FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Semisupervised segmentation of UHD video
Student:	Bc. Oliver Keruľ-Kmec
Supervisor:	doc. Ing. RNDr. Martin Holeňa, CSc.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2018/19

Instructions

One of the advanced video preprocessing tasks is object segmentation with moving camera.
The students' task is to design and implement object segmentation via semisupervised classification of interest points.

1) Get familiar with image processing using points of interest.
2) Get familiar with methods of semisupervised classification.
3) Making use of available implementations, design and implement an acquisition of suitable time sequences from the video.
4) Making use of available implementations, implement a user interface utilising Boykov-Kolmogorov for selection of objects for segmentation.
5) Incorporate into the implemented approach at least two methods of semi-supervised classification on given data.
6) Compare the suitability of all variants of the implemented approach with traditionally used Gaussian mixtures.

References

Will be provided by the supervisor.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 29, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Semi-Supervised Segmentation of UHD Video

Bc. Oliver Kerul'-Kmec

Department of Theoretical Computer Science
Supervisor: doc. Ing. RNDr. Martin Holeňa, CSc.

January 9, 2019

Acknowledgements

I would like to express my sincere gratitude to Martin Holeňa for the help provided with this thesis, his supervising and friendly approach. Equally, I would like to thank Petr Pulc for his help with the implementation of the video processing methods and for providing me with testing videos.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on January 9, 2019 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Kerul'-Kmec, Oliver. *Semi-Supervised Segmentation of UHD Video.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Jednou z hlavných predspracujúcich úloh v oblasti získavania informácii z videa je segmentácia scény, hlavne segmentácia popredných objektov od pozadia. Ide vlastne o klasifikačnú úlohu, ktorá je špecifická v tom, že je časovo náročné získať človekom anotované trénovacie dáta na učenie klasifikátora. Preto je vhodné použiť semi-supervizovanú klasifikáciu. Táto práca sa zaoberá použitím semi-supervizovaných klasifikátorov založených na regularizovaní zhlukov a na fuzzy c-means v spojeni s úlohou segmentácie popredia a pozadia. Na klasifikáciu ďalších snímkov podľa jedného snímku, ktorý oštítkoval človek, je použitý detektor významných bodov založený na kombinácii detektoru rohov s vizuálny deskriptorom. Práca experimentálne porovnáva obe tieto metódy s tradičnou metódou GMM.

Kľúčové slová UHD video, Segmentácia scény, Detektor významných bodov, Semisupervizovaná klasifikácia, Regularizácia klastrov, C-means

Abstract

One of the key preprocessing tasks in information retrieval from a video is the segmentation of the scene, primarily its segmentation into foreground objects and the background. This is actually a classification task, but with the specific property that it is very time consuming and costly to obtain human-labelled training data for classifier training. That suggests to use semi-supervised classifiers to this end. The presented work reports the investigation of semi-supervised classification methods based on cluster regularization and on fuzzy c-means in connection with the foreground / background segmentation task. To classify as many video frames as possible using only a single human-based frame, the semi-supervised classification is combined with a frequently used keypoint detector based on a combination of a corner detection method with a visual descriptor method. The paper experimentally compares both methods of semi-supervised classification in this context with traditional algorithm GMM.

Keywords UHD video, Scene segmentation, Keypoint detector, ORB, Semi-supervised classification, Cluster regularization, C-means

Contents

List of Figures

List of Tables

Introduction

For the indexing of multimedial content, it is beneficial to extract annotations of actors, objects that occur in the video or any other semantic information that can occur in a video. A vital preprocessing task to prepare such annotations is the segmentation of the scene into foreground objects and the background. Traditional algorithms work on the pixel level and are time consuming for higher resolution. That is why we decided to choose a different approach.

The method proposed in this master's thesis is based on the semi-supervised classification of features detected by the ORB (Oriented FAST and Rotated BRIEF) algorithm. The approach is based on the assumption, that the object and the background move in different directions when the camera moves. From the key points obtained in two succeeding frames, we gather a set of motion vectors. Those motion vectors are subsequently classified by a semi-supervised classifier to discriminate the individual objects in the scene and background. To get initial labels, a user is involved to annotate the object(s) of interest and background with rather imprecise scribbles in one of the first frames. Those labels serve as supervised information in the classification of the whole video.

In the analysis part of the thesis – chapters 2 and 3 – we introduce all methods and algorithms used in our proposed approach. They include the already mentioned ORB algorithm and a method for feature tracking built on the ORB detector. For expansion of the initial user annotation to all pixels in the frame, we use the Boykov-Kolmogorov algorithm.

In chapters 4 and 5 we introduce our proposed approach and validate it on a set of videos created for this experiment.

Chapter 1

State-of-the-art

Computer vision is a field of computer science that deals with processing of the digital images and retrieving information from them. A video is a sequence of images called frames. A video is usually processed on the frame level, so the methods used are similar to those used for images.

In this chapter, we state the problem solved in our thesis, show our related research and present one of the traditional algorithms for image segmentation – the Gaussian Mixture Model.

1.1 Problem

There are three main tasks related to objects in computer vision: classification, detection, and segmentation. Classification aims to answer the question "what is in the picture". The other tasks try to also answer the question "where is the object in the picture". [1] Segmentation specifically aims to provide such information at the pixel level.

More precisely, scene segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). Each segment represents one of the objects in the foreground. The remaining part of the image is considered the background.

1.2 Our Previous Research

In [6], we have tested a similar approach. Instead of an initial annotation by a user, the initial labels were taken from the ground truth. We were testing the robustness of the solution with respect to the dependence on the order of the frame when the classifier was trained. The test included the first five frames of the video.

The hypothesis of equality of all five classifiers was rejected (p-value $< 5\%$) only for the delay 1 frame and the F-measure, and weakly rejected (p-value

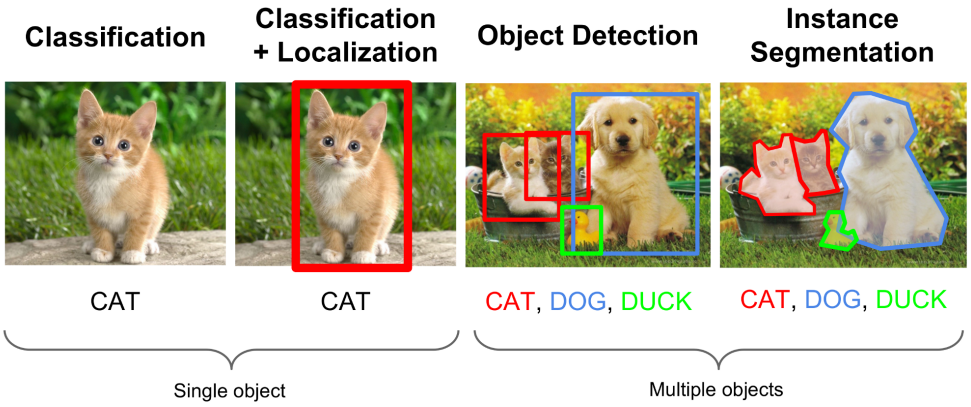

Figure 1.1: The difference between classification, detection and segmentation [1]

Table 1.1: Results of the Friedman test of the hypothesis that for a given delay between classifier training and measuring its quality, a given quality measure is equal for the classifiers trained in each of the 5 first video frames. The combinations for which the tested hypothesis was weakly rejected (p-value $< 10\%$) are in italic, the single combination for which it was rejected (p-value $< 5\%$) is in bold italic.

Quality measure	Delay	p-Value
accuracy	1	1
accuracy	5	0.117
accuracy	10	1
sensitivity	1	*0.052*
sensitivity	5	0.428
sensitivity	10	0.238
specificity	1	1
specificity	5	1
specificity	10	0.25
F-measure	1	***0.043***
F-measure	5	*0.089*
F-measure	10	0.238

$< 10\%$) for the delay 1 frame and the sensitivity, as well as for the delay 5 frames and the F-measure. All p-values from the Friedman test are shown in Table 1.1.

The accuracy and specificity of one video for all classifiers in 50 frames are illustrated in Figure 1.2.

1.3 Gaussian Mixtures Model

One of the traditional algorithms for the foreground segmentation is Gaussian Mixtures Model (GMM). It is used to model the background and the foreground objects of the image by a combination of several Gaussians. The means of those Gaussians serve as a threshold value for distinguishing between the background and the foreground objects.

Each area which comes into the view of a given pixel is represented by one of a set of states $k \in \{1, 2, \ldots, K\}$. Some of the states correspond to background objects and the rest are deemed to be foreground. The process k which generates the state at each frame time is simply modeled by a set of K parameters ω_k . Each represents the a priori probability of an area k appearing in the pixel view, hence $\sum_{k=1}^{K} \omega_k = 1$.

The pixel values are samples of some random process $\mathbf{X}$ which includes the behaviour of k. The process $\mathbf{X}$ is assumed to be modelled by a mixture of K Gaussian densities with parameter sets θ_k, one for each state k:

$$f_{\mathbf{X}|k}(X|k, \theta_k) = \frac{1}{(2\pi)^{\frac{n}{2}} |\sum_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1}(X-\mu_k)} \tag{1.1}$$

where μ_k is the mean and Σ_k is the covariance matrix of the kth density. It is assumed that the components of $\mathbf{X}$ are independent so that Σ_k is diagonal and may be represented by the n-dimensional variance σ_k^2. The density parameter set is defined as $\theta_k = \{\mu_k, \sigma_k\}$ for a given k and the total set of parameters becomes $\Phi = \{\omega_1, \ldots, \omega_k, \theta_1, \ldots, \theta_k\}$.

Because the events k are disjoint, the distribution of $\mathbf{X}$ may be modelled as a sum-of-Gaussians mixture (see Figure 1.3).

$$f_{\mathbf{X}}(X|\Phi) = \sum_{k=1}^{K} P(k) f_{\mathbf{X}|k}(X|k, \theta_k) \tag{1.2}$$

where $P(k) = \omega_k$. All parameters need to be estimated from observations of $\mathbf{X}$ in parallel with the estimation of the hidden state k.

Maximum likelihood parameter estimation from incomplete data is solved with an approximate formulation of the expectation-maximization (EM) algorithm. [7] [8] The EM algorithm works by iterating two steps:

- E-step – finding an estimate based on the hidden data of the likelihood function of the complete data using the observed data and current estimates of the parameters
- M-step – calculating maximum likelihood estimates of the parameters using the observed data and current estimates of the hidden data

The first step is to estimate which of the K distributions most likely gave rise to the current sample $\mathbf{X} = X$. The posterior probability $P(k|X, \Phi)$ is

the likelihood that this pixel value was generated in state k, given by Bayes's theorem (see Figure 1.4):

$$P(k|X,\Phi) = \frac{P(k)f_{\mathbf{X}|k}(X|k,\theta_k)}{f_{\mathbf{X}}(X|\theta)} \tag{1.3}$$

The k which maximizes $P(k|X,\Phi)$ is the maximum a posteriori estimate

$$\hat{k} = \arg\max_k P(k|X,\Phi) = \arg\max_k \omega_k f_{\mathbf{X}|k}(X|k,\theta_k) \tag{1.4}$$

The mixture model models both foreground and background surfaces without distinction. This is why a total of $K = 3$ Gaussians may be considered a practical minimum to model two background surfaces and one foreground surface in each pixel. It is generally considered that not much improvement is obtained beyond $K = 5$ distributions [2].

