the next calm part.

Chapter 7

Evaluation

In this chapter, we show a qualitative and quantitative comparison of the different design choices we have made.

7.1 Data

We have collected a dataset of 22 video sequences, containing various coin-like objects, including several videos of hands and books, which do not exactly fit in to the coin-like object definition, but are close to it and were included to test the robustness of the proposed algorithms. The sequences vary in image quality, the size, motion speed, and appearance of the tracked objects as well as camera movement and complexity of the background. Example frames from the dataset are shown in figures 7.1 and 7.2.

Due to the difficulty of video object segmentation annotation, we have annotated the sequences only with sparse bounding boxes. A tight axes-aligned bounding rectangle ground truth was manually annotated for every 30th frame. The visible side annotation is provided in form of a list of frames containing a flip, although it was often not possible to identify a single frame when the flip occurred, in which case the closest one was chosen. The visible side can then be computed simply by computing number of ground truth flips up to the current frame.

The sequences, together with the bounding box and flip annotations and the ground truth segmentation for each side prototype, are published on the CD accompanying this thesis.

Figure 7.1: Example frames from the coin-tracking dataset.

Figure 7.2: Example frames from the coin-tracking dataset - continued.

Sequence	# of frames
beermat	1000
card1	630
card2	432
coin1	55
coin2	94
coin3	246
coin4	242
husa	741
iccv_bg_handheld	985
iccv_handheld	379
iccv_simple_static	583
iccv_static	561
pingpong1	436
pingpong2	39
plain	1000
ruka	318
ruka_handheld	600
ruka_static	662
statnice	600
tatra	540
tea_diff_2	300
tea_same	505

Table 7.1: Coin-tracking dataset

7.2 Performance evaluation

We propose two evaluation criterions, one for assesing the quality of the segmentation and one for the side classification accuracy.

A classical performance metric for image segmentation is the Jaccard index, also known as Intersection over Union (IoU), computed from the ground truth segmentation S_{GT} and the algorithm output segmentation S_{out} as

$$J(S_{\mathrm{out}}, S_{\mathrm{GT}}) = \frac{S_{\mathrm{out}} \cap S_{\mathrm{GT}}}{S_{\mathrm{out}} \cup S_{\mathrm{GT}}} \tag{7.1}$$

Because of unavailability of per-frame ground truth segmentation for our data, we choose to employ a relaxation of this measure and we only compute the Intersection over Union of the axis-aligned bounding boxes of the respective segmentation masks. To get the score of a sequence, we average the IoU over all annotated frames, resulting in the Average Overlap (AO) metrics.

In order to measure the side classification performance, we use the standard classification accuracy, computed as the percentage of correctly classified frames in a video sequence. This metrics will be called Side Accuracy (SA).

In the following sections, we will first show an experimental evaluation of the three main components of our coin-tracking algorithm, the segmentation network, side classification from shape and the dynamics based sequence partitioning. Finally the method will be evaluated as a whole with the help of the Average Overlap and the Side Accuracy metrics.

7.2.1 Segmentation network

The segmentation network is an essential component of our proposed method, with all of the other parts relying heavily on the segmentation quality.

Lucid dreaming data augmentation

We have qualitatively tested the impact of performing the complicated augmentations described in 3.3.1 as opposed to the simple augmentations used in [4]. Figure 7.3 shows an illustrative comparison of the effect of the tuning strategy choice.

The positive effect of the "Lucid" augmentation is most notable on sequences with complex backgrounds and especially on sequences, where background dissimilar to the background on the ground truth frame appears later during the video because of the camera motion (e.g. pingpong1 sequence). On such sequences, the simple fine-tuning tends to overfit. As we have discussed previously, our augmentation strategy captures the properties of the tracked object and the synthesized images are much closer to the expected frames in the sequence, than the ones which are simply mirrorings and scalings of the ground truth frame.

Network pretraining

The ImageNet pretrained segmentation network was further finetuned on the DAVIS ([13]) dataset as described in [4]. In contrast to our cross-entropy

Figure 7.3: Finetuning strategy results comparison. Left: basic, Right: ours. The segmentation masks before post-processing are visualized by a green overlay.

loss, the loss function from [4] was not normalized by the input image size in the author's reference implementation[1], thus a better suiting base learning rate had to be found for the training to converge. As shown in figure 7.4, the learning rate of 10^{-4} converged the best and was also used for the final fine-tuning.

