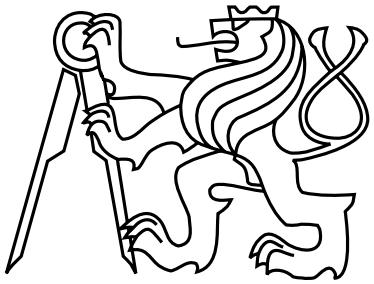
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Bachelor's Project

Detection of Fast Moving Objects

Novotný Lukáš

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

Study Programme: Open Informatics, Bachelor

Field of Study: Computer and Information Science

May, 2016

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Lukáš N o v o t n ý

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Detection of Fast Moving Objects

Guidelines:

1. Collect a dataset of videos containing objects covering in a single frame distance larger than their size. Such objects are captured with signifiant motion blur, appearing as blured strokes.
2. Propose a method to detect such objects in the scene.
3. Estimate color and trajectory of motion blurred objects.
4. Apply the method to the problem of temporal super resolution of the video.
5. Evaluate your method and analyze its failure modes.

Bibliography/Sources:

[1] Video Tracking: Theory and Practice Emilio Maggio, Andrea Cavallaro
ISBN: 978-0-470-74964-7292 pages, January 2011
[2] Visual object tracking. Benchmark www page http://www.votchallenge.net/
[3] Joint tracking and segmentation of multiple targets, Milan, Anton and Leal-Taixe, Laura and Schindler, Konrad and Reid, Ian, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015

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

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, April 25, 2016

Aknowledgements

Firstly I would like to thank my supervisor, prof. Ing. Jiří Matas, Ph.D., for professional guidance and encouragement he has provided throughout the making of this bachelors thesis. Secondly I would like to express my gratitude to Ing. Tomáš Vojíř for his helpful suggestions. Last, but not least, I would like to thank my family, my parents and my sister, for their support during studies, without them I would not have come this far.

Author statement for undergraduate thesis

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, May 26, 2016 .....

Abstract

In this bachelor thesis we introduce a new problem – tracking by detection of very fast moving objects in digital video sequences. These objects appear as blurred strokes, which makes video tracking difficult. We show that there is useful information about object dynamics encoded in this motion blur. We propose a method called BSD to detect such objects by utilizing background subtraction, k-nearest neighbours matching and motion prediction. Our method is evaluated on collected dataset containing nine video sequences covering various different phenomena as well as various blurred strokes. Failure modes are analysed together with suggestions for improvement. One example of real world application is a creation of artificial slow motion video.

Keywords: object detection, visual tracking, motion blur

Abstrakt

V této bakalářské práci představíme nový problém – sledování pomocí detekce velmi rychle se pohybujících předmětů v digitální videosekvenci. Tyto objekty se jeví jako rozmazané čáry, což velmi ztěžuje úlohu sledování. Ukážeme, že právě toto rozmazání uchová užitečné informace o pohybu. Navrhneme metodu nazvanou BSD, která takovéto objekty detekuje pomocí odečítání pozadí, k-nejbližších sousedů a predikce pohybu. Naši metodu vyhodnotíme na kolekci devíti videosekvencí, které pokrývají rozmanitou škálu jevů, a ve kterých se objevují různě rozmazané objekty. Rozebereme, proč v některých případech detekce selže, a navrhneme řešení pro vylepšení. Mimo to uvedeme příklad reálného využití této metody a sice při umělé tvorbě zpomalených záběrů.

Klíčová slova: detekce objektů, sledování objektů, rozmazání pohybem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In computer vision, detection is the task of finding positions of certain objects in digital images. Almost everyone has used a detector already, either in a digital camera, smartphone or on social media – a face detector [21]. Detectors are build to detect instances of an object class like human faces or very specific objects – specific person in a crowd. There are limitless areas of application for example: monitoring highway traffic, surveillance, image retrieval or automatic inspection of defective parts. One particularly interesting is detection of female mosquitoes[1] and then shooting them with lasers to prevent malaria from spreading.

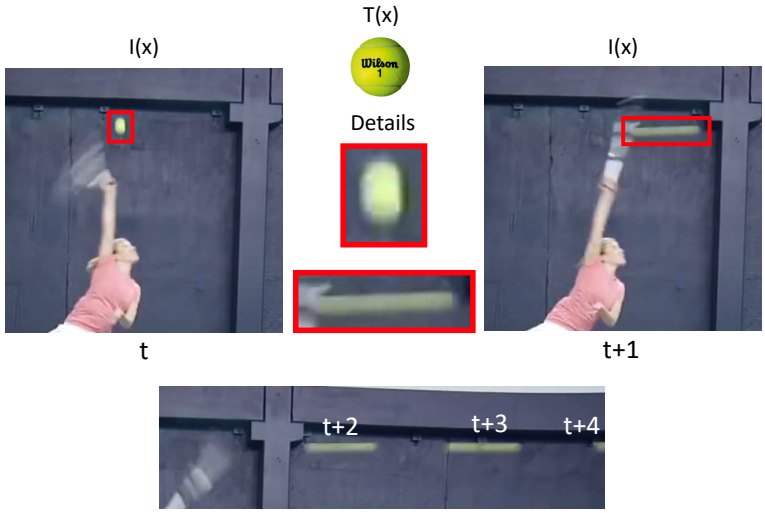


Figure 1.1: Tracking by detection of very fast objects.
Current detection methods are localizing the object template $T(x)$ in an image $I(x)$. In this case it is very hard, because the difference between the template and the object appearance at frames $t, \ldots, t+4$ is significant.

[1] <http://www.intellectualventureslab.com/work/photonic-fence>

Digital video is a sequence of digital images taken at regular time interval. A face detector can be used to detect a human face in an individual frame and a specific detector can be trained on this particular face. In the subsequent frames this face can distinguished and tracked throughout the whole video sequence – this is called *tracking by detection*.

Visual object tracking (sometimes also video tracking) [9] is a task of estimating the pose of objects of interest in all frames of an image sequence. Tracking is one of the most challenging computer vision problems and thanks to rapid improvement of image quality and resolution as well as increase in computational power over the past decade it is a fast evolving field as well.

The main challenges complicating visual object tracking are similar appearance between object of interest and background (specific person in a crowd), changes in pose (front view rotating to side view), illumination changes (indoor to outdoor transition or weather changes), partial and total occlusion (person behind a car) and noise (depends on sensor quality).

The output of a visual object tracking method depends on the area of application, more precisely the accuracy requirement. The object of interest may be represented by a single point in the image, an axis-aligned bounding box, a rotated bounding box, boundary pixels, a Kinect-like pose estimation [14] and more. Segmentation is the most precise one – labelling each pixel as object of interest or background.

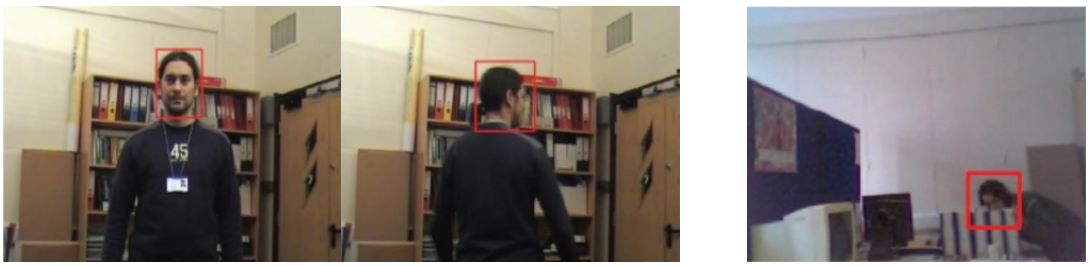


Figure 1.2: Challenges of visual object tracking.
The objects of interest are detected and shown in red axis-aligned bounding boxes. Image a) represents a change of target's pose, while image b) represents partial occlusion. Image source: Video Tracking: Theory and Practice [9]

Tracking is for example used in motion capture systems designed to track special markers attached to an actor's body or limbs and project their poses onto animated characters. Another use is in sport statistics to gather information about player position, from where they shoot and where they aim. It could be also used in advertising to check whether a certain advert was displayed for certain time in unpredictable scenarios like football, where adverts on stronger team's half will be less visible in television. In medical area it is used to track movement of bacteria for research of behaviour patterns or reproduction. The last one to mention but certainly not the least is surveillance – an example of multi-object tracking – on airports, roads, railway stations or ports.

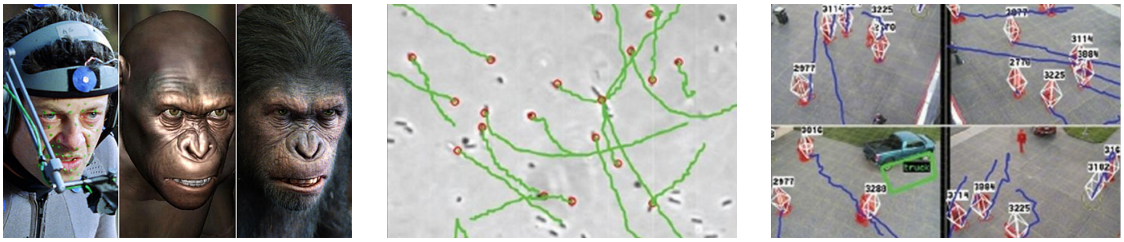


Figure 1.3: Applications of visual object tracking.
a) motion capture system, b) bacteria tracking and c) surveillance application . Image sources: a) ScreenCrush, ©2016, b) - c) Video Tracking: Theory and Practice [9].