The Stauffer-Grimson procedure [7] for demarcation starts by ranking the K states by the criterion ω_k/σ_k. If σ_k is not a scalar the ranking has to be done with $\omega_k/||\sigma_k||$ or $\omega_k^2/||\sigma_k||^2$.

A surface is deemed to be the background with a higher probability if it occurs frequently (high ω_k) and does not vary much (low σ_k). To demarcate the background, they provide an overall prior probability T of anything in view being background. The first B of the ranked states whose accumulated probability accounts for T are deemed to be background,

$$B = \arg\min b(\sum_{k=1}^{b} \omega_k > T) \tag{1.5}$$

and the rest of the states are foreground.

The complete-data likelihood function is given by

$$P(X_1, X_2, \ldots, X_N, k|\Phi) = \sum_{t=1}^{N} \omega_k f_{\mathbf{X}|k}(X_t|k,\theta_k) \tag{1.6}$$

where X_t shows the pixel value at time t. Renewed estimates of the paramteres Φ are obtained by maximizing the expected value of 1.6 with respect to k. The results are:

$$\hat{\omega_k} = \frac{1}{N}\sum_{t=1}^{N} P(k|X_t,\Phi) \tag{1.7}$$

$$\hat{\mu_k} = \frac{\sum_{t=1}^{N} X_t P(k|X_t,\Phi)}{\sum_{t=1}^{N} P(k|X_t,\Phi)} \tag{1.8}$$

$$\sigma_k^2 = \frac{\sum_{t=1}^{N}((X_t - \hat{\mu_k}) \circ (X_t - \hat{\mu_k}))P(k|X_t, \Phi)}{\sum_{t=1}^{N} P(k|X_t, \Phi)} \tag{1.9}$$

where $\circ$ is the element-wise multiplication operator and $P(k|X_t, \Phi)$ is given in each case by 1.3.

Equations 1.7 to 1.9 assume stationary processes $\mathbf{k}$ and $\mathbf{X}$, and also a fixed number of observations N. A practical implementation that is capable of foreground segmentation of each frame as it is acquired has to re-estimate the current surface k and all the parameters incrementally from each new sample $\mathbf{X} = X_t$ as well as adapt to changing scene statistics.

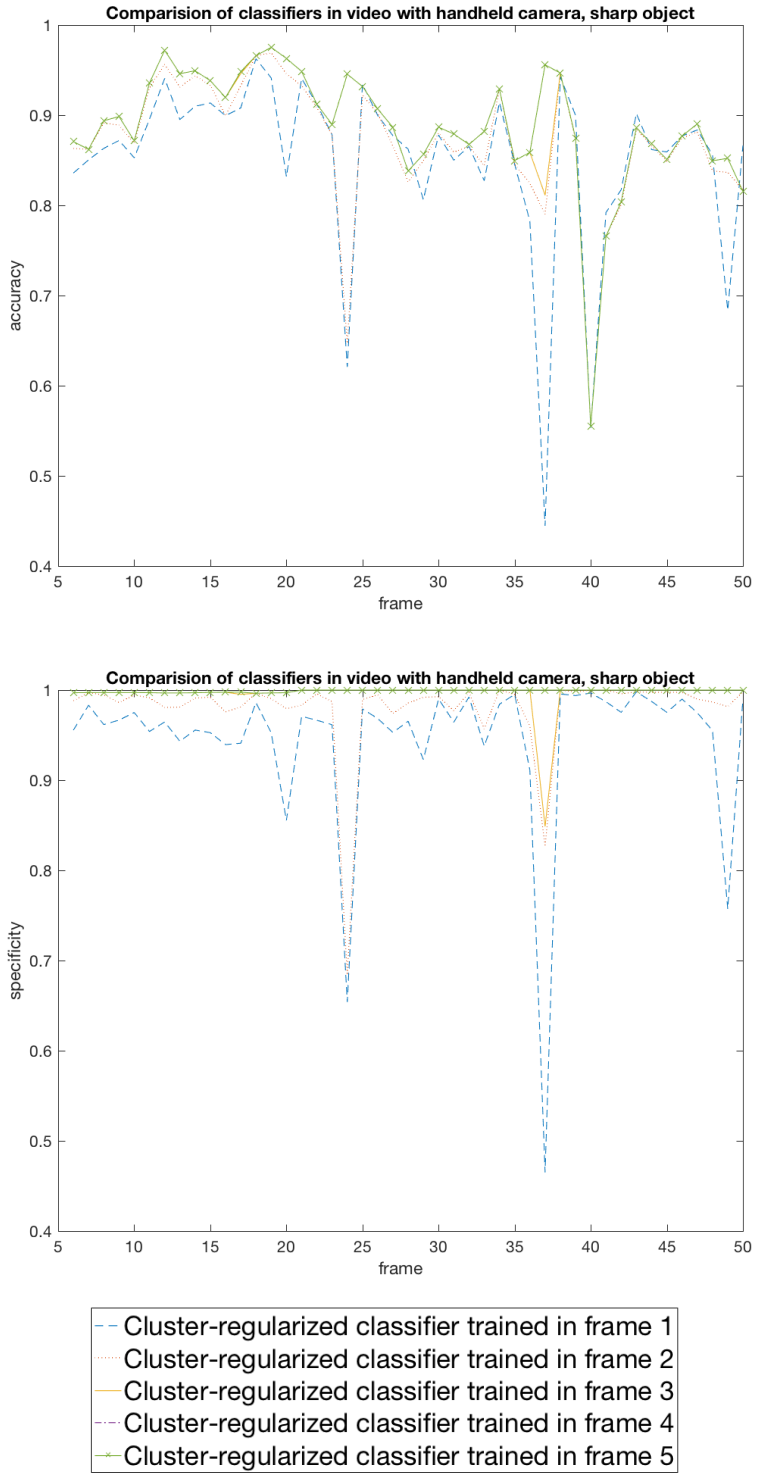

Figure 1.2: The evolution of accuracy (top) and specificity (bottom) of the classifiers trained in each of the 5 first video frames for a handheld-camera video with only the foreground object sharp

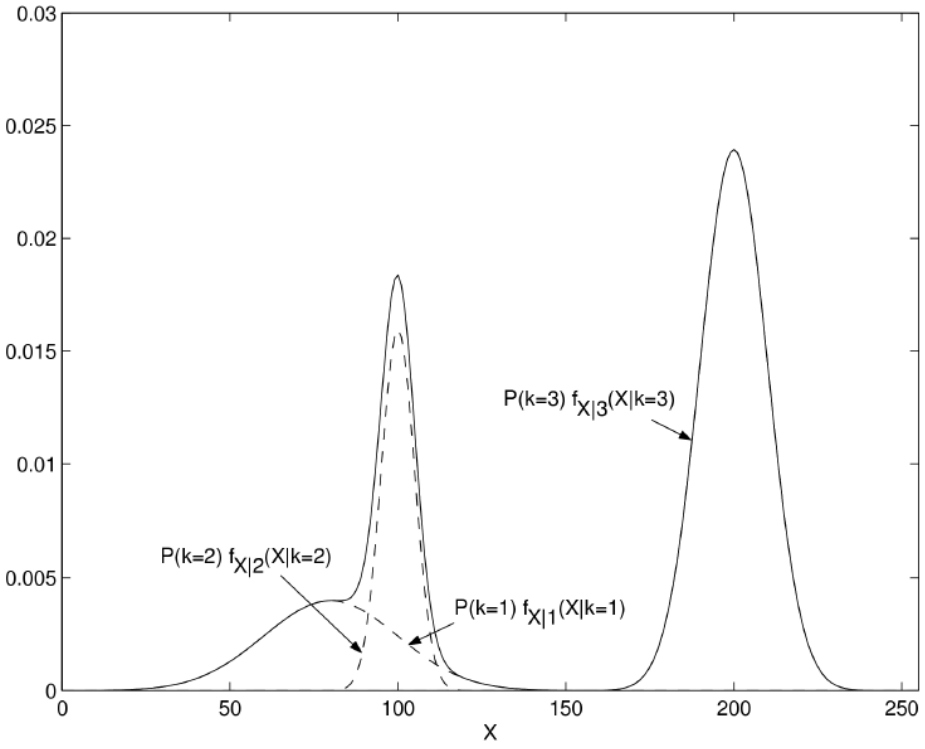

Figure 1.3: The pixel value probability $f_{\mathbf{X}}(X|\Phi)$ for 1D pixel values $X \in \{0, 1, \ldots, 255\}, K = 3, \omega_k = \{0.2, 0.2, 0.6\}, \mu_k = \{80, 100, 200\}$, and $\sigma_k = \{20, 5, 10\}$ [2]

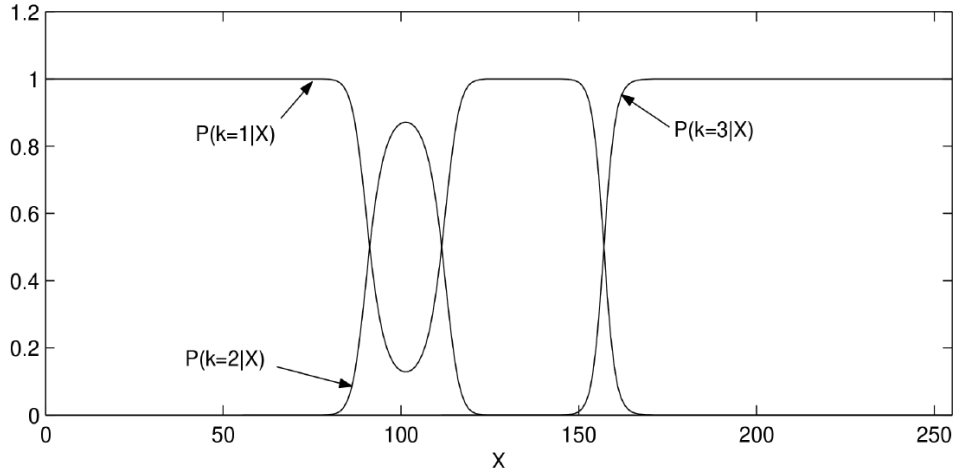

Figure 1.4: The a posteriori probabilities $P(k|X;\Phi)$ plotted as functions of X for each $k = 1, 2, 3$, using the same parameters as in Figure 1.3 [2]

Chapter 2

Methods of Video Processing

In this chapter, methods and algorithms for video processing are introduced. First, we introduce the area of feature detection generally and then one of the algorithms in detail – ORB. Subsequently, we introduce an algorithm that is built on the ORB detector. Finally, we recall the algorithm for the min-cut/max-flow problem in the graph theory, which can be also used in the image segmentation area.

2.1 Feature detection

Feature detection is a very important area of computer vision and image processing. It includes methods for making local decisions at every image point depending on the presence or value of a particular feature. An important characteristic of algorithms for feature detection is reproducibility. That means that the same feature will be detected in different images with the same scene or object.

The features are a starting point for lots of algorithms in image processing - for example, object tracking, face detection, image classification, motion detection and tracking and so on. The quality of those algorithms is dependent on the quality of feature detector.