As a consequence of the tuning on the DAVIS dataset, in which the tracked object is usually a human, an animal or a vehicle, the network outputs false positive segmentations, if some of these types of objects are visible in the video sequence, but not on the side prototype frames. This is the case of the *card1* sequence, in which the card being tracked lies on a table without being held in hand on both the **obverse** and the **reverse** side annotated frame. Since the network was taught to segment hands during the DAVIS pre-training and no negative training examples were provided during the fine-tuning phase, the output segmentation includes the tracked card, as well as the hand holding it as shown in figure 7.5.

Obverse-only finetuning

We have proposed three fine-tuning strategies in the case when the reverse side ground truth example is not provided. The performance of all three strategies are compared to the fine-tuning with both sides known in advance in figure 7.6. Surprisingly, it turned out, that the *ignore reverse* and the *zero reverse* tuning strategies perform equivalently. Recall that the *zero* strategy sets all the training labels for the **reverse** side to zeros, while the *ignore* one only does this for the pixels belonging to background, while setting the label of the pixels belonging to the **obverse** side to a special *ignore* label, for which the value of the loss function is always zero. We argue that the equivalence of these two approaches stems from the fact, that the SGD-based network

[1] https://github.com/kmaninis/OSVOS-caffe

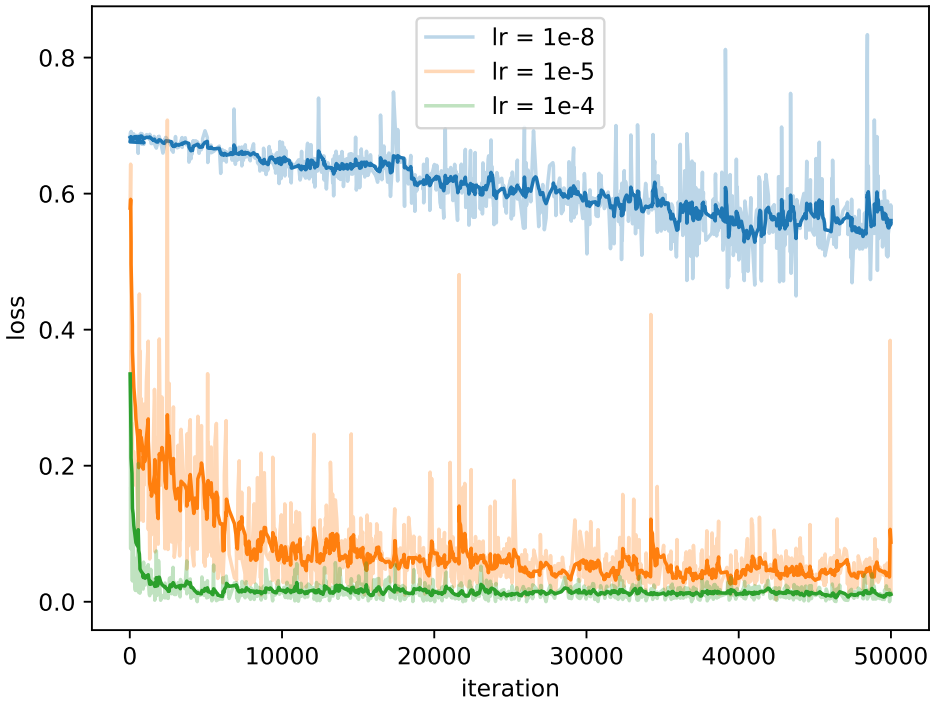


Figure 7.4: Parent network pretraining on the DAVIS dataset ([13]) with varying learning rates. For better visualization of the trends, the loss value smoothed by exponential forgetting is shown in darker color. Notice that the network didn't learn well with the learning rate used in [4]

training tries to find the simplest possible function, that would output the wanted labels on the training data. In case of the **reverse** side classifier, either all of the training labels are zero, or most of them are zero with the rest being ignored. From this point of view it becomes reasonable that both of the training strategies are equivalent as the **reverse** side classifier is essentially just trained to output zeros everywhere.

We hoped that the **reverse** side could be identified by looking for pixels where neither the background, nor the **obverse** side classifier produced significant response, however, we did not observe such behavior and it remains an open question, how to finetune the network without training examples for one of the sides.