Tracking by detection, object of this thesis, estimates object location in each frame independently and can handle occlusions better than other approaches. It keeps a model, usually in a template $T(X)$. It tries to find the template in an image $I(X)$. The model can be updated over-time but it has trouble with rapid changes, which can possibly appear as blurred stroke (see figure 1.1). Other than that, the template is often represented by interest points – features. Features often consist of corners or edges, which appear as high frequency in the image (fast changes over small area). Motion blur has a smoothing effect, thus removing high frequencies and leaving us with no features from $I(X)$ to match template features $T(X)$.

The contribution of this project is a proposal of a method to find fast objects that are in a single frame covering distance larger than their size and to track them over time. It is very hard to find blurred strokes in a single image so we utilize the fact that they are moving and we detect them in video sequences. Such objects lack high frequency details for feature matching approaches and have significantly different appearance than their non-blurred counterparts. From properties of a motion blur we will estimate a trajectory, which can be used for deconvolution. Deconvolution is a process of estimating how object may look like when it is not moving. When we know the appearance of static object, we can place it along the trajectory and make artificial frames which will appear as slow motion video. This project is motivated by the lack of methods covering presented problem and the fusion of visual object tracking and super resolution.

The application of this technique could be in creation of artificial slow motion video. This could be further applied in sports as a Hawk-Eye[2] like system. A Hawk-Eye is mostly known from tennis but is also used in other sports such as football, cricket or badminton. Its goal is to create a three dimensional representation of the trajectory of the ball and show a close-up of the situation, which was undecidable by referee.

[2] <http://www.hawkeyeinnovations.co.uk/>

Chapter 2

Background

In this chapter we explain how motion blur appears and how it can encode information about motion – section 2.1. Definition of the convolution is presented in section 2.2 as well as how it can be used to estimate object appearance when it is not moving – section 2.3. Finally a short introduction to super resolution is presented in section 2.4

2.1 Shutter speed and frame rate

A video camera has a limited temporal resolution which is determined by two parameters – the shutter speed and the frame rate. The shutter speed [13] is a property of a video camera and controls the level of blur. It is the time the shutter is open and light is being captured on image sensor, expressed in fractions of a second: $\frac{1}{30}s$, $\frac{1}{60}s$, $\frac{1}{120}s$ etc. Slower shutter speed means larger motion blur, because the moving object covers a longer distance.

When it comes to capturing a video, we need to factor in another property of a video camera – the frame rate [12]. The frame rate refers to number of frames captured each second. The most common ones offered by non-professional video cameras are 24, 25, 30 or even 60 FPS (frames per second). This limits the range of the shutter speed – for example we cannot choose 30 FPS and shutter speed $\frac{1}{15}s$, because this would require exposures of consecutive frames to overlap. The relation between shutter speed and frame rate is shown in figure 2.1.

Faster shutter speed leads to less motion blur, but from the figure 2.1 we can clearly see, that we lose a lot of information about trajectory of such fast moving object. For our method it is advantageous to have shutter speed equal to $\frac{1}{frame\ rate}$ to maximize information gain captured in motion blur. However majority of videos are captured with different shutter speed to frame rate ratio. The most natural looking motion perceived by humans is produced by shutter speed equal to $\frac{1}{2*frame\ rate}$ [17].

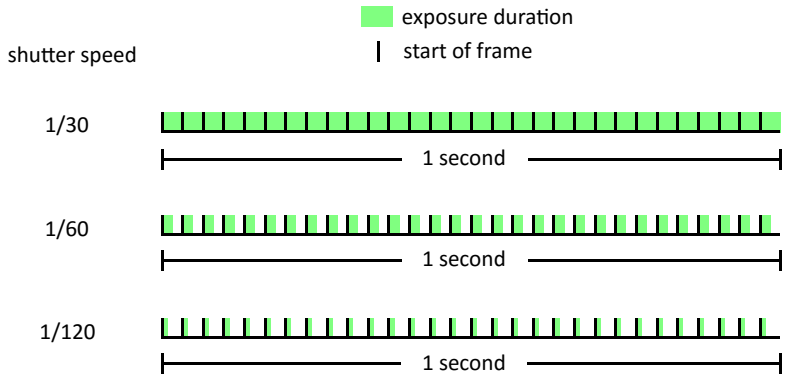


Figure 2.1: Different shutter speeds at 30 FPS.
In green sections camera shutter is open and light (information) is collected by the image sensor. Please notice the loss of information at lower shutter speeds.

2.2 Convolution

Convolution [22] is a mathematical operation on two functions denoted by asterisk symbol. It is defined as a integral of a product of two functions but one is reversed and shifted.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

Convolution for discrete terms is defined as

$$(f * g)(n) = \sum_{m=-\infty}^{\infty} f(m)g(n-m) = \sum_{m=-\infty}^{\infty} f(n-m)g(m)$$

When it comes to image processing, we need to extend this convolution into two dimensions, which is done by adding indices for the second dimension

$$(f * g)(m,n) = \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} f(k,l)g(m-k,n-l).$$

In practice however, we compute convolution for finite intervals. We define one small matrix (usually called kernel) with specific element values which suit our desired operation – blurring, sharpening, edge detection and more.

For example edge detection can be done by the use of kernels K_x for vertical edges and K_y for horizontal edges (also known as Sobel operator [18]).

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & -1 \end{bmatrix}$$

Example of convolution with Sobel kernel can be seen in figure 2.2.

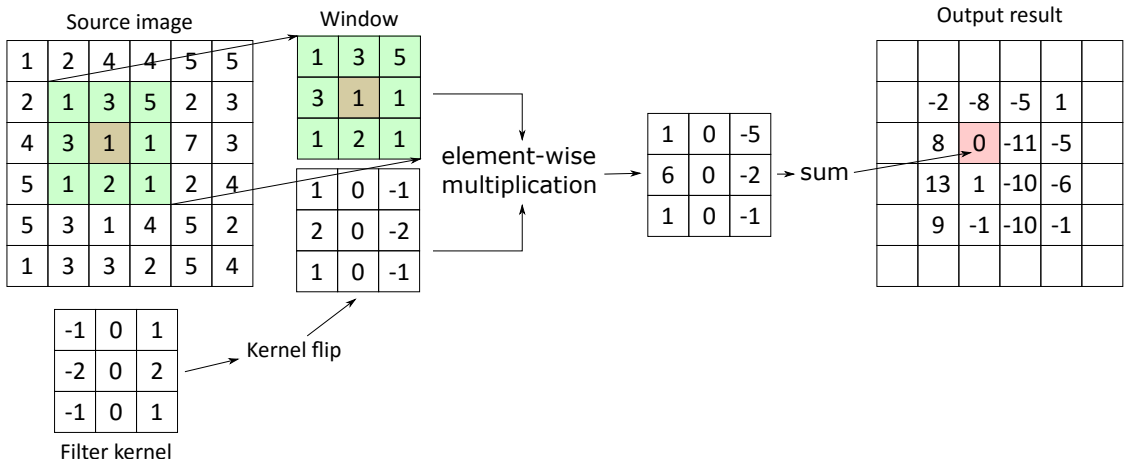


Figure 2.2: 2D convolution

2.3 Point spread function

Common approach to describe a motion blur is through point spread function (PSF) [20]. PSF is a convolution kernel (see section 2.2). Motion blur can be modelled by following noised convolution process

$$I_b(x, y) = I_o(x, y) * h(x, y) + \eta(x, y), \tag{2.2}$$

where x and y are pixel coordinates, $I_b(x, y)$ is the image with motion blur, $I_o(x, y)$ is the original image without motion blur, $h(x, y)$ is the point spread function and $\eta(x, y)$ is additional noise.

For motions following a straight line, PSF can be parametrized by an angle θ and a length l. But for more complex motions (like ball bouncing off a wall at high speed) we need exact PSF $h(x, y)$. With the trajectory knowledge we could restore original image through a process called deconvolution (reverse to convolution). The object of deconvolution is to find the most likely original image $I_o(x, y)$ given $h(x, y)$, $I_b(x, y)$ and properties of the noise. Since there is noise $\eta(x, y)$ present, the solution to deconvolution may be incorrect.

2.4 Super resolution for sequences

When it comes to video sequence, we have two separate resolutions – spatial and temporal. The spatial resolution [15] is related to physical space. The higher the spatial resolution,

the more information we can get out of it, which is desired for most electronic imaging applications. The captured spatial resolution is limited by the used image sensors, which are getting better over time, but their high cost is an important concern in many commercial SR (super resolution) applications. Most modern approaches [15] fuses more low-resolution images into one with higher resolution. Spatial resolution is not and object of this thesis.