Types of features:

- **Edges** – points where there is a boundary between two image regions, for example, object and background
- **Corners** – points where the direction of the edge changes rapidly or there is a high level of curvature in the image gradient
- **Blobs** – regions of interest points; blob detectors can detect areas which are too smooth to be detected by a corner detector

List of important algorithms for feature detection:

- Canny – edges
- SUSAN – edges, corners
- SIFT – corners
- SURF – corners
- FAST – corners, blobs
- Harris corner detection – corners
- ORB – corners

2.1.1 ORB (Oriented FAST and Rotated BRIEF)

This method for feature detection, in detail described in [3], is a combination of a corner detection method FAST (Features from Accelerated Segment Test) with a visual descriptor method BRIEF (Binary Robust Independent Elementary Features). It was demonstrated that ORB is two orders of magnitude faster than SIFT (Scale-Invariant Feature Transform) while performing as well in many situations.[3] It is also less affected by image noise and is capable of being used for real-time performance.

The FAST detector takes one parameter – the intensity threshold between the center pixel and those in a circular ring about the center. The FAST does not produce a measure of cornerness, and it was found that it has large responses along edges. A Harris corner measure is employed to order the FAST key points. To get N key points, the threshold is set low enough to get more than N of them and then they are ordered according to the Harris measure to pick the top N points.

As a measure of corner orientation, *the intensity centroid* is used. It assumes that a corner's intensity is offset from its center, and this vector may be used to impute an orientation. In [9], Rosin defines the moments of a patch as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y), \tag{2.1}$$

where $I(x,y)$ is image intensity and using these moments, we may find the centroid:

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \tag{2.2}$$

Then, we can construct a vector from the corner's center, O, to the centroid. The orientation of the patch is then simply:

$$\theta = atan2(m_{01}, m_{10}), \tag{2.3}$$

where atan2 is the quadrant-aware version of arctan.

BRIEF descriptor [10] is a bit-string description of an image patch constructed from a set of binary intensity tests. Consider a smoothed image patch, p. A binary test τ is defined by:

$$\tau(p; x, y) = \begin{cases} 0, & p(x) < p(y) \\ 1, & p(x) \geq p(y) \end{cases} \tag{2.4}$$

where $p(x)$ is the intensity of p at a point x. A detected feature is defined as a vector of n binary tests:

$$f_n(p) = \sum_{1 \leq i \leq n} 1^{i-1} \tau(p; x_i, y_i) \tag{2.5}$$

The length n of that vector is selected as 256. Before performing the tests, smoothing is achieved using an integral image, where each test point is a 5×5 subwindow of a 31×31 pixel patch.

The matching performance of BRIEF falls off for in-plane rotation of more than a few degrees (see Figure 2.1). An efficient method to allow BRIEF to be invariant to in-plane rotation, is to steer BRIEF according to the orientation of keypoints. For any feature set of n binary tests at location (x_i, y_i), define the $2 \times n$ matrix

$$S = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix} \tag{2.6}$$

Using the patch orientation θ and the corresponding rotation matrix R_θ, we construct a steered version S_θ of S:

$$S_\theta = R_\theta S, \tag{2.7}$$

and the steered BRIEF operator comes

$$g_n(P, \theta) = f_n(p) | (x_i, y_i) \in S_\theta \tag{2.8}$$

BRIEF has an important property that each bit feature has a large variance and a mean near 0.5. This mean gives the maximum sample variance 0.25 for a bit feature. But when it is oriented along keypoint direction, it loses this property and becomes more distributed.

High variance makes a feature more discriminative since it responds differently to inputs. Another desirable property is to have these tests uncorrelated, since then each test will contribute to the result independently.

To recover from the loss of variance in steered BRIEF, and to reduce correlation among binary tests, a learning method was developed for choosing a good subset of binary tests.

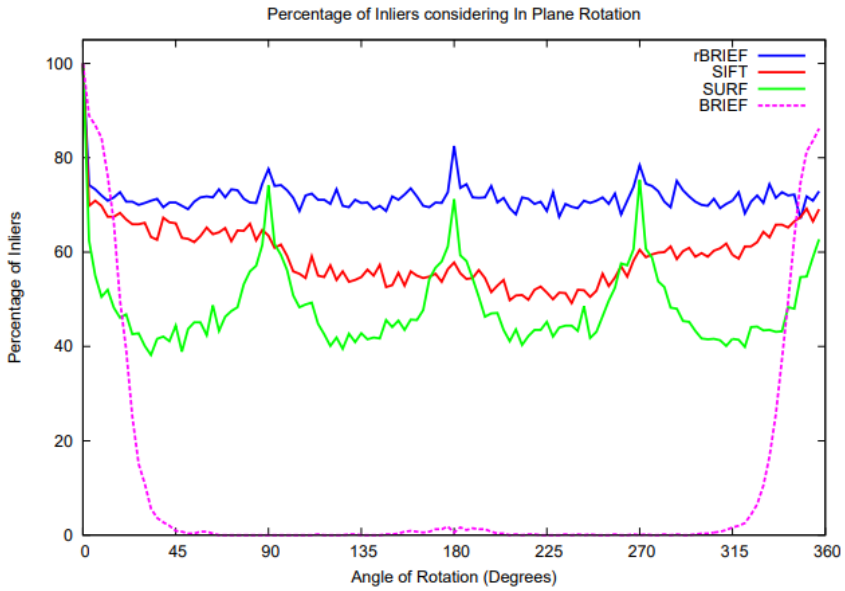

Figure 2.1: Matching performance of SIFT, SURF, BRIEF with FAST, and rBRIEF under synthetic rotations with Gaussian noise of 10 [3]

It searches among all possible binary tests to find ones that both have high variance and are uncorrelated. The method is called rBRIEF. [3] It has a significant improvement in the variance and correlation over steered BRIEF.

Figure 2.2 shows a correlation between the speed and the accuracy of kd-trees with SIFT and LSH with rBRIEF. LSH is faster than the kd-trees, most likely thanks to its simplicity and the speed of the distance computation.

2.2 Features tracking

The following method for features tracking, proposed in [11], is based on the ORB feature detector. In each frame of the video, a key point detector is used to detect points of interest and compute their descriptors. Points of interest detected in a frame are always attempted to match those detected in the next frame.

For a match between points of interest p_k and p_{k+1} in subsequent frames k and $k+1$, the following criteria have been used:

(i) Only the points of interest in the spacial neighbourhood of the expected position are considered. That position is based on last known interest point position and its past motion (if available).

The point p_{k+1} must lie within the radius r_k^p from the estimated new

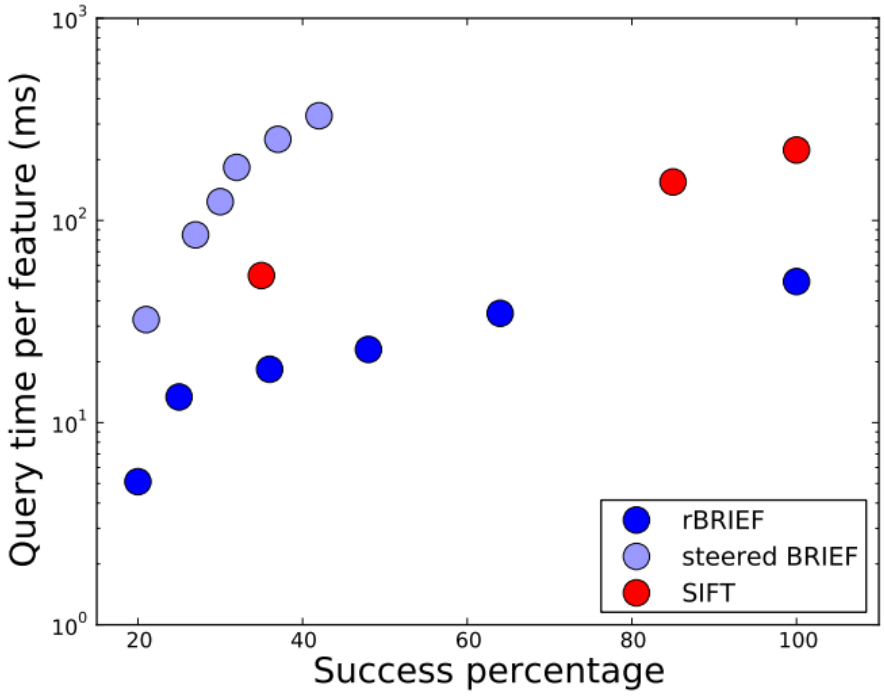

Figure 2.2: Example of features detected by OpenCV implementation of ORB [4]

position of the point $\hat{p}_k$

$$\|p_{k+1} - \hat{p}_k\| < r_k^p. \tag{2.9}$$

Here, the estimated position $\hat{p}_k$ is calculated as

$$\hat{p}_k = \begin{cases} p_k + c_1(p_k - p_{k-1}) & \text{if } p_{k-1} \text{ is available,} \\ p_k & \text{else,} \end{cases} \tag{2.10}$$

with $c_1 > 0$, and the radius r_k^p is calculated as

$$r_k^p = (u_k^p W)^2, \tag{2.11}$$

where u_k^p quantifies the uncertainty pertaining to the point p_k in the k-th frame and W denotes the frame width (in the units in which point positions are expressed). The uncertainty u^p is set to $u_1^p = c_2 > 0$ in the first frame and is then evolved from frame to frame through linear scaling above a lower limit $c_3 > 0$:

$$u_{k+1}^p = \begin{cases} \max(c_3, c_4 u_k^p) & \text{if } p_k \text{ is matched,} \\ c_5 u_k^p & \text{if } p_k \text{ is not matched,} \end{cases} \tag{2.12}$$

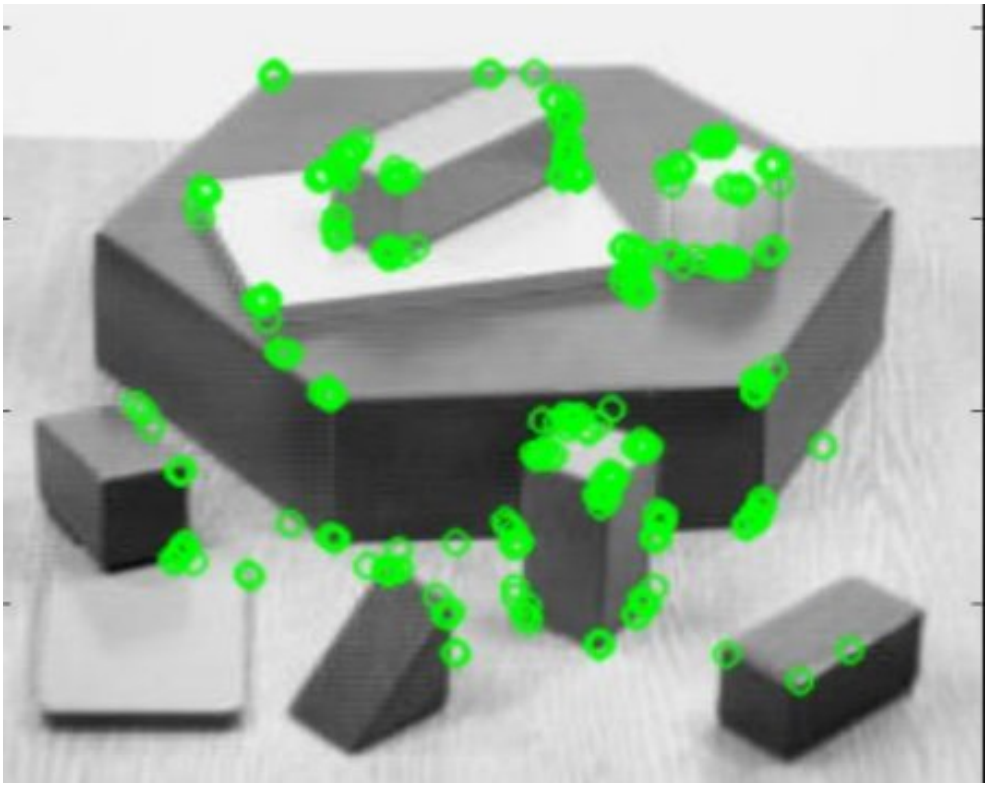

Figure 2.3: Speed vs. accuracy. The descriptors are tested on warped versions of the images they were trained on. We used 1, 2 and 3 kd-trees for SIFT (the autotuned FLANN kd-tree gave worse performance), 4 to 20 hash tables for rBRIEF and 16 to 40 tables for steered BRIEF (both with a sub-signature of 16 bits). Nearest neighbours were searched over 1.6M entries for SIFT and 1.8M entries for rBRIEF. [3]

where $0 < c_4 < 1, c_5 > 1$.
Moreover, if the evolution (2.12) leads to $u_{k+1}^p > c_6$ for some $c_6 > c_3$, then the point p is deactivated and not any more considered for matching.