The last strategy - *fake reverse*, based on generating fake **reverse** side training data, performs significantly worse than the first two. We think there are two reasons causing that, first one being that the random crops of the DAVIS datasets are very far from the real appearance of the objects in our dataset, commonly being parts of sky or grass or containing people faces. The second reason this strategy does not work as well as expected is the fact,

Figure 7.5: Segmentation failure mode originating in the three-phase training procedure. Left: the **obverse** side training frame, Right: incorrect segmentation of parts of the hand.

that the DAVIS crops frequently directly contradicts the DAVIS-pretraining, i.e. the image patch previously trained to be classified as background is now labeled as object **reverse** side.

7.2.2 Shape-based side classification

Given that the affine moment invariants are a *global* descriptor, it is predicable, that their robustness to the segmentation mask quality will not be great. However, our experiments indicate, that the AMIs' robustness is much smaller, than what we would expect. We have noticed that the side probabilities computed from the invariants are very often around 0.5, meaning maximum uncertainty, which would not be surprising for the symmetric objects, but it also holds for non-symmetric ones, where we have expected much more discriminative power. In order to rule out the influence of the segmentation network output quality, we have tried to compute the invariants on the ground truth annotation segmentations. Surprisingly, it was often not possible to distinguish the sides with the invariants, even though the view of the sides is very close to frontoparallel and the segmentation quality is high. One of the examples of such failure is shown in figure 7.7. In order to eliminate the possibility of error in the invariants computation, we have also tried artificially flipping the segmentation masks along each axis, which confirmed, that the invariants are computed correctly, because the absolute values did not change and the signs flipped exactly as expected.

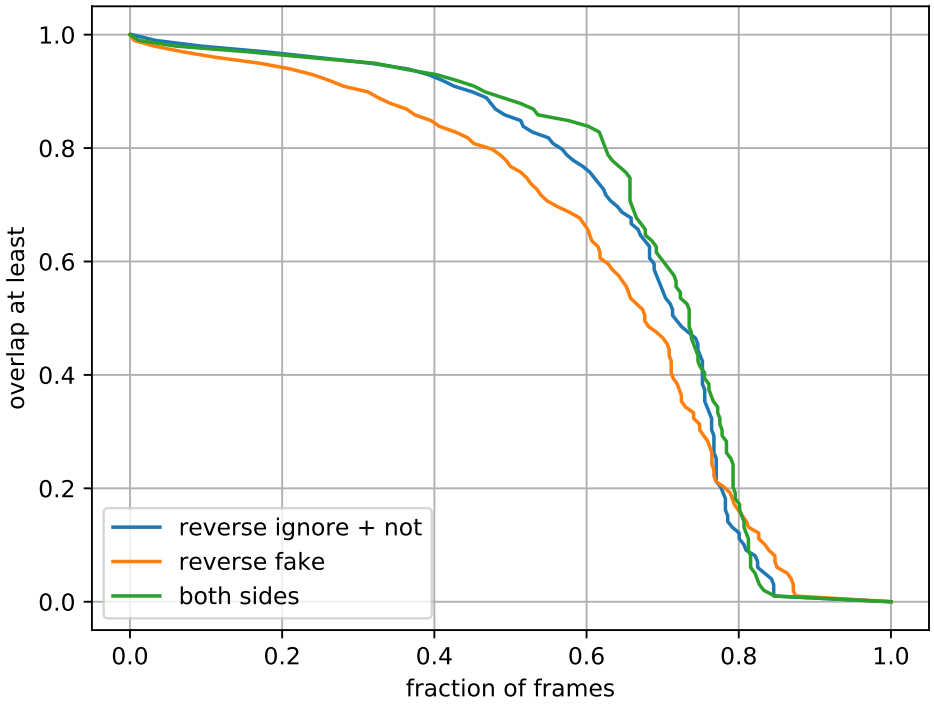


Figure 7.6: Comparison of **obverse**-only fine-tuning with fine-tuning on both sides. The plot shows the fraction of all ground truth annotated frames on which the overlap was greater or equal to the value on the vertical axis. (Better values to the right and to the top.)