The temporal resolution is related to time. A standard video camera records at speed of 30 FPS (frames per second) which is equal to an interval of 0.033 seconds, whereas high speed camera can record at speed of 10,000 FPS or higher, thus having temporal resolution of 0.0001 seconds or lower. It may seem reasonable to use high speed camera for better temporal resolution, but this approach comes with drawbacks. High speed cameras require additional lightning, which may not be available in some scenarios. Since they capture tremendous amount of images each second, the amount of captured time is limited thanks to local memory to several second. In order to capture high temporal resolution they are reducing spatial resolution. The last but not least is the purchase price.

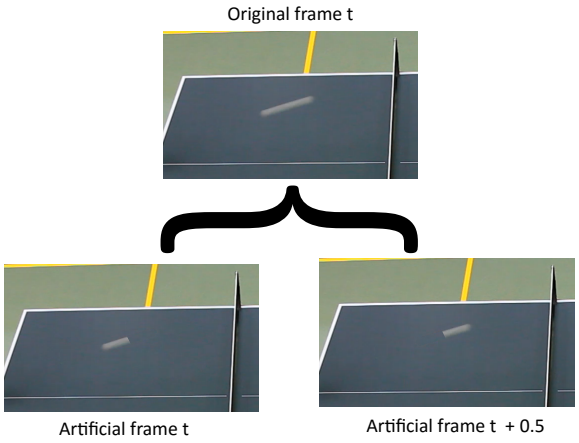


Figure 2.3: N times super resolution visualization.
Each frame is split into N artificial frames. This figure shows two times super resolution. Artificial frames are supposed to be made as convolution of true object appearance with $1/N$ portion of the trajectory. Unfortunately this is outside the scope of this project, so simpler method was used.

Chapter 3

Related work

3.1 Tracking by detection

Tracking by detection is a common approach to visual tracking as it can handle situations when the object of interest leaves the scene and reappears later or in the case of heavy occlusion. This is one of the main problems of **long-term tracking** in which the tracking process should run "indefinitely".

The detector is trained on an object appearance in the initial frame, but this appearance will in most cases become less relevant over time, as the object may change pose, scale or the lighting in the scene may change. This problem is investigated in Tracking-Learning-Detection by Z. Kalal et al. [7]. They use short-term trackers in addition to detector. Short-term trackers estimate object motion and only require initialization, but accumulate error over time resulting in a drift. On the other hand, detectors do not drift, but require a learning stage. Smart cooperation of those components could be beneficial for both. They use detector to re-initialize trackers, thus minimizing drift, and trackers to provide weakly labeled training data for detector. Since the training of the detector is done with unreliable data, there is a high chance of false positives (wrong detections). In order to compensate detector errors and learn from its mistakes, they use two experts: P-expert to detect false negatives and N-expert to detect false positives – providing both positive and negative samples for detector model.

Other state-of-the-art tracking by detection approach is Struck by S. Hare et al. [6], which can adapt to changes of tracked object. It uses structured support vector machine which is a generalization of SVM classifier. This structured SVM is learned online with positive and negative samples. They also introduced budgeting mechanism to keep number of support vectors low which allows real-time applications.

Tracking is a rapidly evolving field, every year dozens of new tracking algorithms are presented, but authors are using different evaluation protocols, thus it is getting harder and harder to compare them against each other. This was one of the reasons VOT Challenge [1] was made in 2013. Since then a competition of visual tracking algorithms is announced every year. Evaluation is done unanimously with novel performace evaluation methodology [8].

[1] <http://www.votchallenge.net/>

3.2 Blur

It may seem easy for human to recognize blurred regions in the image, but it is not easy to formulate an effective algorithm for a machine. Many approaches [20] are expecting some form of PSF and estimating its parameters for deconvolution.

A different approach to blur detection was proposed by Pina Marziliano et al. – A no-reference perceptual blur metric [10]. They exploit the fact that blur has a smoothing effect on edges. After detecting edges in vertical direction (e.g. using Sobel operator 2.1), they measure the width of the edge by finding local minima and maxima in pixel values around the edge. The final blur measurement is calculated as sum of all edge widths divided by number of edges. Advantages of this approach is low computational complexity and no-reference property.

A method by Shengyang Dai and Ying Wu [2] is recovering a motion of blurred object from a single frame. They use image matting techniques which decompose image I as linear combination of foreground image F and background image B. The constant of this combination is called α-channel.

$$I(x,y) = \alpha(x,y)F(x,y) + (1 - \alpha(x,y))B(x,y)$$

They assume that under good lightning conditions, most image boundaries are clear and sharp for slow moving or static objects. This means that most α values will be binary – either 1 for foreground or 0 for background. Parameters of motion blur are further found by looking at non-zero gradient of α-channel. Further they slice the initial image into smaller ones along the grid and estimate motion of each sub-image. By this approach they can cover more complex motion blurs.

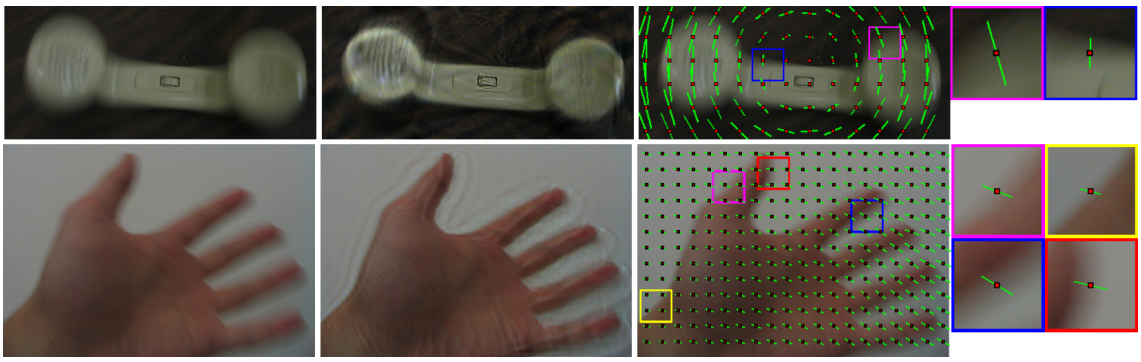


Figure 3.1: Motion from blur results.
Figure shows results of Motion from blur [2] on rotational (top) and non-parametric (bottom) motion blur data. From left to right: blurred image, deblurred result, visualization of estimated motion blur vector field and close-up views. Image source: Motion from blur [2].

3.3 Super resolution

Temporal resolution of a video sequence is limited by the camera's shutter speed and frame rate (see section 2.1). Events occurring faster than the frame rate lead to motion blur. Temporal super resolution tries to avoid this by producing artificial frames, where the temporal resolution is high enough to avoid motion blur.

One of the breakthroughs is a method called coded exposure photography [16] – a method for image deblurring. This method requires modifications of the capturing device – its shutter. This means that this method will not work on any sequence obtained by a device without this modification. The only requirement is to put an external electric shutter in front of the camera lens and to synchronize it with the start of the exposure time. When using conventional shutter, motion blurred objects lose important high-frequency spatial information (e.g. edges). Instead of leaving the shutter open for the entire exposure duration, they open and close external shutter rapidly using a pseudo-random binary sequence. This approach preserves high-frequency spatial details and together with known pseudo-random sequence the deconvolution (see section 2.3) becomes well-posed problem. A figure 3.2 was included for better understanding of this approach and its results.

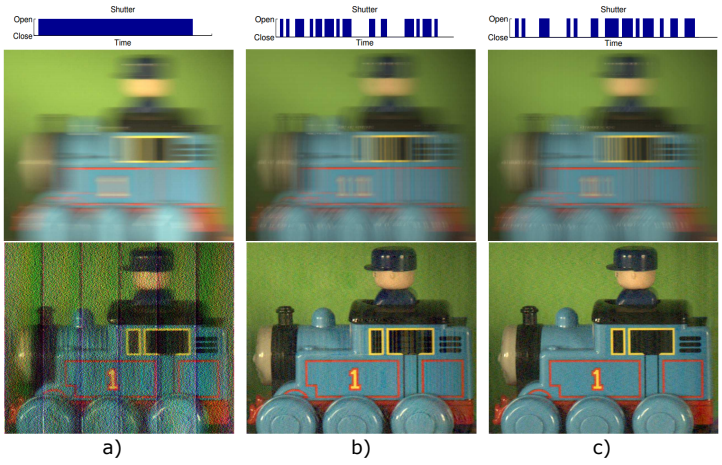


Figure 3.2: Comparison between different deblurring techniques.
Figure shows different shutter exposures (top), captured images (middle) and deblurred images (bottom), traditional exposure (a) and coded exposures (b, c). Image source: Coded exposure photography: motion deblurring using fluttered shutter [16].

Agrawal et al. [1] combine N low speed cameras to produce N times temporal super resolution with minimal possible reconstruction noise. They use coded exposure, but instead of pseudo-random binary sequence, special sequences for each individual camera is used. The main contribution of their work is a process of generating such binary sequences which allow more light to be captured. The optimal sampling is computed by minimizing the mean square error of a binary matrix, which corresponds to closed or opened shutter. This approach is dependent on the geometric calibration of cameras. The calibration error may lead to wobbling artifacts in the reconstructed frames.

