Institute for Parallel and Distributed Systems
Machine Learning and Robotics Lab

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

— Master's Thesis Nr. 15 —

Inferring Object Hypotheses Based on Feature Motion from Different Sources

Steffen Fuchs

Course of Study:	M.Sc. Informatik
Examiner:	Prof. Dr. Marc Toussaint
Supervisor:	M.Sc. Stefan Otte
Commenced:	15.10.2014
Completed:	16.04.2015
CR-Classification:	I.2.10, I.4.8, I.5.3

Abstract

Perception systems in robotics are typically closely tailored to the given task, e.g., in typical pick-and-place tasks the perception systems only recognizes the mugs that are supposed to be moved and the table the mugs are placed on. The obvious limitation of those systems is that for a new task a new vision system must be designed and implemented. This master's thesis proposes a method that allows to identify *entities* in the world based on motion of various features from various sources. This is without relying on strong prior assumptions and to provide an important piece towards a more general perception system. While entities are rigid bodies in the world, the sources can be anything that allows to track certain features over time in order to create trajectories. For example, these feature trajectories can be obtained from RGB and RGB-D sensors of a robot, from external cameras, or even the end effector of the robot (proprioception).

The core conceptual elements are: the distance variance between trajectory pairs is computed to construct an affinity matrix. This matrix is then used as input for a divisive k-means algorithm in order to cluster trajectories into object hypotheses. In a final step these hypotheses are combined with previously observed hypotheses by computing the correlations between the current and the updated sets. This approach has been evaluated on both simulated and real world data. Generating simulated data provides an elegant way for a qualitative analysis of various scenarios. The real world data was obtained by tracking Shi-Tomasi corners using the Lucas-Kanade optical flow estimation of RGB image sequences and projecting the features into range image space.

Contents

Chapter 1

Introduction

1.1 Motivation

Identifying objects of interest is a key requirement for every robot. While navigating through an uncontrolled environment a robot has to avoid potential obstacles such as pedestrians or cars on the road. Localization and mapping tasks (SLAM) require to track landmarks in order to estimate the relative sensor position. For manipulation applications the robot has to find the desired object and know about how it can be manipulated. For all these tasks motion is either the mandatory element or they can benefit from it. In fact the importance of motion for perceiving objects originates from the "Principles of Gestalt Psychology" by Max Wertheimer, Wolfgang Kohler and Kurt Koffka [6]. In particular the principle of "common fate" states that for humans, elements are perceived as grouped together if they move along a common and smooth path. This principle has already been applied successfully to computer vision problems such as image segmentation [9], [5] or estimation of kinematic models [8].

When creating new robot applications designing the necessary perception system usually requires a great amount of effort. As a result for many cases these systems are typically closely tailored to the given task. For example, in a pick-and-place scenario a simple perception system may identify the table as a horizontal plane with a certain distance to the ground and recognize the objects by learning models of them in advance. The obvious limitation to this approach is that for a new task the perception system must be redesigned, e.g., training new objects, changing the model type or using different features. For this reason there is always high demand for more generalization of perception systems in the robotic community. Would it not be great to just show the robot the object we are interested in? Why can we not just pick up the mug from the table and place it back in order to make the robot learn the model? Why not letting the robot observe how to pull out a drawer or open a door? One does not show an infant static

pictures of toys with bounding boxes to teach how to play with them.

1.2 Objective

The goal of this thesis is to provide a small step towards more general perception systems that are based on motion. We will address how the previously mentioned principal of common fate can be applied to infer hypotheses about objects by grouping features of similar motion. In particular we assume that we receive information about the motion of certain features from different sources and analyse their similarity in terms of rigid body motion. These sources typically are trackers of various kinds that work with a diversity of feature types and on different sensors. For example, the Lucas-Kanade tracker uses corners as features in color images while another source might use geometric primitives in point clouds obtained from a laser scanner. In fact it is part of the initial idea to be able to fuse trackers of different types of features in order to compensate the loss of information in situations where one tracker might fail. The output of these sources is already very general as they regularly provide us with observations of the same *thing* at different points in time. We will denote these observations as a *feature trajectory*. Observations can be simply interpreted as measurements of states of the feature depending on time. These states are usually points or poses in Euclidean space. Therefore the only requirement for comparability is that the states of different features are comparable. Further we will use the term *object hypothesis* to define the set of all features that have similar motions with respect to there trajectories.

In order to obtain these hypotheses we compute the distance variance between all trajectory pairs to construct an affinity matrix. This matrix is then used as input for a divisive k-means algorithm in order to cluster trajectories into object hypotheses. In a final step these hypotheses are combined with previously observed hypotheses by computing the correlations between the current and the updated sets.