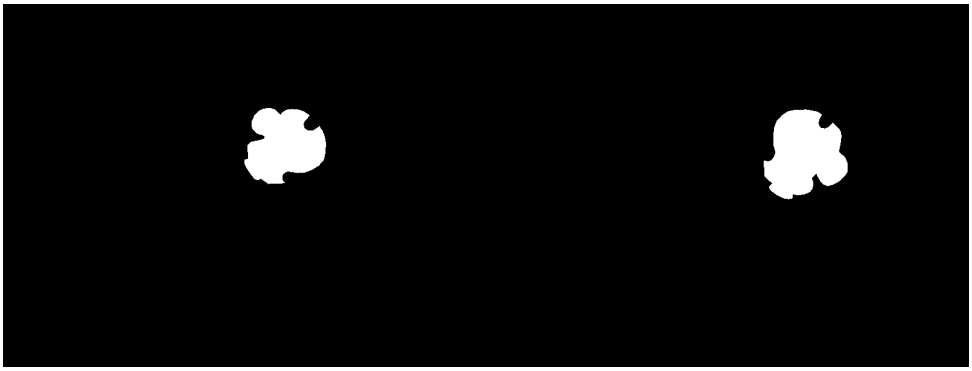
7.2.3 Dynamics

We have proposed two different ways to measure the tracked object out-of-the-plane angle used for the sequence partitioning into *calm* and *flippy* parts. In order to visually compare the two approaches, we have converted the area measurements into the same range as the angle output of the out-of-the-plane regression CNN.

$$A'_i = 90 \left(1 - A_i / \max_i A_i\right) \tag{7.2}$$

As we do not have any ground truth out-of-the-plane annotation, we have compared the two methods only qualitatively. Although the outputs of both methods are clearly strongly correlated, the baseline area-based out-of-the-plane angle estimator seems to provide more stable results, especially around the flips, so we use it for the sequence partitioning.

In order to further asses the performance of the out-of-the-plane rotation regression CNN, we have performed a synthetic experiment, where an object was rotated the full 0° to 90° out-of-the-plane angle range, while keeping



(a) : Obverse: - +

(b) : Reverse: - +

Figure 7.7: Affine invariant failure example. The signs of the first and the second invariant used are shown under each image. Notice that they are the same on both sides, even though the object is not symmetric and clearly mirrored.

the other two angles (3D rotation performed as discussed in section 3.3.1) at a randomly generated values. Figure 7.9 shows the relation between the ground truth angle and the angle regressed by the network. The network outputs are slightly biased near the 0° and 90° extremes, and the precision is not perfect either, but the task of estimating the out-of-the-plane angle was learned successfully.

Sequence partitioning

As the final visible side classification is performed on the partitioned sequence, we have to evaluate the quality of this partitioning as well. It is particularly important, that all of the object flips are covered by a *flippy* part, otherwise the final side classification cannot be possibly correct. One example of such fatal partitioning failure is shown in figure 7.10. This particular example is from a sequence *ruka*, where the tracked object is a hand, which does not fit the coin-like object definition well enough.

On the other hand, classifying a part of the video sequence as *flippy* even if no real flip occurs in it is not as big of an issue. An example of successful partitioning is shown in figure 7.11.

All of the sequences in our coin-tracking dataset were partitioned correctly (in the sense of all flips being covered by a *flippy* part), except for the *ruka*, *ruka_static*, *ruka_handheld* sequences, in which a hand is tracked an the *iccv_bg_handheld* sequence, which fails because of poor segmentation quality.