Chapter 4

Problem formulation

In this section we will decompose tracking by detection of fast moving objects into several sub-problems. Each of them will be discussed separately together with an explanation why was this approach used, what are other alternatives or advantages and disadvantages of proposed technique. In following sections the notation displayed in table 4.1 will be used.

Variable	Definition
$t \in \mathbf{N}$	frame number
$w \in \mathbf{N}$	width of input image
$h \in \mathbf{N}$	height of input image
$s \in \mathbf{Z} \times \mathbf{Z}$	true object appearance
$s_t \in \mathbf{Z} \times \mathbf{Z}$	true object appearance in frame t
$r_t \in \mathbf{Z} \times \mathbf{Z}$	blurred object appearance in frame t
$I_t(x) \in \mathbf{N}_0^{w \times h \times 3}$	input image, frame t, pixel x
$p_t(\tau) \in \mathbf{R}^2$	point in trajectory in frame t in time $\tau \in [0; 1]$
$p_t \in \mathbf{R}^2$	travelled path in frame t
$v_t \in \mathbf{R}$	objects velocity magnitude in frame t
$\alpha_t \in [0, 1]$	objects opacity in frame t
$c_t \in [0, 255]^3$	color of foreground (object) in frame t
$b_t(x) \in [0, 255]^3$	background color of pixel x in frame t
$\overset{\Delta}{I}_t(x) \in \mathbf{Z}^{w \times h \times 3}$	difference image

Table 4.1: Notation

4.1 Assumptions

For simplification we assume nearly static background.

$$b_{t+1}(x) \approx b_t(x). \tag{4.1}$$

Relaxation of this restriction will be done in future work. We would need to align backgrounds by some transformation function f, but for now under assumption of static back-

ground, we define f as identity.

$$\overset{\Delta}{I}_t(x) = I_{t+1}(x) - I_t(f(x)) \tag{4.2}$$

Since an object travelling at high velocity appears as blurred stroke and is probably rotating as well, we assume its final color is an integral over all of its colors, and does not change much in two consecutive frames

$$c_{t+1} \approx c_t. \tag{4.3}$$

And because the time interval between two frames is very short, we suppose that objects acceleration is negligible, thus

$$v_{t+1} \approx v_t. \tag{4.4}$$

As in many matting algorithms [2], we assume that each image pixel $I_t(x)$ is a convex combination of foreground c_t and background $b_t(x)$.

$$I_t(x) = (1 - \alpha_t(x))b_t(x) + \alpha_t(x)c_t, 0 \leq \alpha_t(x) \leq 1. \tag{4.5}$$

Because of negligible acceleration 4.4 the color of the object will be spread uniformly into blurred stroke meaning constant $\alpha_t(x)$ for foreground and zero $\alpha_t(x)$ for rest of the scene.

4.2 Difference image

Difference image or a background subtraction is a technique to obtain image foreground which usually contains objects of interest. This process is often preceded with other image processing techniques that align two images so that they backgrounds overlap as much as possible. Background subtraction strongly relies on static background assumption 4.1 which is often not applicable in real-world environments. Even when the camera is static, there may be reflections, illumination changes or weather phenomena (wind, rain) happening in the background and thus affecting results of background subtraction. These artifacts can be seen in figure 5.1. Difference image is defined as

$$\begin{aligned} \overset{\Delta}{I}_t(x) &= I_{t+1}(x) - I_t(x) \\ &= (1 - \alpha_{t+1}(x))b_{t+1}(x) + \alpha_{t+1}(x)c_{t+1} - [(1 - \alpha_t(x))b_t(x) + \alpha_t(x)c_t] \end{aligned} \tag{4.6}$$

We will get different results for different pixels x, more precisely

$$\overset{\Delta}{I}_t(x) \approx \begin{cases} 0 & x \text{ from background} \\ \alpha(b_t(x) - c) & x \text{ from object at time } t \\ \alpha(c - b_t(x)) & x \text{ from object at time } t+1 \end{cases} \tag{4.7}$$

Each case is analyzed separately in following subsections.

4.2.1 Background pixels

When subtracting background pixels $b_t(x)$ from background pixels $b_{t+1}(x)$, both $\alpha_t(x)$ and $\alpha_{t+1}(x)$ are zero. Therefore we get

$$\overset{\Delta}{I}_t(x) = b_{t+1}(x) - b_t(x) \tag{4.8}$$

Furthermore after using 4.1 we get

$$\overset{\Delta}{I}_t(x) \approx b_t(x) - b_t(x) \approx 0 \tag{4.9}$$

4.2.2 Pixels from blurred stroke at time t+1

From the blurred stroke at frame $t+1$, we are subtracting background $b_t(x)$ and we get

$$\overset{\Delta}{I}_t(x) = (1 - \alpha_{t+1}(x))b_{t+1}(x) + \alpha_{t+1}c_{t+1} - b_t(x) \tag{4.10}$$

Using assumptions 4.1, 4.3 and 4.4, the equation 4.10 is simplified to

$$\overset{\Delta}{I}_t(x) \approx \alpha_t(c_t - b_t(x)) \tag{4.11}$$

4.2.3 Pixels from blurred stroke at time t

When subtracting blurred stroke at frame t from the background $b_{t+1}(x)$ we get

$$\overset{\Delta}{I}_t(x) = b_{t+1}(x) - [(1 - \alpha_t(x))b_t(x) + \alpha_t(x)c_t] \tag{4.12}$$

For simplification of 4.12, assumptions 4.1, 4.3 and 4.4 are used:

$$\overset{\Delta}{I}_t(x) \approx \alpha_t(b_t(x) - c_t) \tag{4.13}$$

Figure 4.1: Difference image.

Note: Difference image has values that are negative. For the purpose of visualization each color channel was shifted by the minimal value of this channel and then rescaled to interval $[0, 255]$.

4.3 Trajectory

The major portion of the trajectory of the fast moving object is encoded in the captured motion blur. We would like to extract this trajectory and describe it as a function of time. For this purpose $p_t(\tau) \in \mathbf{R}^2$ is defined as a function of parameter $\tau \in [0; 1]$ returning two-dimensional image coordinates. The direction of the movement also matters, thus we define $p_t(0)$ as a starting point of the trajectory and $p_t(1)$ as the ending point. Please notice that we cannot estimate the direction of the movement from a single frame (see figure 4.3), and we need to combine knowledge from at least two frames.

Camera shutter opens at the start of the frame, but can close sooner than the end of the frame, preventing light from reaching the image sensor and losing much needed information about trajectory. This phenomenon is explained in detail in section 2.1. The ending point of the trajectory travelled during open shutter at frame t is defined as

$$p_t(\tau_t^M), 0 \leq \tau_t^M \leq 1. \tag{4.14}$$

We require a contiguous trajectory between multiple frames

$$p_{t+1}(0) = p_t(1). \tag{4.15}$$

From the knowledge of the next starting point $p_{t+1}(0)$ define points $p_t(\tau_t^M) \ldots p_t(1)$ as points on line going from $p_t(\tau_t^M)$ to $p_{t+1}(0)$.

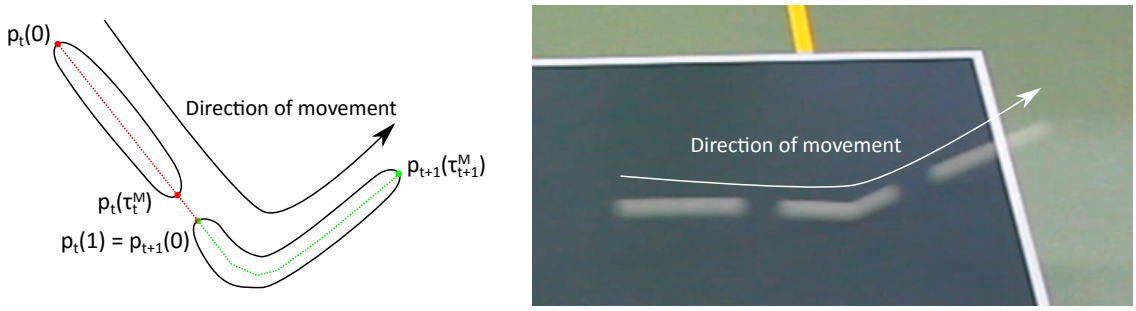


Figure 4.2: Trajectory of a blurred stroke.
Illustration (on the left) of a blurred stroke's trajectory and real-world example (on the right).

4.4 Motion prediction

Motion predictors use objects dynamics obtained from previous frames to predict object's position in the following frame. Such predictions can narrow our search to smaller area saving computational power. Motion predictors can be complex, taking into account all object's positions in the sequence or even use some motion model from previous behaviour studies [19]. On the other hand the simplest motion predictors are no motion and constant motion, which require only one previous frame. Fast moving object will usually follow ballistic trajectory when there is no contact. Direction changes in ballistic curve occurs over

longer periods of time and constant motion predictor is a sufficient method to handle those changes. This prediction will not work when there is a contact and the trajectory changes significantly. But in order to calculate the angle of reflection, we would need to know how the surface looks like, which is even harder problem, so we may as well call the reflection random.