(ii) Among the points of interest resulting from (i), as well as among all detected in the current frame for which no information about their past motion is available, points in the previous frame are searched based on the Hamming distance between the descriptors of both points. Hamming distance between the 256-bit binary descriptors of the points is allowed to be at most 64.

Whereas the dependence of matching success on the difference between positions of the points and on the movement of the first point has a straightforward geometric meaning, its dependence on the Hamming distance between their

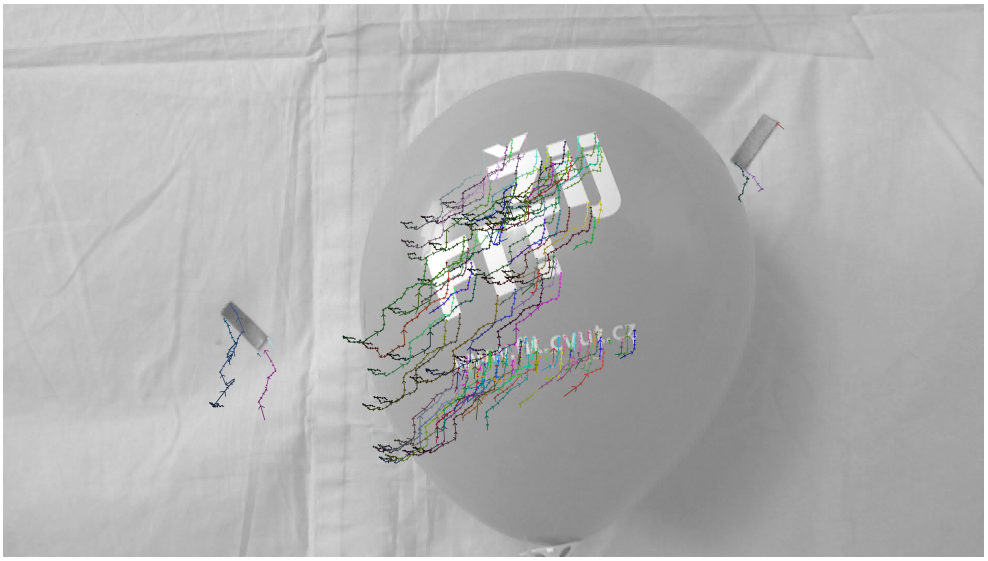

Figure 2.4: Example of the motion description of each point of interest. The darker line is, the older information it represents

descriptors has a probabilistic character. In [3], this dependence was investigated and was found that if the Hamming distance between 256-bit binary descriptors of the points is greater than 64, then the probability of successful match is less than 5%.

The choice of the real-valued constants in the criterion (i) has been based on the resolution of the video (4K), on the frame rate (25) and on the defaults in the ORB implementation based on [3]. They have been set to the following values: $c_1 = 0.6, c_2 = 0.02, c_3 = 0.009, c_4 = 0.9, c_5 = 1.1, c_6 = 0.03$.

If two points of interest in subsequent frames are considered matching, the point in the later frame is added to the history vector of the point in the previous frame. In this way, we get the motion description of each point of interest. An example of the motion description is in Figure 2.4.

2.3 Boykov-Kolmogorov max-flow

The min-cut/max-flow algorithm, proposed in [5], belongs to the group of algorithms based on augmenting paths. The difference compared to other similar algorithms is that it reuses trees and never starts building them from scratch.

Figure 2.5 shows the basic terminology. There are two non-overlapping search trees S and T. The nodes that are not in S or T are called "free". The nodes in the search trees can be either "active" or "passive". The active nodes represent the outer border in each tree while the passive nodes are internal.

Active nodes allow trees to grow by acquiring new children from a set of free nodes. An augmenting path is found as soon as an active node in one of the trees detects a neighbouring node that belongs to the other tree.

The algorithm iteratively repeats the following three stages:

(i) growth stage – search trees S and T grow until they touch giving a $s \to t$ path

(ii) augmentation stage – the found path is augmented, search trees can fall apart and change into forests

(iii) adoption stage – trees S and T are restored

At the growth stage, the search trees expand. The active nodes explore adjacent non-saturated edges and acquire new children from a set of free nodes. The new nodes become active. As soon as all neighbours of a given active node are explored the active node becomes passive. The growth stage terminates if an active node of one tree touches a node from another tree.

The augmentation stage augments the path found at the growth stage. First, it finds the bottleneck capacity of the found path, and then it updates the residual capacity of the edges from this path by substracting the bottleneck capacity from the residual capacity. This phase can destroy the built-up search trees.

The goal of the last stage is to restore the single-tree structure of sets S and T with roots in the source and the sink. A new valid parent is tried to be found for each orphan. If there is no qualifying parent, the orphan is removed from S or T and it is made a free node. The stage terminates when no orphans are left and the search tree structures are restored.

After the adoption stage is completed the algorithm returns to the growth stage. The algorithm terminates when the search trees cannot grow and the trees are separated by saturated edges. This implies that a maximum flow is achieved. The corresponding cut can be determined by the search trees.

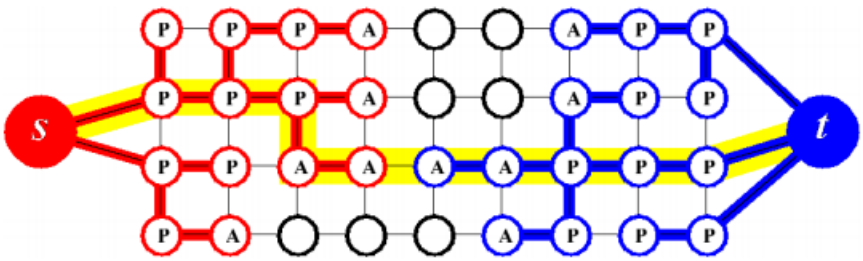

Figure 2.5: Example of the search trees S (red nodes) and T (blue nodes) at the end of the growth stage when a path (yellow line) from the source s to the sink t is found. Active and passive nodes are labeled by letters A and P, correspondingly. Free nodes appear in black. [5]

Chapter 3

Classification

3.1 Definition

Classification is one of the most common problems in the area of machine learning [12]. Its task is to assign labels to new observations based on a training set of observations. In contrast to clustering, where we do not know the classes to assign, in case of classification, we know the labels for the observations in the training set. Observations from the set $\mathscr{X} = \mathscr{F}_1 \times \mathscr{F}_2 \times \cdots \times \mathscr{F}_d$ are defined by a set of features. The features can be categorical (e.g. blood types "A", "B", "AB" or "0'), ordinal (e.g. "small", "medium" or "large"), integer or real numbers. Mathematically, classification can be viewed as a function $\hat{c} : \mathscr{X} \to \mathscr{L}$, where $\mathscr{L} = \{C_1, C_2, \ldots, C_k\}$ is a finite set of classes. The algorithm that implements actual classification is called classifier.

3.1.1 Binary classificiation

The most elementary type of classification is binary classification when there are only two classes. Usually, one is considered positive and the other negative. A typical example of binary classification is spam filtering. In this case, we consider spam as the positive class (similarly to a disease) because we want to identify it.

3.1.2 Multiclass classification

In the real world, classification into multiple classes is more common. There are two approaches how to classify into multiple classes using binary classifier. Consider classification into k classes:

- *one-versus-rest* – k binary classifiers are used, where each of them is trained to separate one class from the union of all remaining ones

- *one-versus-one* – in this case, $k(k-1)/2$ binary classifiers are used, each of them is trained to distinguish a pair of classes

3.2 Evaluation of classifier's quality

3.2.1 Contingency table

For the evaluation of classifiers' quality, we can use a contingency table. The rows of the table represent real classes as they are labeled in the testing set and in the columns, there are classes that were predicted by the classifier. In Table 3.1, there are 50 positive samples, 30 of them were classified correctly and 20 of them were classified wrongly as negative.

In case of binary classification, correctly classified positive and negative records are called *true positives (TP)*, and *true negatives (TN)*, respectively. Wrongly classified positive are called *false negatives (FN)* and similarly incorrectly classified negative records are called *false positives (FP)*. These four values are placed in the contingency table. We will use this terminology in the evaluation of quality metrics.

3.2.2 Quality metrics

From the contingency table, it is possible to calculate these quality metrics:

Accuracy It is one of the easiest indicators. It is calculated as the rate of the number of correctly classified records to the size of the testing set:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Error rate It is the complement to the Accuracy:

$$\frac{FP + FN}{TP + FP + TN + FN}$$

Table 3.1: Contingency table serves for the evaluation of classifier's quality. In case of a binary classifier, the values on the decreasing diagonal show correct predictions, while the values on the increasing diagonal show classification errors.

Predicted \Real	$\oplus$	$\ominus$	
$\oplus$	**30**	10	40
$\ominus$	20	**40**	60
	50	50	100

Sensitivity (true positive rate) It expresses the accuracy for classification of positive class. It is calculated as the rate of the number of correctly classified positive records to the number of positive records:

$$\frac{TP}{TP + FN}$$

Specificity (true negative rate) It expresses the accuracy for classification of negative class. It is calculated as the rate of the number of correctly classified negative records to the number of negative records:

$$\frac{TN}{FP + TN}$$

False negative rate It expresses the error rate for the classification of positive class. It is calculated as the rate of the number of incorrectly classified positive records to the number of positive records:

$$\frac{FN}{TP + FN}$$

False positive rate It expresses the error rate for the classification of negative class. It is calculated as the rate of the number of incorrectly classified negative records to the number of negative records:

$$\frac{FP}{FP + TN}$$

Precision It expresses the probability that the record classified as positive is really positive. It is calculated as the rate of the number of correctly classified positive records to the number of records classified as positive:

$$\frac{TP}{TP + FP}$$

Accuracy can be deceiving when classes are not distributed uniformly. For example, we want to distinguish between unauthorized payments by bank card. 99 % of payments is authorized and only 1 % is unauthorized. Classifier, that would label all payments as authorized would achieve 99 % accuracy. But it wouldn't be very useful. That is why it is better to calculate precision rather than accuracy in these cases.

F-measure It is a compromise between sensitivity and precision. It is calculated as the harmonic mean of those values:

$$\frac{2TP}{2TP + FP + FN}$$

For multiclass classification, we can calculate only accuracy and its complement – error rate. We can also calculate accuracy and precision for each class.