Chapter 2

Related Work

The idea to use motion in order to learn from the environment is not really new. One popular type of application builds around image segmentation. For example Kenney et al. [5] use the interactive capabilities of a robot manipulator to move objects and compute the difference of two consecutive color images in order to segment the objects from the background. Ochs and Brox [9] present a variational method to combine motion and color segmentation. Their off-line approach computes the feature trajectories for an image sequence to obtain a first segmentation based on motion. Then they refine this sparse segmentation for every image based on the color information.

Learning kinematic models is another field of application. Sturm et al. [13] fit and track planes in a 3D scene reconstructed from a sequence of stereo images. These tracks are then used to learn articulation models of drawers and doors as they typically can be found in domestic environments. The method proposed by Martin and Brock [8] has a similar motivation but is not limited to planar surfaces. They use the KLT feature tracker on the color image of an RGB-D sensor to feed a multi-level state estimation approach in order to obtain articulation models. Their method seems very capable of detecting unknown objects based on motion. However they state they require at least 15 feature points for each rigid body which would place limitations on the type of trackers that can be used.

Clustering of feature trajectories has also been addressed in previous work. Yan and Pollefeys [16] propose a method for motion segmentation under affine projections. It is based on a multi-body factorization method originally introduced by Costeria and Kanade [4]. They also suggest a divisive clustering step using spectral clustering. Because their method requires trajectories to be of equal length and frequency it seems not really practical for online use where trajectories spawn and die regularly.

The approach proposed by Brox and Malik [3] probably had the most influence on the method described in this thesis. Their off-line method com-

putes the large displacment optical flow [14] between every image pair of a given video and uses this flow to create long term trajectories of points in dense grid cells. They also perform a pair-wise comparition in order to construct an affinity matrix. However they define dissimilarity of trajectories where both motion vectors are most dissimilar which only allows translational motion. They compensate for this by relying on the dense set of tracked points and only allow high affinities for spatially close points.

Chapter 3

Inferring Object Hypotheses

This chapter addresses the details of the proposed solution by providing the theoretical foundation as well as giving some insight into implementation related issues. The chapter is split into three parts corresponding to the three core elements. Figure 3.1 illustrates these components. The first section addresses the *kernel* which receives updates of the feature trajectories from different tracker sources. The kernel constructs an affinity matrix by pair-wise comparing the feature trajectories. The second sections describes the *clustering* process. A divisive k-means approach uses this matrix to create new object hypotheses. The final section explaines how these results are combined with previously obtained object hypotheses and propagated over time.

3.1 Distance Variance of Trajectory Pairs

Given a set of feature trajectories

$$\mathcal{F} := \left\{ f_i : \mathbb{R} \to \mathbb{R}^d \right\}_{i=1}^{N}$$

where each trajectory $f_i : t \mapsto \boldsymbol{x}$ describes the state $\boldsymbol{x}$ of the ith feature at time t. A feature can be anything that allows to easily find correspondenses

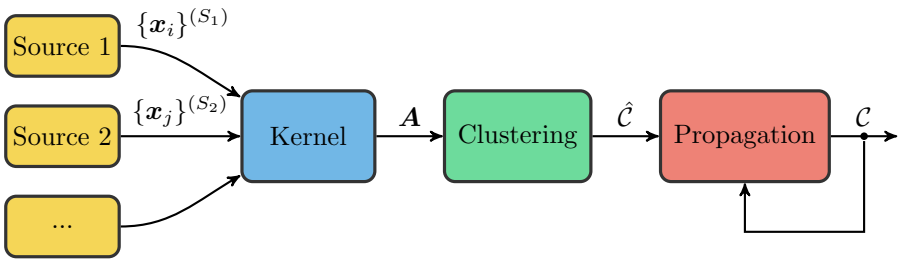

Figure 3.1: Conceptual overview of the proposed method

of the *same "thing"* over a period of time. Hence the feature itself is time invariant, while its state is not. A state usually is a position in 2D or 3D Euclidean space but could as well be a 6-D pose.

Now assume there exists some time independent association between two different features of the form $\phi_i : \boldsymbol{x} \mapsto \hat{\boldsymbol{x}}$. Let $X \sim f_j(t)$ be the set of states described by trajectory f_j over a finite observation time $t \in [0, T]$ and $\hat{X} \sim \phi_i(f_i(t))$ be the results of some association function. We can then write the mean squared error of this association as

$$\begin{aligned} E[(\hat{X} - X)^2] &= E[\hat{X}^2] + E[X^2] - 2E[\hat{X}X] \\ &= E[\hat{X}^2] - (E[\hat{X}])^2 + (E[\hat{X}])^2 + E[X^2] - (E[X])^2 + (E[X])^2 \\ &\quad - 2(E[\hat{X}X] - E[\hat{X}]E[X] + E[\hat{X}]E[X]) \\ &= \mathrm{Var}(\hat{X}) - 2\mathrm{Cov}(\hat{X}, X) + \mathrm{Var}(X) + (E[\hat{X} - X])^2 \end{aligned}$$