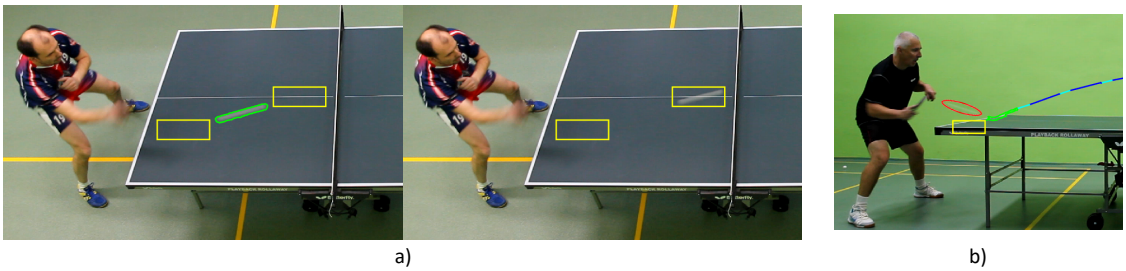


Figure 4.3: Visualization of motion prediction.
In the left picture a), a blurred stroke is detected and its boundaries are shown in green. The velocity vector is estimated from the parameters of motion blur and constant motion predictor guesses two areas (yellow boxes) where the object might appear in the next frame. To form a contiguous trajectory and estimate the direction of the motion in current frame, we would need a detection in the previous frame. Since there was none, we consider both directions.
In the right picture a) we can see position of blurred stroke in the next frame and predicted areas from previous frame.
In the picture b) the motion prediction (yellow box) is wrong. The object bounces off the table and appears in the red ellipse.

4.5 Deconvolution

We would like to further estimate the true objects appearance from the blurred stroke. This problem can be solved using deconvolution, which is unfortunately an ill-posed problem – meaning that a tiny change on the input may lead to a huge output difference – and is the reason why it is also called blind deconvolution. Rather than blindly estimating the convolution kernel, we can predict it using the knowledge of travelled trajectory in the captured image. For this purpose travelled trajectory in the frame t is defined as a set

$$p_t = \{p_t(0), \ldots, p_t(\tau_t^M)\}. \tag{4.16}$$

The true object appearance s means how the object looks when it is not moving. Our concern is that every object moving at high speed is possibly rotating at high speed as well and the deconvoluted object will most likely look like a sphere. Further more it is impossible to reconstruct fine details (such as the white lines on tennis ball), because motion blur destroys important high-frequency information. There is a solution to the loss of high-frequency information and it has been discussed in section 3.3.

Thanks to the knowledge of the travelled path in the image, resulting blurred stroke can be perceived as a convolution

$$r_t = s_t * p_t, \tag{4.17}$$

where r_t is known blurred appearance, p_t is estimated trajectory and s_t is true object appearance. With each additional frame N, where this blurred stroke appears, we can refine final object appearance s as a mean of $s_0, s_1 \dots s_{N-1}, s_N$.

Chapter 5

Proposed algorithm

An overview of the whole BSD method is shown in the algorithm 1. In sections 5.1 – 5.7 individual parts of the algorithm are explained in detail.

5.1 MATLAB

The BSD algorithm is written in MATLAB [11]. MATLAB (abbreviation for Matrix laboratory) is a programming language mainly focused on matrix manipulations, plotting of functions and data, creation of user interfaces and much more. MATLAB was chosen because images are stored in matrix-like form and MATLAB offers Computer Vision System Toolbox[1] which can facilitate a development of such algorithm. For future work it is suggested to reimplement this algorithm in a language such as C++ for a faster performance and an overall better control over algorithm components.

5.2 Difference and binary image

The difference image is computed as a difference of 2 frames (see figure 4.1), which is straightforward in MATLAB. Because 4.9 is only approximately zero, a small used-defined threshold is used. A binary image is produced with ones on pixels where the difference was higher than threshold and zeros elsewhere.

5.3 Connected components

Using the binary image 5.2, we compute connected components. A connected component – term from graph theory – is a subgraph in which any two vertices are connected by edges and are not connected to additional vertices in the supergraph.

When using connected components in computer vision (also known as blob extraction or region labelling), a graph is constructed – each pixel is corresponding to a vertex and neighbouring pixels are connected by edges. Either 4-connectivity (only horizontal and vertical

[1] <http://www.mathworks.com/help/vision/index.html>

Algorithm 1: Blurred stroke detector (BSD)

function BSD
$(image_t, image_{t+1}, cc_{t-1}, prediction_t, object_{t-1}, trajectory_{t-1}, noiseThr, maxDst)$;

Input : Two consecutive frames from a video, previous connected components, motion prediction, previously detected object, previously estimated trajectory, noise threshold for background subtraction, maximal distance for KNN matching

Output: Connected components, prediction for future frame, object pose, estimated trajectory

$differenceImage_t \leftarrow image_{t+1} - image_t$;
$binaryMask_t \leftarrow differenceImage > noiseThr$;
$cc_t \leftarrow getConnectedComponents(binaryMask)$;
if $notEmpty(prediction_t)$ **then**
    $object_t \leftarrow cc_t$ from $prediction_t$ with closest color to $object_{t-1}$;
    if $object_t$ *not similar to* $object_{t-1}$ **then**
        return $BSD(image_t, image_{t+1}, cc_{t-1}, [], object_{t-1}, trajectory_{t-1}, noiseThr, maxDst)$
    end
else
    $o, distance \leftarrow knnMatch(cc_{t-1}, cc_t)$;
    if $distance < maxDst$ **then**
        $object_t \leftarrow o$;
    else
        $object_t \leftarrow []$;
    end
end
$prediction_{t+1} \leftarrow makePrediction(object_t)$;
$trajectory_t \leftarrow estimateTrajectory(object_t, trajectory_{t-1})$;
return $cc_t, prediction_{t+1}, object_t, trajectory_t$;

neighbours are connected) or 8-connectivity (all neighbours are connected – horizontal, vertical and diagonal) can be used. In addition we can get some useful information about each connected component – its center of mass, a major axis length, an orientation, boundary pixels and a mean color. This information is used by motion predictor to speed up detection of blurred stroke in following frame.

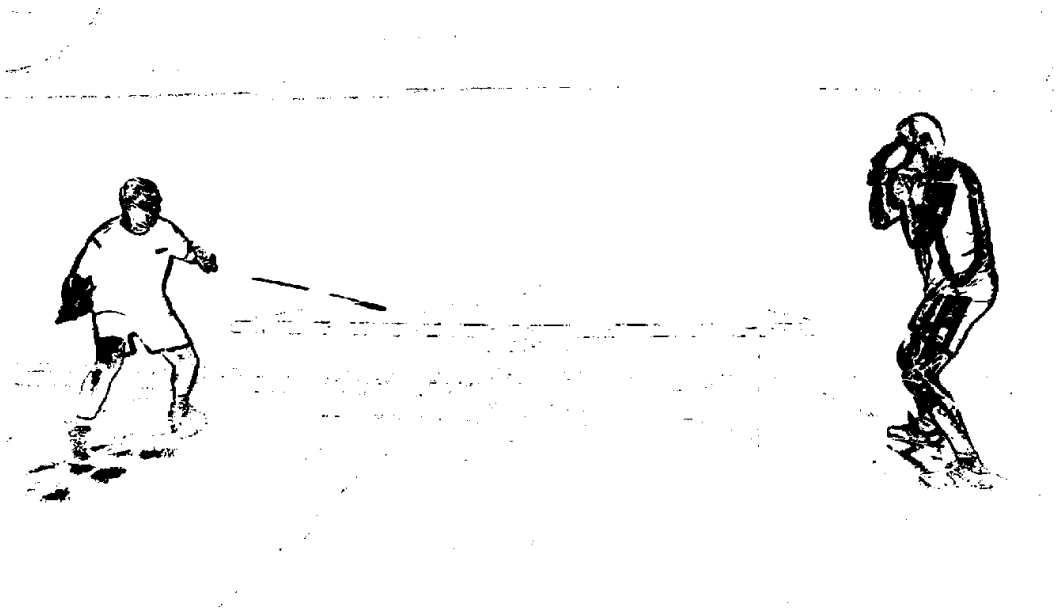


Figure 5.1: Connected components of foreground objects.
This binary image was inverted for better visualization. Black pixels represents binary value 1 (moving objects), white pixels value 0 (background).

5.4 K-nearest neighbours

K-nearest neighbours (KNN for short) [4] is a non-parametric method used for classification and regression. A non-parametric technique does not make any assumptions on the underlying data distribution. This property is very useful, because most of the real world data does not follow typical assumption (e.g. gaussian distribution). KNN requires that the data is in a feature space, in another words data can be scalars or vectors. For such data we can define a distance function, the one commonly used is Euclidean distance. The single number "k" decides how many neighbours influence the classification.