3.3 Semi-Supervised Classification

In situations where the number of available labelled instances is insufficient and labelling is expensive and time consuming, semi-supervised classification can be employed, which uses both labelled and unlabelled instances for learning. Unlabelled data with a small amount of labeled data can bring improvement in the accuracy of a classifier. Semi-supervised classification can be either inductive or transductive. The goal of inductive learning is to find, by means of all available training data, a mapping of inputs to labels. The goal of transductive learning is to find labels for a specific subset of inputs based on a specific subset of training data.

Semi-supervised classifiers make use the following assumptions: [13]

(i) **Continuity assumption** – Points which are close to each other are more likely to share a label. It assumes that there exist simple geometrical boundaries. This assumption is present also in supervised learning.

(ii) **Cluster assumption** – The data in the same cluster share the same label. This assumption is present in unsupervised learning and makes it usable for semi-supervised learning.

(iii) **Manifold assumption** – "The (high-dimensional) data lie (roughly) on a low-dimensional manifold." [13]

In the reported research we used the following two methods for semi-supervised classification.

3.3.1 Semi-Supervised Classification with Cluster Regularization

The principle of this method, in detail described in [14], consists in clustering all labelled and unlabelled instances and estimating, for the instance x_k, $k = 1, \ldots, N$, its probability distribution q_k on the set of clusters. In addition, the following penalty function is proposed for the differences between the pairs (q_k, q_n) of probability distributions of the instances.

$$P(q_k, q_n) = \sin\left(\frac{\pi}{2}(r(q_k, q_n) * s(q_k, q_n))^{\kappa}\right), \quad k, n = 1, \ldots, N, k \neq n, \quad (3.1)$$

where $r(q_k, q_n)$ denotes the Pearson correlation coefficient between q_k and q_n, κ is a parameter controlling the steepeness of the mapping from similarity to penalty, and $s(q_k, q_n)$ is a normalized similarity of the probability distributions q_k and q_n, defined

$$s(q_k, q_n) = 1 - \frac{\|q_k - q_n\| - d_{\min}}{d_{\max} - d_{\min}} \quad (3.2)$$

using the notation

$$d_{\min} = \min Q,\ d_{\max} = \max Q,$$
$$\text{with } Q = \{\|q_k - q_n\| | k, n = 1, \ldots, N, k \neq n\}. \quad (3.3)$$

The results of clustering allow to assign pseudolabels to unlabelled instances. In particular, the pseudolabel assigned for the j-th among the M considered classes to an unlabelled instance x_n in a cluster Ψ is

$$\hat{y}_{n,j} = \frac{\exp\left(\sum_{x_k \in \Psi \text{ is labelled }} y_{k,j}\right)}{\sum_{i=1}^{M} \exp\left(\sum_{x_k \in \Psi \text{ is labelled }} y_{k,i}\right)}, \quad (3.4)$$

where $y_{k,i}, i = 1, \ldots, M$ is a crisp or fuzzy label of the labelled instance x_k for the class i. For uniformity of notation, the symbol $\hat{y}_{k,j}, j = 1, \ldots, M$ can also be used for $y_{k,j}$ if x_k is labelled.

The penalty function (3.1) can be used as a regularization modifier in some loss function $L : [0,1]^2 \to [0, +\infty)$ measuring the discrepancy between the classifier outputs $F(x_n) = ((F(x_n))_1, \ldots, (F(x_n))_M)$ for an instance x_n, and the corresponding labels $(y_{n,1}, \ldots, y_{n,M})$ or pseudolabels $(\hat{y}_{n,1}, \ldots, \hat{y}_{n,M})$:

$$E = \frac{1}{N} \sum_{j=1}^{M} \left(\sum_{x_n \text{ labelled}} L((F(x_n))_j, y_{n,j}) + \right.$$
$$\left. \sum_{x_n \text{ unlabelled}} \frac{\lambda \max(q_n)}{|\phi(x_n)|} \sum_{x_k \in \phi(x_n)} P(q_k, q_n) L((F(x_k))_j, \hat{y}_{k,j}) \right), \quad (3.5)$$

where $\lambda > 0$ is a given parameter determining the tradeoff between supervised loss and unsupervised regularization, and the set of instances $x_k \neq x_n$ with the highest value of $P(q_k, q_n)$ is denoted $\phi(x_n)$.

In [14], the following design decisions have been made for the loss function and the classifier in (3.5):

1. The employed loss function can be derived from $D_{\mathrm{KL}}\left((\hat{y}_{n,1}, \ldots, \hat{y}_{n,M}) \| F(x_n) \right)$, the Kullback-Leibler divergence, from classifier outputs to labels or pseudolabels. If both the labels or pseudolabels and the classifier outputs form probability distributions on classes, then

$$D_{\mathrm{KL}}((\hat{y}_{n,1}, \ldots, \hat{y}_{n,M}) \| F(x_n)) =$$
$$= \sum_{j=1}^{M} \hat{y}_{n,j} \ln \left(\frac{(F(x_n))_j}{\hat{y}_{n,j}} \right), n = 1, \ldots, N. \quad (3.6)$$

Therefore, the considered loss function is

$$L((F(x_k))_j, \hat{y}_{k,j}) =$$
$$= \hat{y}_{n,j} \ln \left(\frac{(F(x_n))_j}{\hat{y}_{n,j}} \right), n = 1, \ldots, N, j = 1, \ldots, M. \quad (3.7)$$

2. As a classifier, a multilayer perceptron with one hidden layer is used, such that the activation function g in its hidden layer is smooth and includes no bias, and its output layer performs the softmax normalization of the hidden layer. Hence,

$$(F(x))_j = \frac{\exp(g(w_{j\cdot}^\top x))}{\sum_{i=1}^{M} \exp(g(w_{i\cdot}^\top x)}. \tag{3.8}$$

The weight vectors $w_{1\cdot}, \ldots, w_{M\cdot}$ in (3.8) are learnt through the minimization of the error function (3.5).

3.3.2 Semi-Supervised Kernel-Based Fuzzy C-means

This method, in detail described in [15], originated from the fuzzy c-means clustering algorithm [16]. Similarly to the original fuzzy c-means, the method is parameterized by a parameter $m > 1$. What makes this method more general than the original fuzzy c-means is its dependence on the choice of some kernel K, i.e., a symmetric function on pairs (x, y) of clustered vectors, which has positive semidefinite Gramm matrices (e.g., Gaussian or polynomial kernels). In fact, the fuzzy c-means algorithm corresponds to the choice $K(x, y) = x^\top y$.

First, the membership matrix U^l is constructed, for clustering n_l labelled instances $x_1^l, \ldots, x_{n_l}^l$ into as many clusters as there are classes, i.e., M. For $j = 1, \ldots, M, k = 1, \ldots, n_k$,

$$U_{j,k}^l = \begin{cases} 1 & \text{if the instance } x_k^l \text{ is labelled with the class } j \\ 0 & \text{else.} \end{cases} \tag{3.9}$$

From U^l, the initial cluster centers are constructed as

$$v_j^0 = \frac{\sum_{k=1}^{n_l} U_{j,k}^l x_k^l}{\sum_{k=1}^{n_l} U_{j,k}^l}, j = 1, \ldots, M. \tag{3.10}$$

If for some $t = 0, 1, \ldots$, the cluster centers $v_1^t, \ldots, v_M^t$ are available, such as (3.10), then they are used together with the chosen kernel K to construct the membership matrix $U^{u,t}$ for clustering n_u unlabelled instances $x_1^u, \ldots, x_{n_u}^u$, as follows:

$$U_{j,k}^{u,t} = \frac{(1 - K(x_k^u, v_j))^{-\frac{1}{m-1}}}{\sum_{i=1}^{M}(1 - K(x_k^u, v_i))^{-\frac{1}{m-1}}}, \quad j = 1, \ldots, M,\ k = 1, \ldots, n_u. \tag{3.11}$$

Finally, the cluster centers are updated, for $t = 0, 1, ..$ by calculating

$$v_j^{t+1} = \\ = \frac{\sum_{k=1}^{n_u}(U_{j,k}^l)^m K(x_k^l, v_j^t)x_k^l + \sum_{k=1}^{n_u}(U_{j,k}^{u,t})^m K(x_k^u, v_j^t)x_k^u}{\sum_{k=1}^{n_u}(U_{j,k}^l)^m K(x_k^l, v_j^t) + \sum_{k=1}^{n_u}(U_{j,k}^{u,t})^m K(x_k^u, v_j^t)}. \quad (3.12)$$

The computations (3.11)–(3.12) are iterated until at least one of the following termination criteria is reached:

(i) $\|U^{u,t} - U^{u,t-1}\| < \varepsilon, t \geq 1$, for a given matrix norm $\|\cdot\|$ and a given $\varepsilon > 0$;

(ii) a given maximal number of iterations $t_{\max}$.

Chapter 4

Proposed Approach

4.1 Overall strategy

Our methodology for the segmentation of video frames into foreground objects and background relies on the assumption that the user labels some area of the image as the object and the background. This manual annotation will be done only in the first frame or in one of the frames in the beginning of the video. This gives us some input information that will be used for training of the semi-supervised classifier.

No matter whether the considered method of semi-supervised classification is semi-supervised classification with cluster regularization or semi-supervised kernel-based fuzzy c-means, the methodology always proceeds in the following steps:

1. In the first frame, the user labels some of the pixels.

2. Using the Boykov-Kolmogorov algorithm, the remaining part of the image is labelled.

3. Matching points detected in the next frame are assigned the same labels as the points to which they are matched.

4. Using the considered method of semi-supervised classification, the remaining points of interest detected in the next frame are labelled.

5. Steps 3 and 4 are repeated till either the points of interest in all frames have been classified or the scene has been so much disrupted between two frames that no points of interest could be matched between them (in such a case, new labelling by the user is needed).

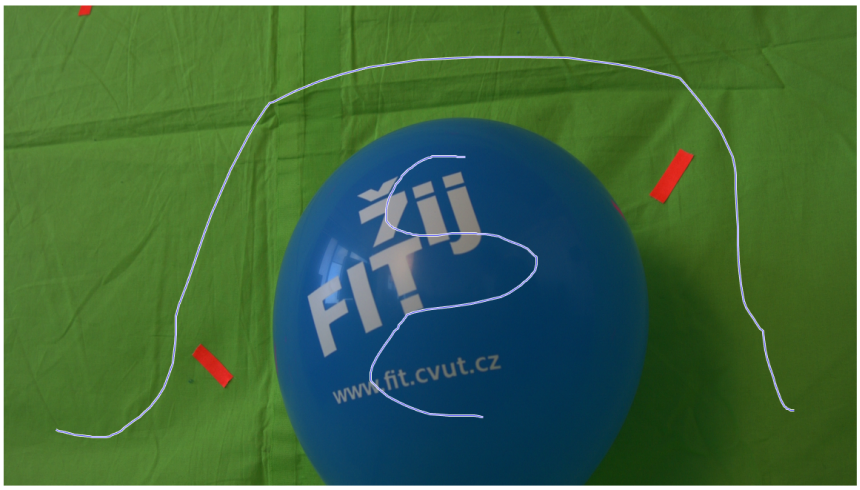

Figure 4.1: Example of the user annotation. Each line corresponds to a different segment of the scene.

4.2 Initial labels provided by the user

The process of manual annotation is as simple as possible. The user is required to draw two lines – one in the region of the object and one in the region of the background. You can see an example of the annotation in Figure 4.1.