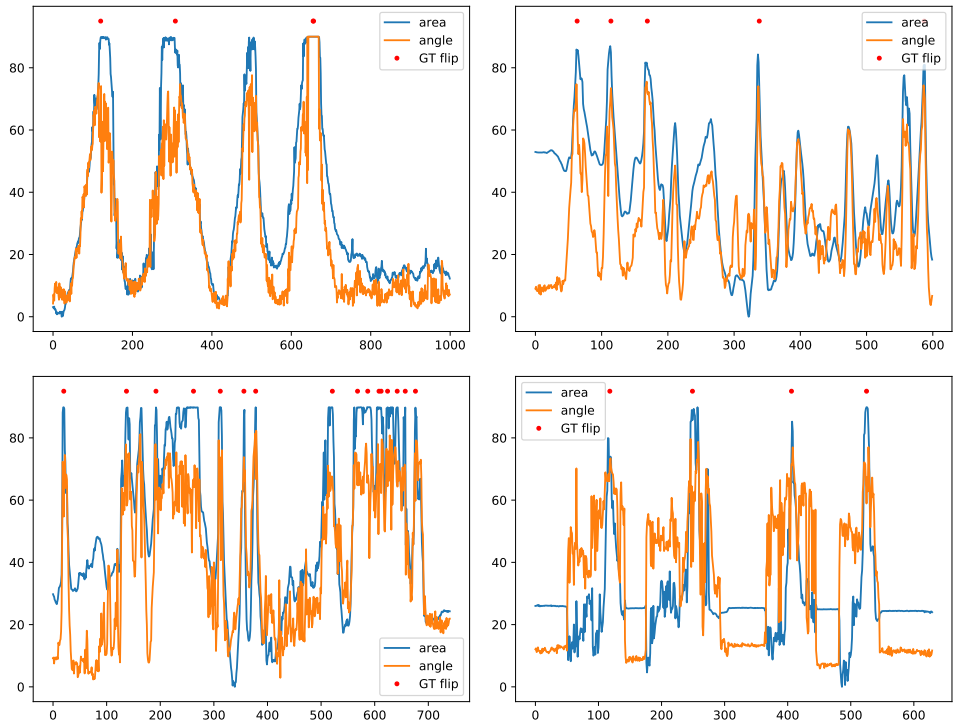


Figure 7.8: Comparison of the estimated out-of-the-plane angle and the segmentation area, modified by formula 7.2. Notice the strong correlation between the two methods output.

7.2.4 Overall performance

In the previous sections, we have examined some of the design choices qualitatively. In order to get a quantitative insight into the performance of the whole proposed method, the per-sequence results will be discussed in this section.

Average overlap

We have tested the segmentation thresholds θ_{bg} in range from 0.1 to 0.9. The results shown in figure 7.12 indicate, that the segmentation algorithm is very robust with respect to choice of the threshold. By comparing the area under curve (AUC) between the different settings, we have selected the best performing one: $\theta_{\text{bg}} = 0.7$. The per-sequence average overlap is shown together with the side accuracy in table 7.2.

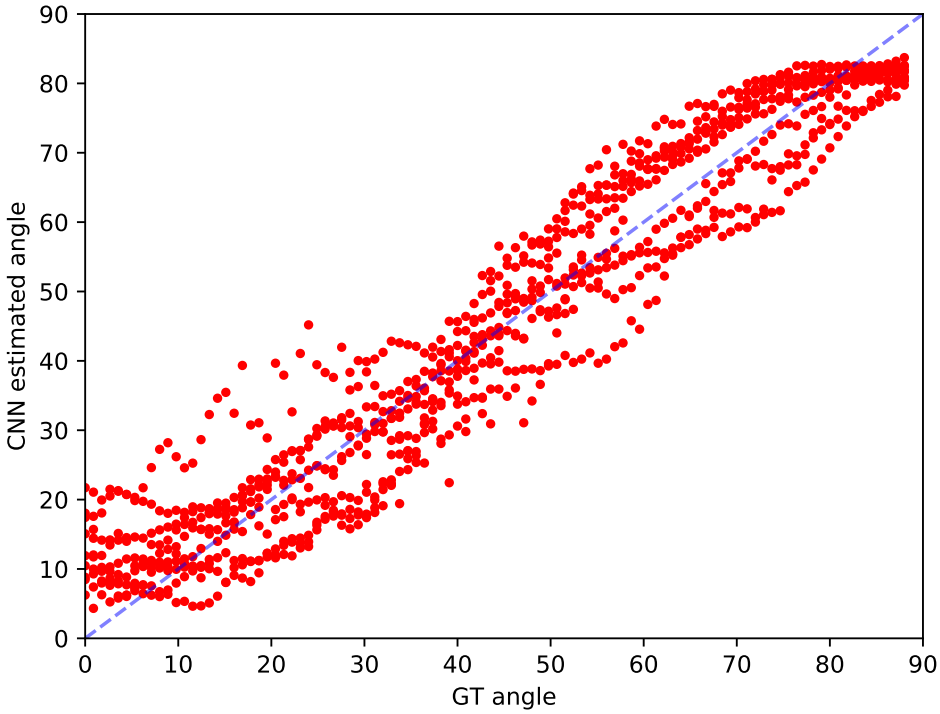


Figure 7.9: Out-of-the-plane CNN synthetic experiment

Side accuracy

Because of the low observed robustness of the affine moment invariants, discussed in section **??**, we have evaluated two variants of the combination algorithm, first one taking the shape into account as defined in chapter 6 and a second one working with the side probabilities from the segmentation network only. The side accuracy averaged over the whole dataset was 66.94% in case of the variant with invariants and 67.42% in the segmentation-only variant. Although the *shape* component of our algorithm did not provide satisfactory results, it did not affect the results much, thanks to the algorithm combining the *appearance*, *shape* and *dynamics* together. See table 7.2 for the complete table of per-sequence results and the qualitative results on the CD accompanying this thesis.