One of the inputs to KNN algorithm is a training set. Such training set Γ is consisted of n measurements $x \in X$ with corresponding classes c. The other input is a distance function d.

$$\Gamma = \{(x_1, c_1), \ldots, (x_n, c_n)\}$$

$$d = X \times X \to \mathbf{R}_0^+$$

When we want to classify a measurement y, we find set S of k nearest measurements from the training set Γ ordered by distance function $d(y, x_i)$.

$$r_i : \text{ the rank of } (x_i, k_i) \in \Gamma \text{ given by ordering } d(y, x_i)$$

$$S = \{(x_{r_i}, c_{r_i}), \dots (x_{r_k}, c_{r_k})\}$$

The final class assigned to measurement y is class c which has the majority in S.

$$c = \underset{l \in \mathbf{R}}{\operatorname{argmax}} \sum_{i=1}^{k} [\![c_i = l]\!], (x_i, c_i) \in S$$

Our measurements consist of connected components properties, namely center of mass and mean color. In two consecutive difference images $\overset{\Delta}{I}_t(x)$ and $\overset{\Delta}{I}_{t+1}(x)$, we find such connected components, that their centres of mass are as close as possible and their mean colors are as close to being opposite as possible (see 4.7). Such connected components are labelled as tentative blurred strokes and are further checked against trajectory constraint (see 4.15).

5.5 Motion prediction

Constant movement predictor is used. The object velocity vector is estimated by connected component's major axis length and orientation. In the next frame KNN approach is not used, instead connected components in predicted area are checked for color similarity with detected object in the previous frame. If no such connected component exists or its properties are far away from previous blurred stroke properties, we run this frame again using the KNN approach.

5.6 Skeletonization

Skeletonization is a process for reducing regions in a binary image. A skeleton preserves the extent and connectivity of the original region while discarding most of the original pixels.

We use skeletonization to reduce blurred strokes into their trajectories. We extract boundary pixels for each blurred stroke and define two pixels – entry and exit. Those are obtained as nearest boundary pixels between two blurred strokes, thus we need 3 frames ($t-1$, t, $t+1$) to obtain entry and exit boundary pixels for blurred stroke at t. Since the boundary pixels are ordered in a clockwise direction, we can easily obtain two pixel lines – one going from entry pixel to exit in clockwise direction and the other in counter-clockwise direction. We than compute skeleton pixels which are equidistant from those line pixels. Since the pixel lines does not to be equal in length, linear mapping is used (see figure 5.2). We further approximate skeleton by straight lines, this process is described in following section.

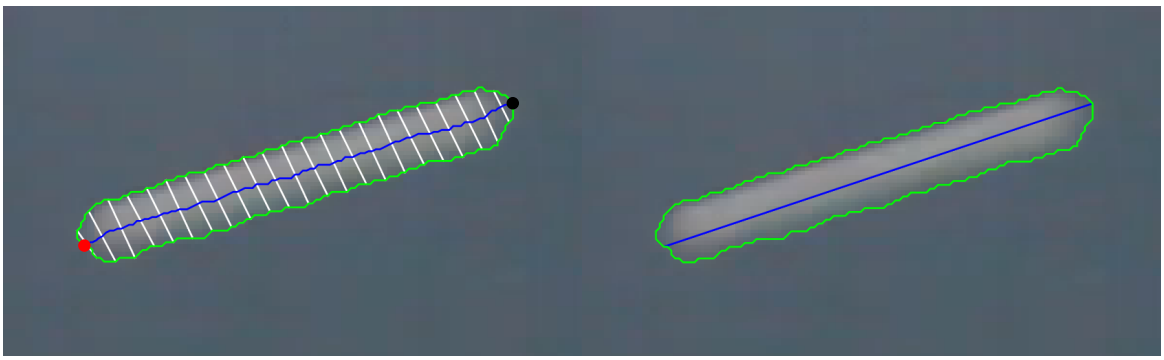


Figure 5.2: The skeletonization and line approximation.
In the left image: entry point (red), exit point (black), points along the boundary in the same distance from entry point in clockwise and counter-clockwise directions are connected by white lines (only few are shown). Final skeleton is formed from centres of white lines (blue line).
In the left image: Approximation of skeleton by single line.

5.7 Trajectory line approximation

Ramer–Douglas–Peucker [3] algorithm is used to further reduce number of points in skeleton by line segments aproximation. This algorithm takes as input ordered set S of n points and maximal error $\epsilon > 0$. It approximates this set of n points by a line intersecting 1^{st} and n^{th} point from this set. Then furthest point $i \in S$ from this line is found. If its perpendicular distance to the line is lower than maximal error ϵ, approximation is complete using this line. Otherwise this algorithm is called recursively with first set containing points $1 \dots i \in S$ and second set containing points $i \dots n \in S$ and results are merged.

Algorithm 2: Ramer–Douglas–Peucker algorithm

```
1  function RDP (points, ε);
   Input  : Ordered set of 2D points, maximal error ε > 0
   Output: Ordered set of 2D points
2  maxDistance ← 0;
3  index ← 0;
4  for i ← 1 to length(points) − 1 do
5  |  d ← perpendicularDistance(points[i], Line(points[0], points[end]));
6  |  if d > maxDistance then
7  |  |  maxDistance ← d;
8  |  |  index ← i;
9  |  end
10 end
11 if maxDistance > ε then
12 |  rec1 ← RDP(points[0...index], ε]);
13 |  rec2 ← RDP(points[index...end], ε]);
14 |  result ← {rec1[0...length(rec1)], rec2[1...length(rec2)]};
15 else
16 |  result ← {points[0], points[end]};
17 end
18 return result
```

Chapter 6

Data

6.1 Input data

Input data to proposed algorithm is in a form of video sequence. MATLAB supports most of ordinary video file formats such as "avi", "wmv" or "mp4". For additional supported file formats visit MATLAB documentation[1]. Another format of input data widely used in computer vision is frame by frame images, for example used in VOT Challenge[2]. In order to use this format please use provided code to convert single frames into a video sequence.

6.2 Output data

The output of BSD algorithm are frame by frame binary masks of blurred strokes as well as their estimated trajectory. Binary masks contains 0 for background pixels and 1 for blurred stroke pixels. The trajectory is represented by continuous line segments with time stamps so the whole trajectory p_t can be obtained. This output is ready to be fed to another method performing deconvolution (see section 4.5) as well as temporal super resolution. Such method is outside the scope of this project and was replaced by simpler solution.

[1] <http://www.mathworks.com/help/matlab/ref/videoreader.html>
[2] <http://www.votchallenge.net/vot2015/dataset.html>

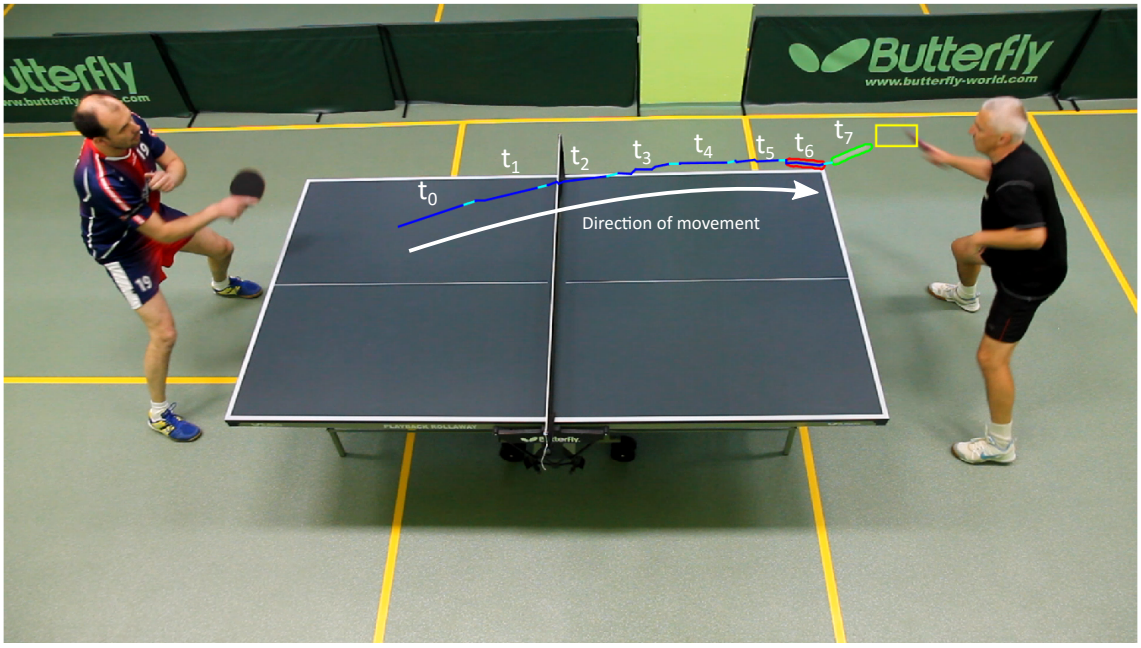


Figure 6.1: The output of the BSD algorithm.
The boundary pixels of detected blurred stroke are highlighted in green for t_7 frame and in red color for t_6 frame. The yellow box represents the prediction for the next frame. The trajectory is shown by a blue and cyan curve. Cyan segments are interpolated because camera shutter was closed, whereas the blue pieces were estimated from captured blurred stroke.