4.3 Propagation of the labels

Labels from the user usually cover only a small amount of the pixels. To get the labels for all pixels in the image, max-flow algorithm, described in 2.3 is used.

First, the image is converted to grayscale. To highlight edges between the object and the background, Laplacian of Gaussian filter with the size $(5, 5)$ is applied to the image (Figure 4.2).

Then the image is converted to the four-connected graph. Each node of the graph represents one pixel. For each inner pixel, there are four edges - with the upper, lower, right and left neighbouring pixel. Each edge has weight computed according to the pixel intensities corresponding to the boundary nodes. Let p_i be intensity of the pixel i and $e_{i,j}$ is the edge between the nodes corresponding to the pixels i and j the edge weight $w(e_{i,j})$ is computed as follows:

$$w(e_{i,j}) = 255 - |p_i - p_j| \tag{4.1}$$

Figure 4.2: Image after Laplacian of Gaussian filter was applied.

Then the graph is extended by two artificial nodes. One is marked as source and the other as sink. All pixels labeled by a user as the object are connected with the source node by an edge with the weight 255. Analogically, all pixels labeled as the background are connected with the sink node by an edge with the weight 255.

On this extended graph, the Boykov-Kolmogorov algorithm is executed. A minimum cut partitions the graph nodes into two sets. One set corresponds to the source and all the pixels in the set are labeled as the object. The other pixels are labeled as the background. In Figure 4.3, the nodes that belong to the source are marked.

At the end of this process, we have labels of all pixels in the first frame, which are later used for the classification.

4.4 Implementation of Semi-Supervised Classifiers

As input features for both classification methods, the Cartesian coordinates $([p_k]_1, [p_k]_2)$ of the point in the k-th frame and and the polar coordinates $([p_k - p_{k-1}]_{||}, [p_{k+1} - p_k]_{\varphi})$ of its movement with respect to the previous frame are used.

The Cartesian coordinates $([p]_1, [p]_2)$ of a point p of interest are expressed with respect to the top left corner of the frame, using as units the frame height and width. Due to that, $[p]_1$ and $[p]_2$ are normalized to $[0, 1]$.

Figure 4.3: An object found by Boykov-Kolmogorov algorithm from the image in the Figure 4.1.

4.4.1 Semi-Supervised Classification with Cluster Regularization

In the implementation of the semi-supervised classification with cluster regularization method described in 3.3.1, we used k-means clustering for an initial clustering of all instances. Although this method allows choosing the number of clusters independently of the number of classes, we have set it to the same value for comparability with semi-supervised kernel-based fuzzy c-means, i.e., to the value 2 corresponding to the classes of foreground objects and background. Hence, we performed k-means clustering with $k = 2$. Since the k-means algorithm does not output a probability distribution on the set of clusters, we employed a simple procedure proposed in [14] to transform the original distances from an instance x_n to cluster centers $v_1, \ldots, v_k$ to a probability distribution q_n, which assures that x_n more likely belongs to clusters which centers it is closer to:

$$(q_n)_i = \frac{1 - \left(\frac{\|x_n - v_i\|}{\sum_{j=1}^{k} \|x_n - v_i\|} \right)}{k - 1}. \tag{4.2}$$

Consequently, for our case $k = 2$:

$$(q_n)_1 = \frac{\|x_n - v_2\|}{\|x_n - v_1\| + \|x_n - v_2\|}, \tag{4.3}$$

$$(q_n)_2 = \frac{\|x_n - v_1\|}{\|x_n - v_1\| + \|x_n - v_2\|}. \tag{4.4}$$

The remaining parameters pertaining to semi-supervised classification with cluster regularization were set as proposed in [14]: $\lambda = 0.2, \kappa = 2, |\phi(x_n)| = 10$.

4.4.2 Semi-Supervised Kernel-Based Fuzzy C-means

For the semi-supervised kernel-based fuzzy c-means algorithm described in 3.3.2, we used a Gaussian kernel function for updating the membership matrix $K(x, y) = exp(-\|x - y\|^2/\sigma^2)$, in which the parameter σ is computed as proposed in [15]:

$$\sigma = \frac{1}{M}\sqrt{\frac{\sum_{n=1}^{N} \|x_n - v\|^2}{N}}, \tag{4.5}$$

where v is the center of all instances. The remaining parameters were set as follows: $m = 2, \varepsilon = 0.001, t_{max} = 50$.

Chapter 5

Experimental Validation

This chapter deals with the experimental validation of our proposed approach. First, the employed data and implementation is described. Then, the obtained results are illustrated and we analyze them.

5.1 Employed Data

For the validation of the proposed approach, we prepared 6 short videos. In all videos, there is a yellow or blue balloon as a foreground object and a green background. On the background, there are a few small red sticky notes to help detecting some key points. The videos were recorded in a UHD resolution.

Here is a brief characterization of all employed videos:

- a handheld camera, both the foreground object and the background are sharp,
- a static camera, only the foreground object is sharp, a hand is interfering with the background (2 videos),
- a static camera, only the foreground object is sharp, it is close to the camera,
- a static camera, only the foreground object is sharp, it is moving towards the camera,
- a static camera, only the foreground object is sharp, it is moving away from the camera.

For the testing, labels were available for all points of interest. Unfortunately, those labels were often unreliable.

5.2 Implementation

An ORB tracker was implemented in C++, employing the OpenCV library. A video was processed by this tracker and motion vectors were exported as csv files for further analysis.

A user interface for initial annotation with the propagation of those labels to all pixels and semi-supervised classification were implemented in Matlab.

The GMM algorithm for comparison of our approach was implemented in Python employing the scikit-learn library.

5.3 Results and Their Analysis

In all the employed videos, we measured the quality of classification by means of the accuracy, sensitivity, specificity and F-measure of both implemented classification methods.

For the fuzzy c-means method, those quality indicators are illustrated for four particular videos in Figures 5.1 and 5.2. The video with a handheld camera has a very good accuracy and F-measure, but others are only slightly above 50 %.

For the cluster regularization method, the same is illustrated in Figures 5.3 and 5.4. We can see that all videos, except for the video with the object moving from the camera, have a F-measure between 70 % and 90 % and an accuracy between 60 % and 90 %.

For the comparison with our proposed approach, we measured the quality of the GMM method in the same videos. The results are in Figuress 5.5 and 5.6

Table 5.1 shows that, according to the Friedman test, we can't reject the hypothesis that all methods have equal quality measures in delays 1, 5 and 10 frames between classifier training and measuring its quality. The results are comparable but the reason for choosing semi-supervised classification over GMM can be more efficient processing of videos in high resolution.

Table 5.1: Results of the Friedman test of the hypothesis that for a given delay between classifier training and measuring its quality, a given quality measure is equal for all considered classifiers.

Quality measure	Delay	p-Value
accuracy	1	0.8465
accuracy	5	0.5134
accuracy	10	0.3114
sensitivity	1	0.8465
sensitivity	5	0.6065
sensitivity	10	0.3114
specificity	1	0.6065
specificity	5	0.1146
specificity	10	0.3114
F-measure	1	0.8465
F-measure	5	0.5134
F-measure	10	0.3114

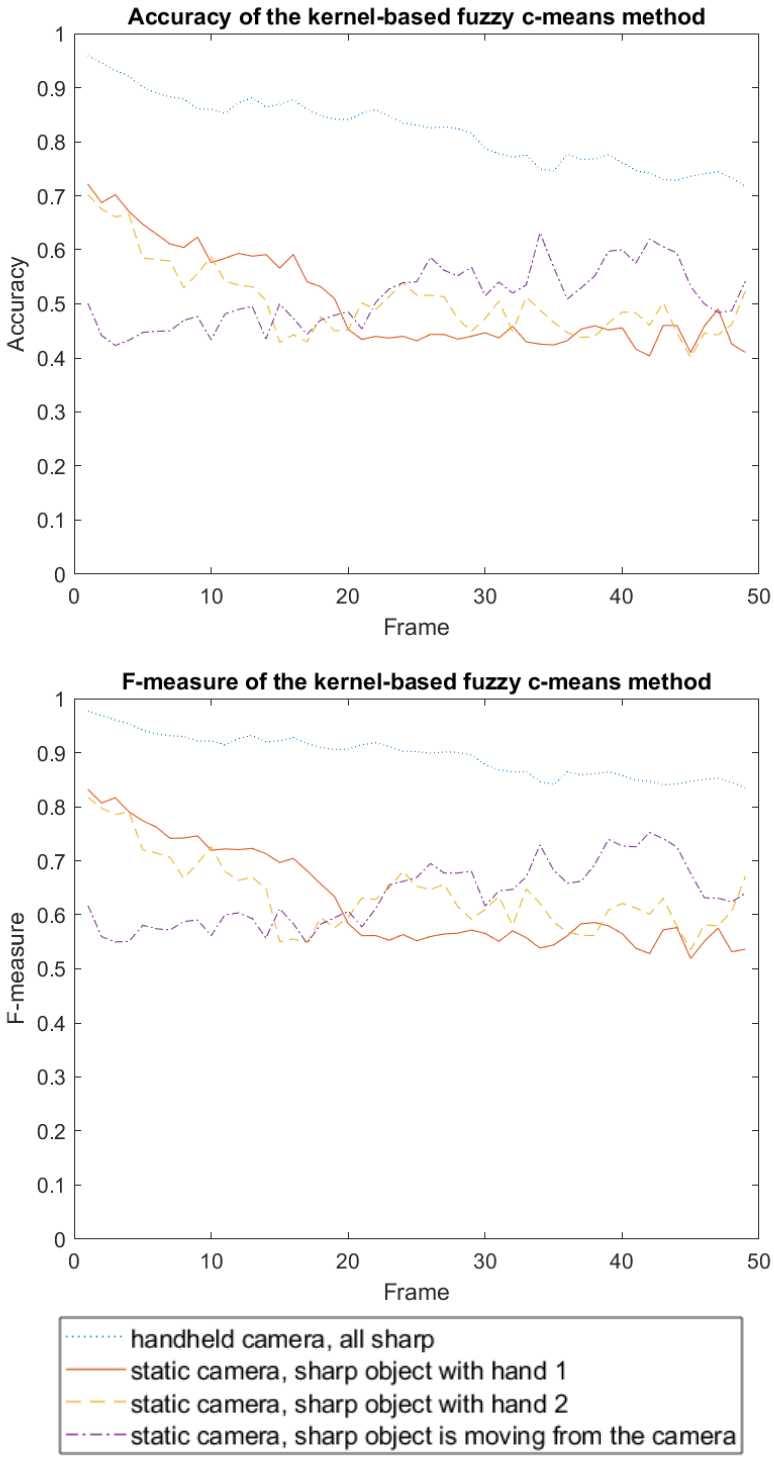

Figure 5.1: The evolution of accuracy and F-measure of the c-means method on the unlabelled data for four particular videos