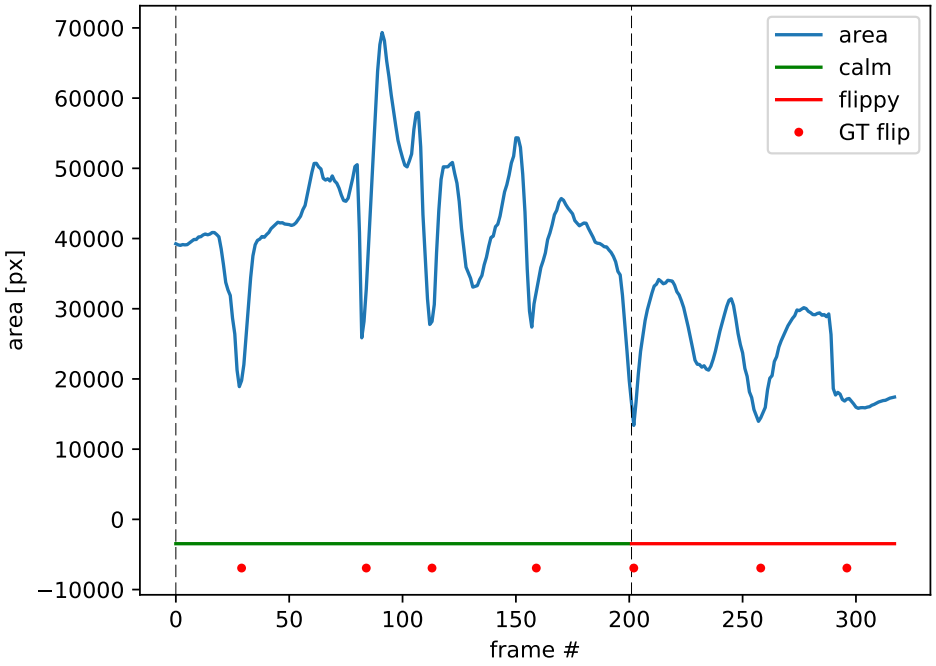


Figure 7.10: Example of a sequence partitioning failure caused by the low coin-likeness of the object.

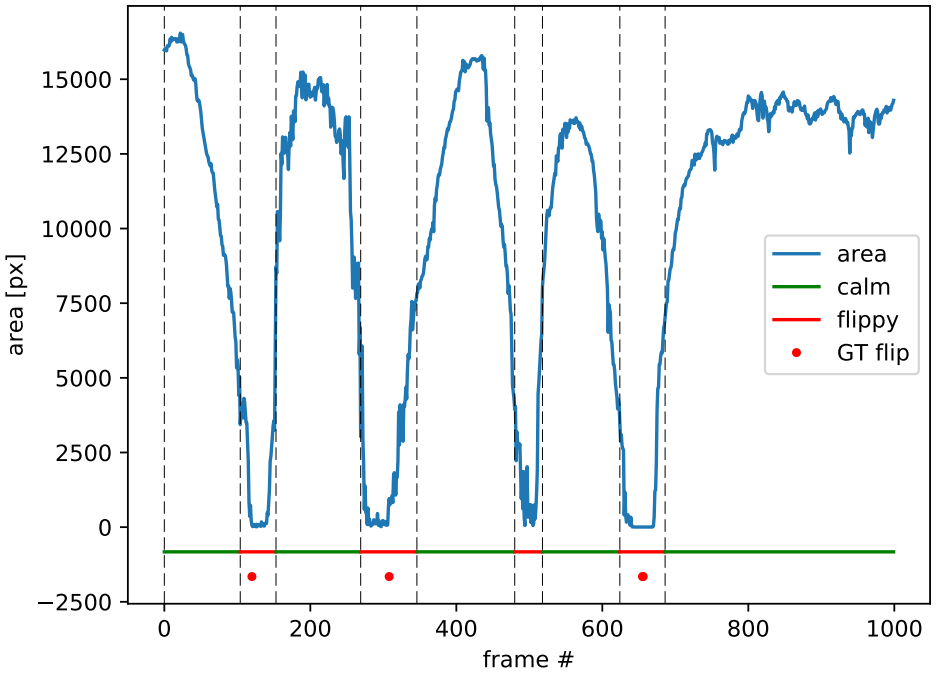


Figure 7.11: Example of a successful sequence partitioning on the *beermat* sequence. Notice the flippy part without a ground truth flip present (around frame 500), which originates from the object almost flipping, but stopping just before the side flip would occur.