Chapter 7

Evaluation

During the evaluation of BSD detector, we encountered across several problems. Firstly, this area of computer vision is very much unexplored leaving us with no state-of-the-art methods for comparison. Secondly there is no standard dataset with video sequences containing blurred strokes. And finally during the annotation process of collected sequences, it was very hard to decide whether some objects should be considered as blurred strokes or not. We went with very strict approach which may have influenced these results significantly. Ideally we would like to obtain annotation from various sources and get final ground truth by averaging, but this would be both time and resource demanding.

Figure 7.1: Evaluation sequences.
From left to right, top to bottom: ping_pong_side, ping_pong_top, tennis_serve_side, tennis_serve_back, hockey, squash, tennis_1, tennis_2, william_tell.

7.1 Dataset

In order to find strengths and weaknesses of tested method, the dataset should consist of easy, intermediate and hard sequences and contain as many of possible real world phenomena as possible. For that reason we chose video sequences containing:

- blurred strokes in horizontal, vertical and diagonal directions
- sudden change of direction of movement (bounce)

- reflection
- noise / worse quality
- blurred stroke with similar color as background
- size change
- short blurred duration.

Other phenomena omitted due to our assumptions or overall complexity in computer vision are for example: illumination changes, partial and full occlusions, camera motion or image degradation. Preview of collected dataset is shown in figure 7.1 and detailed info is in table 7.1.

Sequence name	**Resolution**	**#frames**	**#blur visible**	**Phenomenons**
ping_pong_side	1920×1088	445	371	horizontal, bounce, size change
ping_pong_top	1920×1088	350	303	horizontal, bounce
tennis_serve_side	1280×720	68	17	horizontal, short
tennis_serve_back	1280×720	156	39	vertical, bounce
hockey	960×540	350	55	horizontal, reflection, bounce
squash	960×540	250	74	diagonal, reflection, bounce, noise
tennis_1	960×540	128	35	vertical, bounce, worse quality
tennis_2	960×540	278	216	vertical, bounce, worse quality
william_tell	1920×1080	119	20	horizontal, size change

Table 7.1: Evaluation sequences

We would like to thank Ing. Filip Šroubek, Ph.D. DSc. for providing sequences (a) - (d). Sequences (e) - (i) were obtained from YouTube[1], links to original videos are listed in Appendix B.

7.2 Annotation

Each frame in the dataset was annotated by a segmentation binary mask, where zero means background pixel and one stands for pixels in the blurred stroke. This ground truth annotation was made manually in GIMP[2] (GNU Image Manipulation Program).

7.3 Performance

To compute precision and recall, we first need to define accuracy of a detection. **Accuracy** or **overlap** [8] ϕ is computed for every frame t as the intersection-over-union

$$\phi_t = \frac{A_t^G \cap A_t^D}{A_t^G \cup A_t^D}, \tag{7.1}$$

[1] <https://www.youtube.com/>
[2] <https://www.gimp.org/>

where A_t^G is a ground truth binary image and A_t^D is mask of detected blurred stroke. Intersection $\cap$ is defined as a number of ones after binary "and" operation. Union $\cup$ is defined as a number of ones after binary "or" operation. In order to separate good and bad detections an accuracy threshold of 0.5 is usually used in bounding box evaluation [5]. But the only justification why this value is used is that it accounts for inaccuracies in ground truth. Considering very subjective ground truth annotation even correctly detected blurred stroke may have smaller accuracy due to classification of unclear boundary between blurred stroke and background. **Average overlap** $\bar{\phi}$ is defined as arithmetic mean of overlaps ϕ_t, where ground truth A_t^G was not empty.

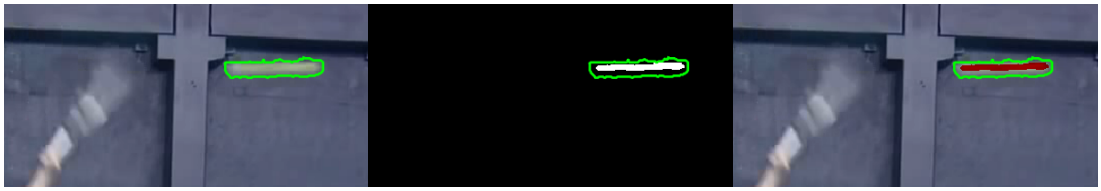


Figure 7.2: Ground truth and detection visualization.
Green line represents detection on a) input image, b) ground truth binary mask, c) combination of image and ground truth (in red).

We use the terms true positive, true negative, false positive and false negative to compare results of our detector with our ground truth. Interpretation of these terms is in table 7.2.

Outcome	**Interpretation**
true positive (TP)	blurred stroke is in the frame and it is detected ($\phi_t \geq 0.5$)
false positive (FP)	something is detected but somewhere else ($\phi_t < 0.5$)
true negative (TN)	blurred stroke is not in the frame and nothing is detected
false negative (FN)	blurred stroke is in the frame, but it is not detected

Table 7.2: Possible outcomes from hypothesis testing

Since wrong detection is worse than no detection, we prefer low false positive rate over false negative rate. This also means we are focusing on high precision over high recall.

Precision is defined as

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \tag{7.2}$$

and can be interpreted as a probability that a random detection is correct ($\phi_t \geq 0.5$).

Recall is defined as

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \tag{7.3}$$

and can be interpreted as a probability that a random blurred stroke will be detected.

7.4 Results

Our detector was run on all sequences from dataset 7.1 with same internal parameters. For each frame, an overlap ϕ_t was computed and overlap threshold of 0.3 was used to decide between TP or FP. We prefer low false positive rate, which is projected in higher precision, but lower recall.

Sequence name	TP	FP	TN	FN	Precision	Recall	Average overlap
ping_pong_side	255	3	71	114	98.8%	69.1%	44.6%
ping_pong_top	176	7	44	122	96.2%	59.1%	35.2%
tennis_serve_side	2	0	43	15	100.0%	11.8%	6.0%
tennis_serve_back	6	3	115	31	66.7%	16.2%	6.9%
hockey	7	52	248	41	11.9%	14.6%	5.8%
squash	2	30	143	74	6.3%	2.6%	2.3%
tennis_1	2	1	89	33	66.7%	5.7%	2.5%
tennis_2	2	3	61	210	40.0%	0.9%	0.5%
william_tell	0	0	85	20	0.0%	0.0%	0.0%

Table 7.3: Results for overlap threshold of 0.3

From the table of results 7.3 we see that this method is performing fairly well on sequences "ping_pong_side" and "ping_pong_top", where our assumption of static background was satisfied the most, the blurred strokes were the longest of all sequences and ballistic curve lasted for several frames. On sequences "tennis_serve_side" and "tennis_serve_back" most of the blurred strokes were not that long and the long ones lasted for only couple of frames. Sequences "hockey" has a shaky camera and shadow reflection which results in higher false positive rates. Most movement in background occurs in sequences "tennis_1" and "tennis_2", which is correctly recognized and it does not produce false positives but it has much stronger response than blurred strokes that are suppressed. Sequence "squash" is very hard, because the blurred stroke has very similar color as background and is often barely visible. The arrow in "william_tell" sequence appears as blurred on the sides, but is not moving very fast, because it is from a slow motion footage.

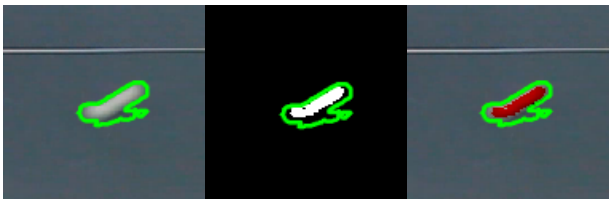


Figure 7.3: Ground truth and detection visualization of an object with shadow.
Green line represents detection on a) input image, b) ground truth binary mask, c) combination of image and ground truth (in red). When a fast moving object casts a shadow on a surface, this shadow appears in foreground and is a part of detected object.

7.5 Failure modes

In this section we will explain known problems and drawbacks of current approach that can be avoided by using different procedures.

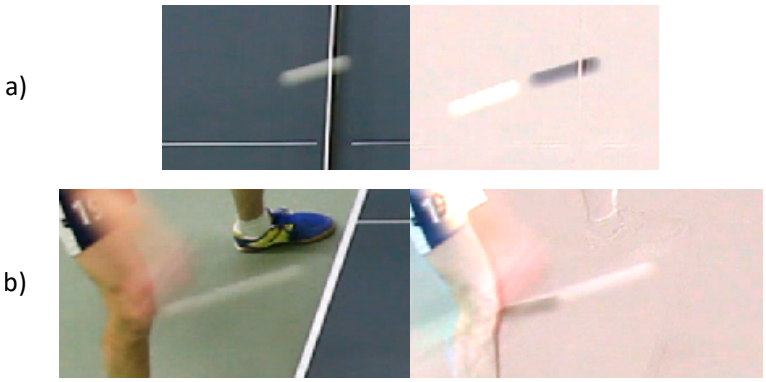


Figure 7.4: Failure modes.
Left images are frames from a video sequence, right images are difference images.
In a), white object is travelling at high speed over white line which is in the background. This results in one blurred stroke being separated into two connected components.