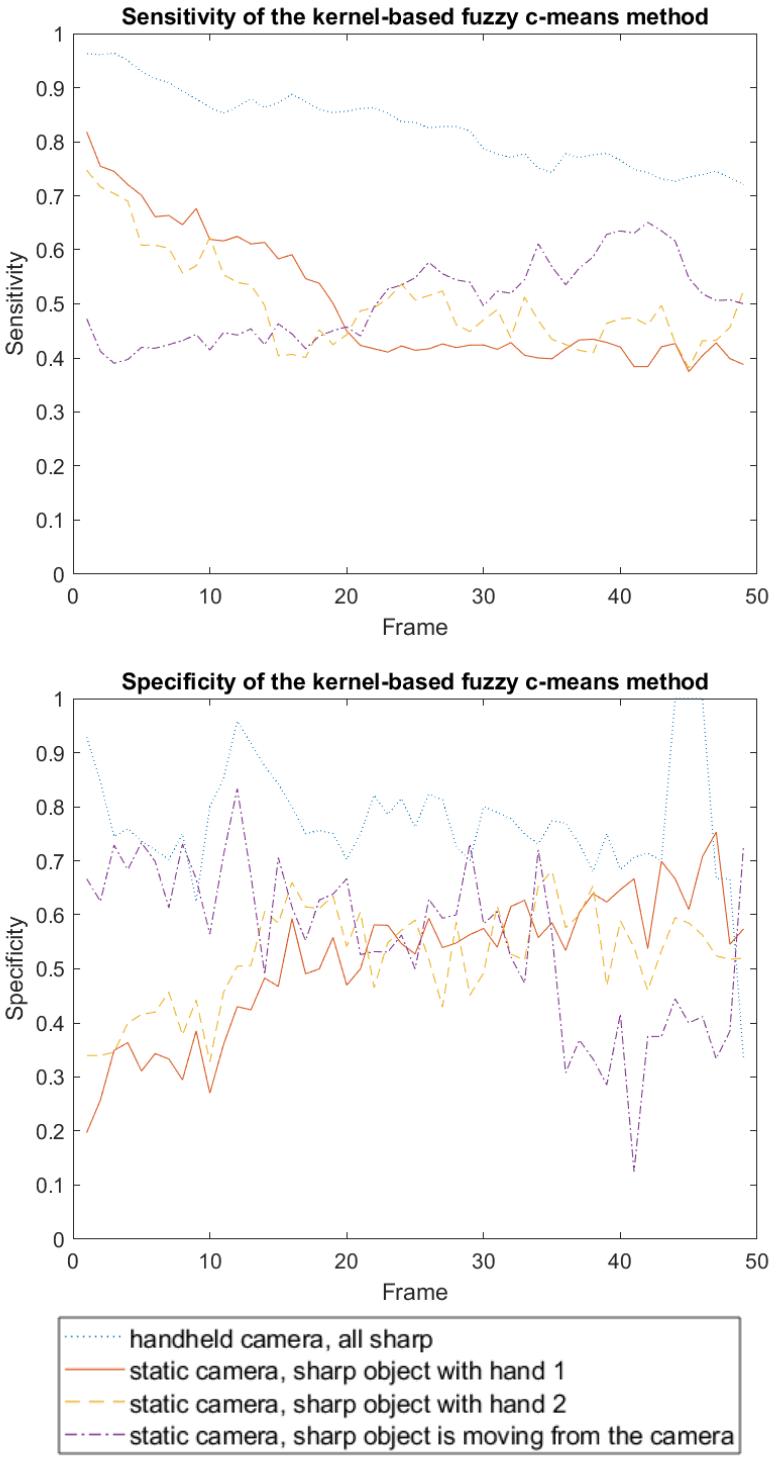

Figure 5.2: The evolution of sensitivity and specificity of the c-means method on the unlabelled data for four particular videos

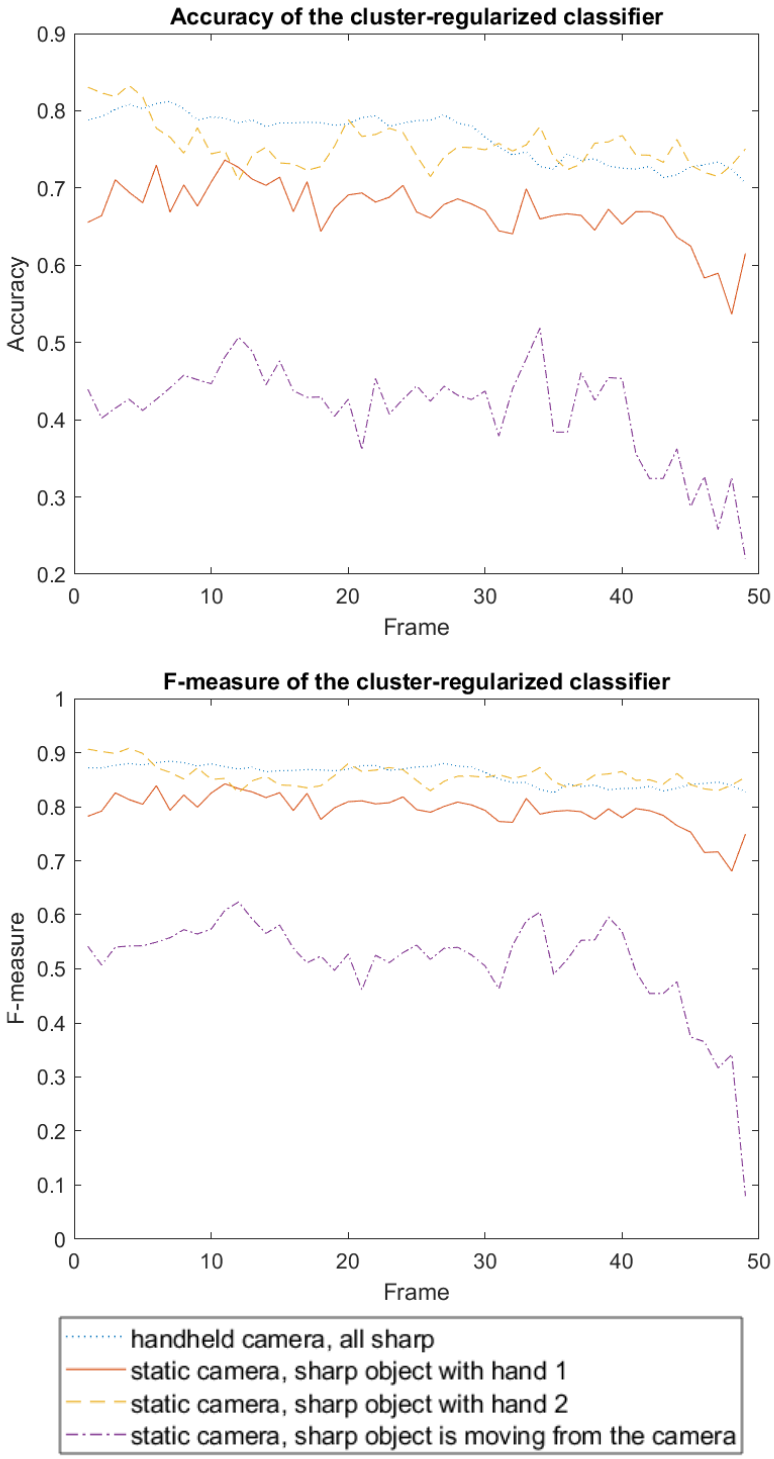

Figure 5.3: The evolution of accuracy and F-measure of the cluster regularization method on the unlabelled data for four particular videos

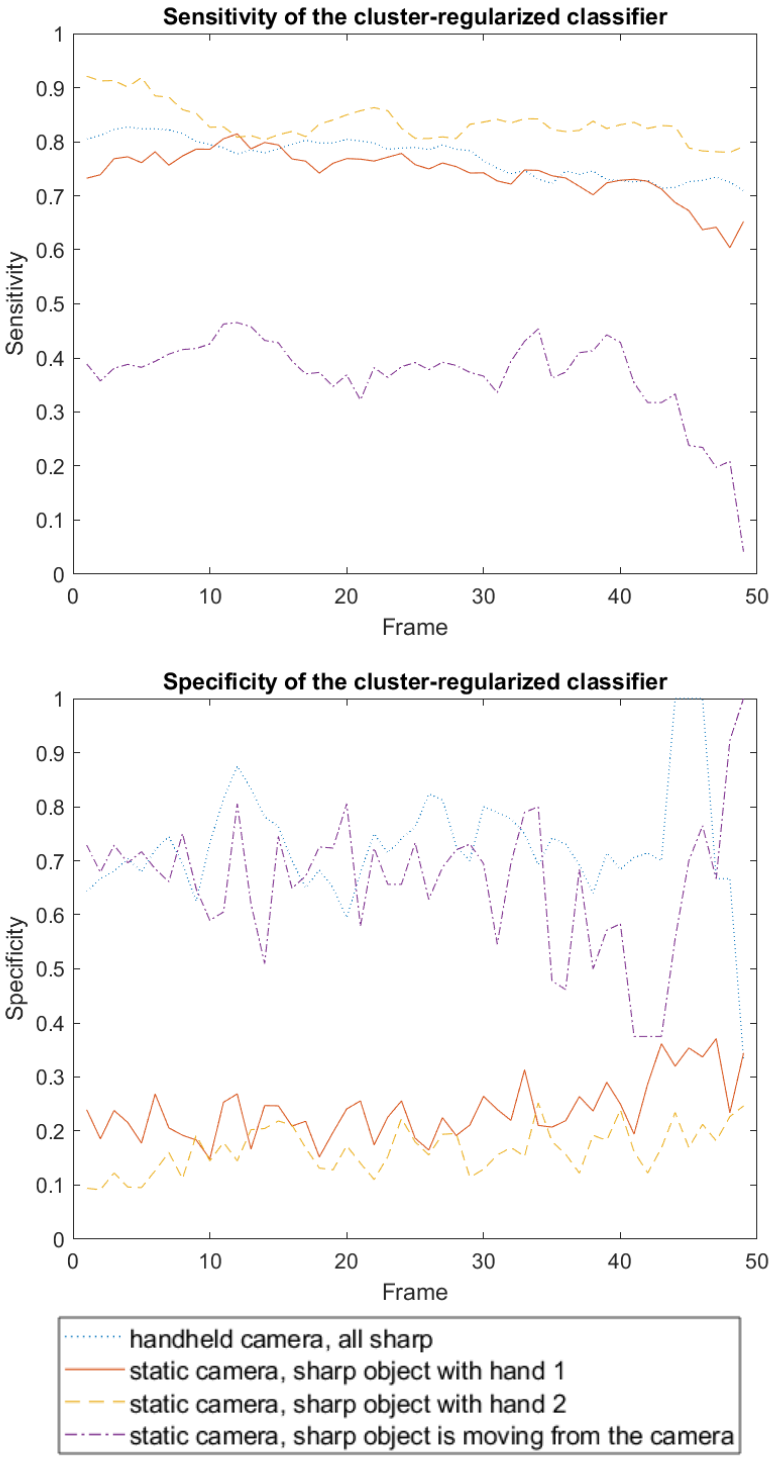

Figure 5.4: The evolution of sensitivity and specificity of the cluster regularization method on the unlabelled data for four particular videos