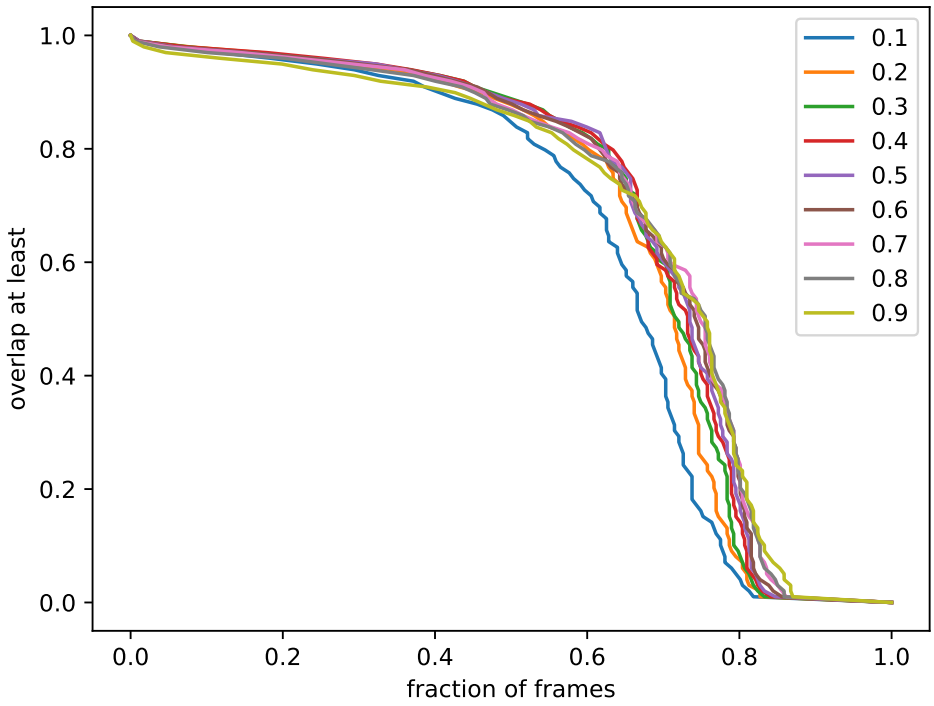


Figure 7.12: Average overlap sensitivity to segmentation threshold

Sequence	Side accuracy	Average overlap
card2	0.99	0.87
beermat	0.98	0.81
statnice	0.97	0.90
coin2	0.87	0.40
coin3	0.80	0.37
husa	0.80	0.80
plain	0.69	0.77
coin1	0.67	0.93
iccv_static	0.66	0.59
card1	0.60	0.71
tatra	0.59	0.92
ruka_handheld	0.56	0.85
iccv_bg_handheld	0.52	0.41
iccv_handheld	0.52	0.49
pingpong1	0.52	0.31
iccv_simple_static	0.59	0.56
ruka_static	0.49	0.66
ruka	0.43	0.96
pingpong2	0.41	0.20
coin4	0.39	0.30

Table 7.2: Per-sequence evaluation results. The side accuracy on the sequences below the red line is worse than random.

Chapter 8

Conclusions and future work

In the first part of this thesis, we have introduced a novel visual tracking problem, called "coin-tracking". The properties of the problem were discussed and three possible approaches to solving it - *appearance*, *shape* and *dynamics* - were proposed. In order to utilize all of them, we have designed a tracking by segmentation type of algorithm.

After reviewing the state-of-the-art literature on video object segmentation, we have adapted OSVOS [4] - a state-of-the-art video object segmentation DNN published recently at CVPR17 - to our task. In order to train the network, we have designed a data augmentation strategy suitable for coin-tracking inspired by [3]. Apart from the segmentation mask of the tracked object, our network also outputs a visible side probability estimation. Atop of the appearance-based side classification a method based on affine moment invariants was proposed.