In b), two moving objects are touching each other. This will result in one large connected component which cannot be handled by current motion predictor or KNN approach.

7.5.1 Frames do not change

For unknown reasons, during a compression of a video a strange phenomenon can occur – same frame is rendered twice. This is not noticeable when watching a video, but can be seen when going frame by frame. When this happens there is no motion what so ever and our constant motion prediction fails. Even the difference image is blank and we cannot detect any blurred strokes with the KNN approach.

7.5.2 Two objects are touching

The drawbacks of using connected components is when two objects appear as one component because they are touching each other. This happens when a fast moving object is bouncing off some other moving object (see figure 7.4). Not only that the change of direction is not expected by our motion predictor, but also KNN approach will fail in this frame, because we are not able to match such connected components. This problem could be solved by different method for computing connected components, like by color similarity.

7.5.3 Blurred stroke is split by a line

When a fast moving object travels across a background with same color (ping pong ball across white wall) it is practically undetectable because the final color is same as background. But such background does not have to cover the whole trajectory (see figure 7.4) and we will only see parts of blurred stroke. This is also problem for connected components, because one blurred stroke will be split into two separate components.

7.5.4 Shadow

When fast moving objects approach other surface, they may cast a shadow which will appear together with them in one connected component (see figure 7.3). Since we are estimating blurred stroke's trajectory using boundaries of connected components, the extra surface area belonging to the shadow will cause problems and lead to incorrect estimation of trajectory.

Figure 7.5: Successful detections.
The detected object is shown in green, its position in previous frame in red, its trajectory is in blue and the prediction in yellow. From top to bottom, left to right: *tennis_serve_side*, *tennis_1*, *ping_pong_top*, *ping_pong_side*, *tennis_serve_back*, *squash*, *hockey*, *tennis_2*, tennis_serve_side. Best seen when zoomed at in attached pdf.

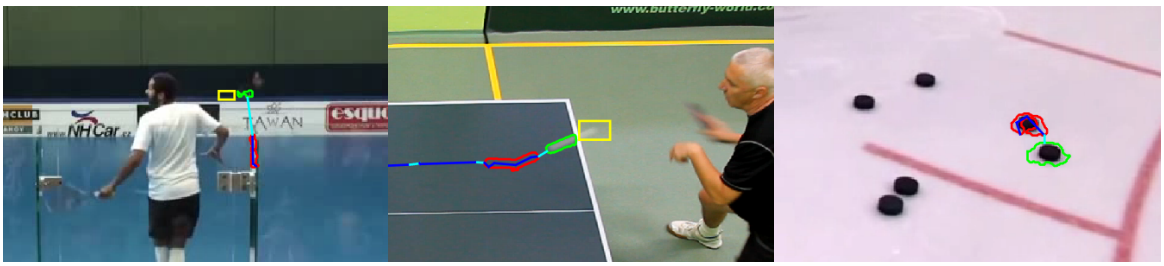


Figure 7.6: False detections.
The detected object is shown in green, its position in previous frame in red, its trajectory is in blue and the prediction in yellow. Best seen when zoomed at in attached pdf.

In the **first image** the detection is wrong due to noise. In the **second image**, trajectory is splitted into two connected components (discussed in 7.5.3). In the **third image** multiple instances of the same object are visible. Due to small camera movement they appear in foreground and are detected as blurred stroke.

Chapter 8

Conclusion

We have formulated a new problem – tracking by detection of very fast moving objects. These objects are captured with significant motion blur, which makes widely used template matching approaches difficult. Instead we show that there is useful information about object dynamics encoded in the blur, which makes tracking easier.

A new method for detection of such objects in a video sequence has been proposed. The method requires blurred stroke to appear in several consecutive frames in order to be detected by utilizing background subtraction, k-nearest neighbours matching and motion prediction. It has been shown that this method performs well when the blurred strokes are long and follow ballistic trajectory.

In order to evaluate the performance of our method we have collected a dataset of video sequences, covering many possible scenarios like a sudden change of movement, reflections, noise or scale changes. Blurred strokes appear in different orientations, lengths or colors. The dataset was annotated using segmentation masks of background and blurred stroke.

In addition, we showed how trajectory can be estimated through a skeletonization process and how is this trajectory used in deconvolution to guess the true appearance of a fast moving object. The trajectory and the true object appearance are be used in temporal resolution problem to create artificial, more detailed, frames or in another words slow motion video of the blurred stroke.

Bibliography

[1] Amit Agrawal, Mohit Gupta, Ashok Veeraraghavan, and Srinivasa G Narasimhan. Optimal coded sampling for temporal super-resolution. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 599–606. IEEE, 2010.

[2] Shengyang Dai and Ying Wu. Motion from blur. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[3] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[4] Ondrej Drbohlav and Jiri Matas. Nonparametric methods for density estimation nearest neighbour classification. University Lecture, 2015. URL <`https://cw.fel.cvut.cz/wiki/_media/courses/a4b33rpz/pr_04_nonparametric_methods_knn_2015_10_23.pdf`>.

[5] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[6] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270. IEEE, 2011.

[7] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[8] Matej Kristan, Jiri Matas, Ales Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Cehovin. A novel performance evaluation methodology for single-target trackers. 2015.

[9] E. Maggio and A. Cavallaro. *Video Tracking: Theory and Practice*. Wiley, 2011. ISBN 9781119956860. URL <`https://books.google.cz/books?id=v8g6_N1E-tYC`>.

[10] Pina Marziliano, Frederic Dufaux, Stefan Winkler, and Touradj Ebrahimi. A no-reference perceptual blur metric. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–57. IEEE, 2002.

[11] *MATLAB version 8.5.0.197613 (R2015a)*. The Mathworks, Inc., Natick, Massachusetts, 2015. URL <http://www.mathworks.com/>. visited on 7. 5. 2016.

[12] Nikon. DSLR Camera Basics - Frame rate, . URL <http://imaging.nikon.com/lineup/dslr/basics/25/03.htm>. visited on 7. 5. 2016.

[13] Nikon. DSLR Camera Basics - Shutter speed, . URL <http://imaging.nikon.com/lineup/dslr/basics/04/03.htm>. visited on 7. 5. 2016.

[14] Stepan Obdrzalek, Gregorij Kurillo, Ferda Ofli, Ruzena Bajcsy, Edmund Seto, Holly Jimison, and Misha Pavel. Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. In *Engineering in medicine and biology society (EMBC), 2012 annual international conference of the IEEE*, pages 1188–1193. IEEE, 2012.

[15] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 20(3):21–36, 2003.

[16] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. *ACM Transactions on Graphics (TOG)*, 25(3): 795–804, 2006.

[17] RED. Shutter angles & creative control. URL <http://www.red.com/learn/red-101/shutter-angle-tutorial>. visited on 10. 5. 2016.

[18] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[19] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 611–622. ACM, 2004.

[20] Shamik Tiwari, VP Shukla, SR Biradar, and AK Singh. Review of motion blur estimation techniques. *Journal of Image and Graphics*, 1(4):176–184, 2013.

[21] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[22] Eric W. Weisstein. Convolution. From MathWorld—A Wolfram Web Resource, 2016. URL <http://mathworld.wolfram.com/Convolution.html>. visited on 7. 5. 2016.

Appendix A

Abbreviations

2D	two dimensional
BSD	blurred stroke detector
FN	false negative
FP	false positive
FPS	frames per second
KNN	k-nearest neighbours
MATLAB	matrix laboratory
PSF	points spread function
SR	super resolution
TN	true negative
TP	true positive

Appendix B

Youtube videos

1. Sequence "tennis_1" and "tennis_2" were obtained from
   channel: "Tennis Production",
   video name: "Tennis Best Points Ever (Part 2) HD".
   Start time: 0:00, full link: <https://youtu.be/uy1ULXjkM-E>.

2. Sequence "squash" was obtained from
   channel: "Pavel Klimunda",
   video name: "MČR Squash 2012 - Petr Martin vs. Daniel Mekbib 3:2".
   Start time: 8:42, full link: <https://youtu.be/OcYC4bjElZs?t=8m42s>.

3. Sequence "hockey" was obtained from
   channel: "How To Hockey - Coach Jeremy",
   video name: "How To Take a Snapshot - On Ice Lesson - Howtohockey.com".
   Start time: 5:05, full link: <https://youtu.be/lsxYCuu-DUY?t=5m5s>.

4. Sequence "william_tell" was obtained from
   channel: "Dude Perfect"
   . video name: "Archery Trick Shots | Dude Perfect".
   Start time: 0:58, full link: <https://youtu.be/eCtb_ylVDvU?t=58s>.

Appendix C

CD content

Directory	Content
thesis	thesis LaTeXsource code
thesis/figures	figures in higher resolution
code	MATLAB source codes
input	used video sequences and their ground truth as a binary masks
output	binary masks of detected blurred strokes in provided video sequences

Table C.1: CD folder structure.