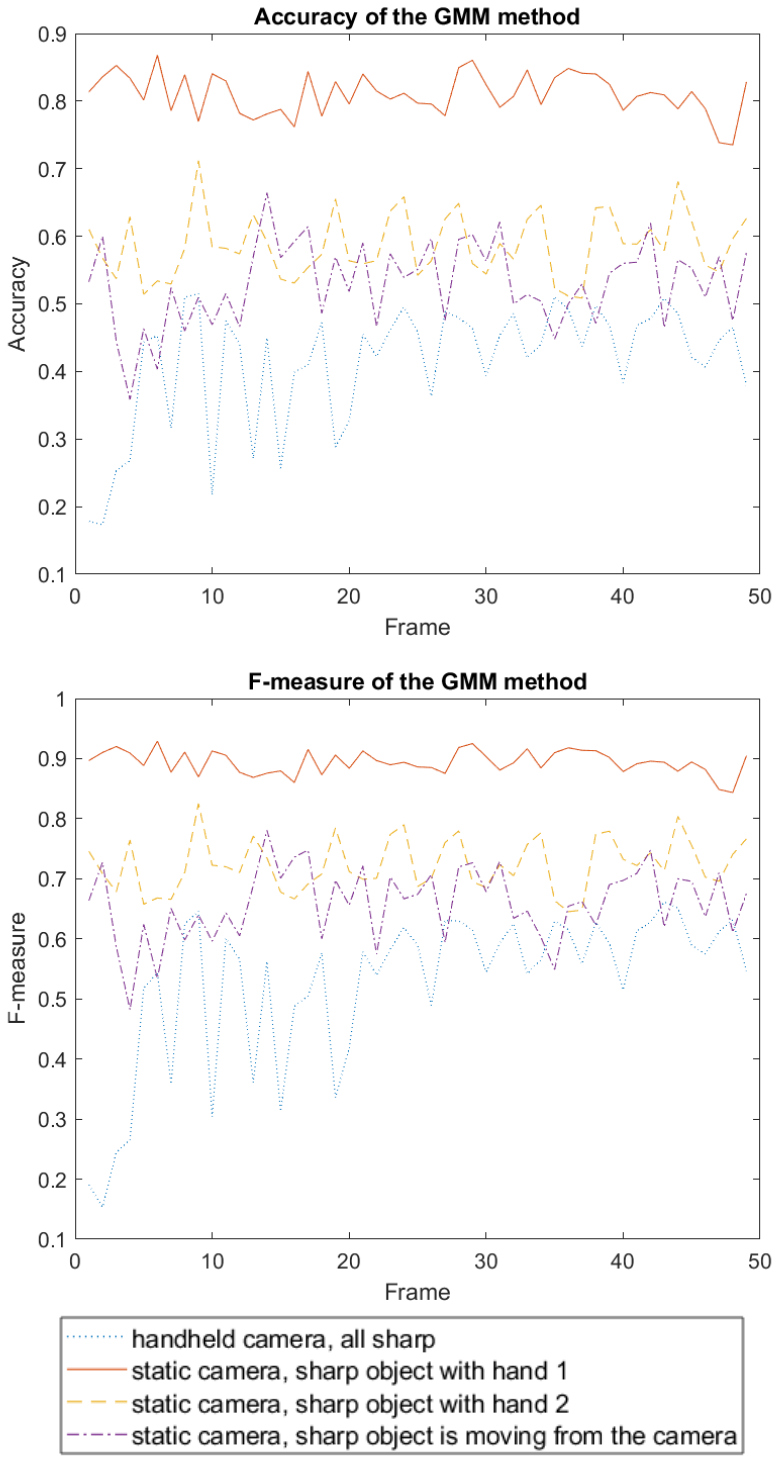

Figure 5.5: The evolution of accuracy and F-measure of the GMM method on the unlabelled data for four particular videos

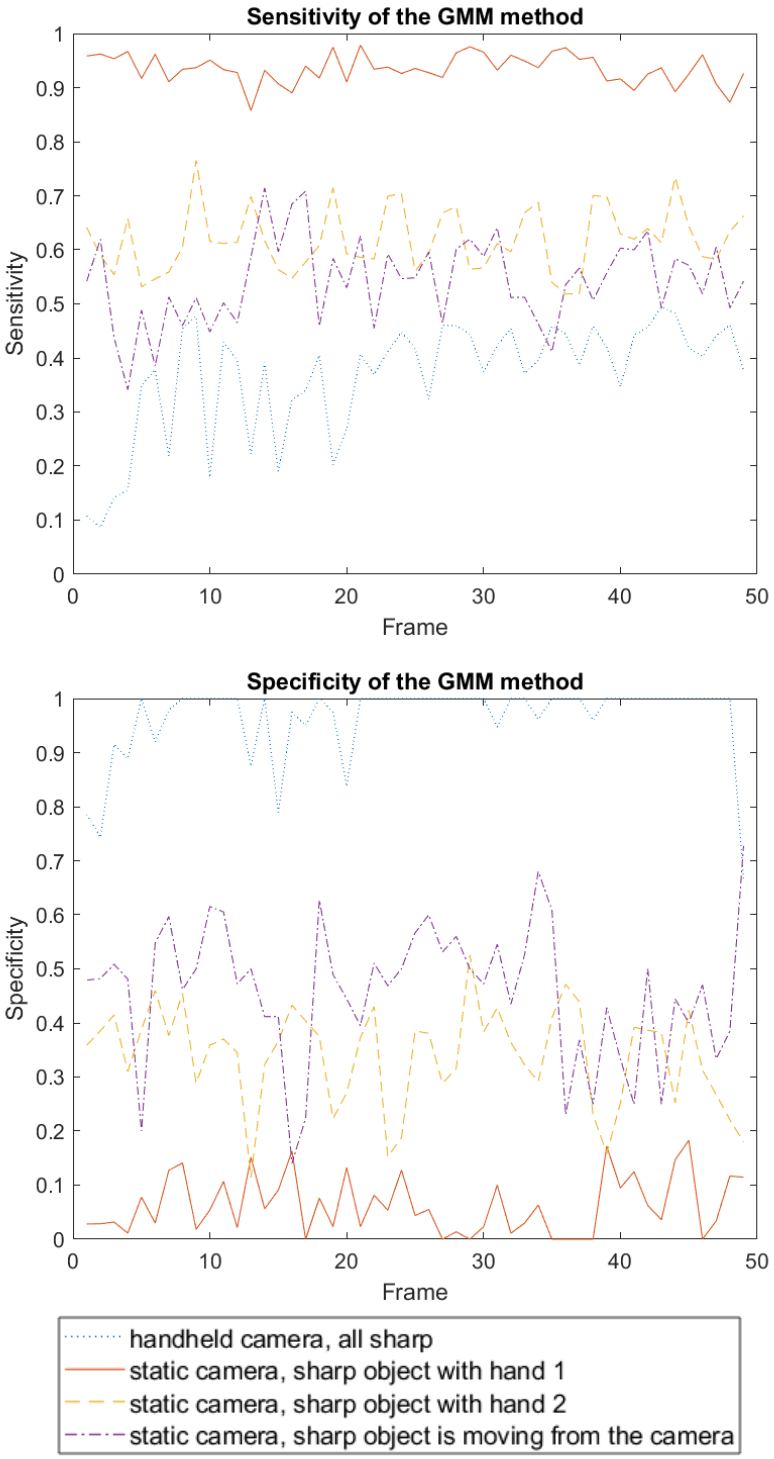

Figure 5.6: The evolution of sensitivity and specificity of the GMM method on the unlabelled data for four particular videos

Conclusion

This thesis deals with scene segmentation of the video in UHD resolution. Traditional methods work on the pixel level and they are not efficient for use on high resolution.

The presented research integrates two comparatively recent approaches, the key point detector ORB, which is a combination of a corner detection method FAST with a visual descriptor method BRIEF, and two semi-supervised classification methods – Semi-supervised classification with cluster regularization and Semi-supervised kernel-based fuzzy c-means. To our knowledge, this is the first time these approaches are used together for the task of scene segmentation into the foreground objects and the background.

For the initial annotation, a user is involved to provide some labels. Those are propagated to the whole image with the usage of max-flow algorithm Boykov-Kolmogorov. The labels are moved to the matched key points on the following frames. Key points that don't have labels are classified using one of the semi-supervised classifiers. By this process, we get labels for all detected key points in each frame.

Results we get are comparable to the results achieved by one of the traditional methods - Gaussian Mixture Model. Friedman test shows that we can't decline the hypothesis that all methods have equal quality measures in delays 1, 5 and 10 frames between classifier training and measuring its quality. An advantage of our proposed approach is time efficiency.

Both approaches should be investigated in the context of more complex segmentation and more realistic scenes. To this end, however, especially the ORB detector needs to be more deeply elaborated using with methods of semisupervised classification.

Bibliography

[1] Smolyansky, E. The Basics of Video Object Segmentation. *Techburst.io* [online]. Sep 12, 2017 [cit. 2018-12-03]. Available from: `https://techburst.io/video-object-segmentation-the-basics-758e77321914`

[2] Power, P. W.; Schoonees, J. A. Understanding background mixture models for foreground segmentation. In *Proceedings image and vision computing New Zealand*, volume 2002, 2002.

[3] Rublee, E.; Rabaud, V.; et al. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, IEEE, 2011, pp. 2564–2571.

[4] OpenCV. ORB (Oriented FAST and Rotated BRIEF). *OpenCV documentation* [online]. ©2011-2014 [cit. 2018-12-03]. Available from: `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html`

[5] Boykov, Y.; Kolmogorov, V. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, volume 26, no. 9, 2004: pp. 1124–1137.

[6] Kerul̓-Kmec, O.; Pulc, P.; et al. Semisupervised Segmentation of UHD Video. In *Proceedings of the 18th Conference Information Technologies - Applications and Theory (ITAT 2018), Hotel Plejsy, Slovakia, September 21–25, 2018.*, 2018, pp. 100–107. Available from: `http://ceur-ws.org/Vol-2203/100.pdf`

[7] Stauffer, C.; Grimson, W. E. L. Adaptive background mixture models for real-time tracking. In *cvpr*, IEEE, 1999, p. 2246.

[8] Dempster, A. P.; Laird, N. M.; et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1977: pp. 1–38.

[9] Rosin, P. L. Measuring corner properties. *Computer Vision and Image Understanding*, volume 73, no. 2, 1999: pp. 291–307.

[10] Calonder, M.; Lepetit, V.; et al. Brief: Binary robust independent elementary features. In *European conference on computer vision*, Springer, 2010, pp. 778–792.

[11] Pulc, P.; Holeňa, M. Towards real-time motion estimation in high-definition video based on points of interest. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sep. 2017, pp. 67–70, doi:10.15439/2017F417.

[12] Flach, P. A. *Machine learning: the art and science of algorithms that make sense of data*. New York: Cambridge University Press, 2012, ISBN 1107422221.

[13] Chapelle, O.; Scholkopf, B.; et al. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, volume 20, no. 3, 2009: pp. 542–542.

[14] Soares, R. G.; Chen, H.; et al. Semisupervised classification with cluster regularization. *IEEE transactions on neural networks and learning systems*, volume 23, no. 11, 2012: pp. 1779–1792.

[15] Zhang, D.; Tan, K.; et al. Semi-supervised kernel-based fuzzy c-means. In *International Conference on Neural Information Processing*, Springer, 2004, pp. 1229–1234.

[16] Bezdek, J. C. Objective function clustering. In *Pattern recognition with fuzzy objective function algorithms*, Springer, 1981, pp. 43–93.

Appendix A

Acronyms

BRIEF Binary Robust Independent Elementary Features

FAST Features from Accelerated Segment Test

GMM Gaussian Mixture Model

ORB Oriented FAST and Rotated BRIEF

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `thesis` ..... the directory of LaTeX source codes of the thesis
  - `impl` ..... the directory of source codes of the implemented methods
    - `data` ..... sample data
- `text` ..... the thesis text directory
  - `DP_Kerul-Kmec_Oliver_2019.pdf` ..... the thesis text in PDF format