In the following chapters, we have described two ways of detecting possible side flips in the sequences, based on dynamics of the object out-of-the-plane rotation angle. Finally the visible side classification was then formulated in terms of a probability framework, combining all of the proposed methods into one algorithm.

We have acquired and annotated a coin-tracking dataset, which was then used for performance evaluation of our proposed algorithm. Although we have discovered that the affine moment invariants are not robust enough in our settings, the method as a whole worked reasonably well on some of the sequences, setting a baseline for the coin-tracking problem. Even with the help of state-of-the-art deep-learning methods, the coin-tracking problem proved to be challenging and the dataset contains several sequences, on which

our method fails because of the reasons discussed in the previous chapter.

In future work, a better shape-based side classifier should be designed as an alternative for the affine invariant moments. As the occluding boundaries are known to be homography related to the occluding boundary of the **obverse** side prototype a robust point-wise matching could provide better results. Moreover, because our method relies heavily on the segmentation quality, causing it to fail on some of the more challenging sequences, different recently published segmentation DNN architectures should be tested.

Appendix A

Content of the CD

The directories on the CD are structured in the following way:

```
thesis.pdf
dataset/
    images/
    segmentations/
    sides/
    flips/
    bboxes/
results/
```

The `images` directory contains one subdirectory for each of the collected sequences, containing numbered video frames as jpg files. The `segmentations` directory is structured in the same way, with the segmentation masks in png files. The `sides` directory contains two files - `front.txt` and `back.txt` - each containing the number of the GT annotated frame for each object's side respectively.

The `flips` and `bboxes` contain the ground truth for performance evaluation. The flips are represented by a JSON file for each sequence, containing a list of frame numbers, where a flip has occurred. The bounding boxes are represented by a tuple $x_{tl}, y_{tl}, x_{br}, y_{br}$, containing the coordinates of the top-left and the bottom-right corner of the bounding box. In the case that none of the object's sides is visible at the frame a `null` value is stored instead of the coordinates. These bounding box representations are then stored in a map structure, with frame number as a key and the bounding representation as value.

Finally, the `results` directory contains videos visualizing the outputs of our coin-tracking method.

Appendix B

Bibliography

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

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[3] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for object tracking. *CoRR*, abs/1703.09554, 2017.

[4] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. *CoRR*, abs/1611.05198, 2016.

[5] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernández, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1–23, 2015.

[6] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

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

[9] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. 2012.

[10] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. Extreme clicking for efficient object annotation. *ArXiv e-prints*, August 2017.

[11] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common Objects in Context. *ArXiv e-prints*, May 2014.

[12] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

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

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[15] Anna Khoreva, Federico Perazzi, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. *arXiv preprint arXiv:1612.02646*, 2016.

[16] P. Voigtlaender and B. Leibe. Online Adaptation of Convolutional Neural Networks for Video Object Segmentation. *ArXiv e-prints*, June 2017.

[17] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[19] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[20] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[21] Saining Xie and Zhuowen Tu. Holistically-Nested Edge Detection. 2015.

[22] Amirreza Shaban, Alrik Firl, Ahmad Humayun, Jialin Yuan, Xinyao Wang, Peng Lei, Nikhil Dhanda, Byron Boots, James M Rehg, and Fuxin Li. Multiple-instance video segmentation with sequence-specific object proposals.

[23] Jingchun Cheng, Sifei Liu, Yi-Hsuan Tsai, Wei-Chih Hung, Shalini De Mello, Jinwei Gu, Jan Kautz, Shengjin Wang, and Ming-Hsuan Yang. Learning to Segment Instances in Videos with Spatial Propagation Network. 2017.

[24] Gilad Sharir, Eddie Smolyansky, and Itamar Friedman. Video Object Segmentation using Tracked Object Proposals. 2017.

[25] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS Challenge on Video Object Segmentation. 2017.

[26] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Bridging category-level and instance-level semantic image segmentation. *CoRR*, abs/1605.06885, 2016.

[27] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[28] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 315–321. ACM, 2004.

[29] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004.

[30] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989.

[31] Jan Flusser, Tomas Suk, and Barbara Zitova. *Moments and Moment Invariants in Pattern Recognition*. Willey, 2009.

[32] 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 e-prints*, April 2017.

[33] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *ArXiv e-prints*, November 2013.