which is

$$E[(\hat{X} - X)^2] = \mathrm{Var}(\hat{X} - X) + (E[\hat{X} - X])^2 \tag{3.1}$$

since

$$\begin{aligned} \mathrm{Var}(\hat{X} - X) &= E[(E[\hat{X} - X] - (\hat{X} - X))^2] \\ &= E[(\hat{X} - X)^2] - (E[\hat{X} - X])^2 \\ &= E[\hat{X}^2] - 2E[\hat{X}X] + E[X^2] - (E[\hat{X}])^2 + 2E[\hat{X}]E[X] - (E[X])^2 \\ &= \mathrm{Var}(\hat{X}) - 2\mathrm{Cov}(\hat{X}, X) + \mathrm{Var}(X). \end{aligned}$$

Let us now think in terms of rigid body motion. If two features belong to the same rigid object then their association function

$$\phi_i(f_i(t)) - f_j(t) = \boldsymbol{G}_i \boldsymbol{x}_i - \boldsymbol{x}_j = \boldsymbol{0}$$

states that at any time t there exists exactly one affine transformation $\boldsymbol{G}_i$ that maps the state of trajectory $f_i(t)$ to $f_j(t)$. Or putting it differently, if $\boldsymbol{G}_i$ is the identity transformation then the distance between both trajectories

$$d_{ij}(t) = \|f_i(t) - f_j(t)\| = \sqrt{[f_i(t) - f_j(t)]^T [f_i(t) - f_j(t)]}$$

should be constant. In this case the mean squared error of an association can be decomposed into the squared mean distance $(E[\hat{X} - X])^2$ and the distance variance $\mathrm{Var}(\hat{X} - X)$ as stated in (3.1). Although trajectories of different objects still have a mean distance for an observation time interval $[0, T]$, their variance is significantly higher as if they were moving along the same path. This makes the distance variance a strong indicator of whether

a pair of feature trajectories result from the same rigid body, at least if one of the trajectories represents motion.

In case both trajectories do not move at all, although their variance is small, this does not imply a strong association between them. In other words motion is an important requirement in order to make assumptions about trajectory pairs and needs to be present in at least one of them. Hence instead of just using the uniform weighted variance

$$\frac{1}{T}\int_{t=0}^{T} d_{ij}^2(t)\,\mathrm{d}t - \left(\frac{1}{T}\int_{t=0}^{T} d_{ij}(t)\,\mathrm{d}t\right)^2$$

we assign weights

$$w_{ij}(t) = 1 - \exp\{-\lambda\,(v_{ij}^2(t) + \epsilon)\} \tag{3.2}$$

with

$$v_{ij}^2(t) = \max\left\{\|\dot{f}_i(t)\|^2,\ \|\dot{f}_j(t)\|^2\right\}$$

as the greater velocity of both trajectories at t, a small constant factor $\epsilon > 0$ to omit zero weights, and $\lambda > 0$ as a scaling parameter. This penalizes the stationary sections of trajectory pairs without emphasising high motion to much. An important aspect of this is to add some dynamic to the comparing process so that when an object suddenly starts to move, this information immediately can affect the variance. Otherwise the variance would rise too slowly.

This leads to the new measure

$$\sigma_{ij}^2 = \eta\int_{t=0}^{T} w_{ij}(t)d_{ij}^2(t)\,\mathrm{d}t - \left(\eta\int_{t=0}^{T} w_{ij}(t)d_{ij}(t)\,\mathrm{d}t\right)^2 \tag{3.3}$$

with

$$\eta = \left(\int_{t=0}^{T} w_{ij}(t)\,\mathrm{d}t\right)^{-1}$$

as the normalization. Putting all this together we obtain an affinity matrix $\boldsymbol{A} \in [0,1]^{N\times N}$ with elements

$$a_{ij} = a_{ji} = \exp\{-\gamma\,\sigma_{ij}^2\} \tag{3.4}$$

and another scaling parameter $\gamma > 0$. The next part of this section talks more about the special considerations needed when implementing this. It also points out how the parameters λ and γ of (3.2) and (3.4) affect the outcome and how to choose them.

Implementation

Previously we have silently assumed that a feature trajectory is a continuous function of states $\boldsymbol{x}$ over time and every trajectory is observable over the same time interval $t \in [0, T]$. In practice this is rarely the case and we rather have to work with a discrete set of measured states at different points in time which introduces some additional difficulties. In the following let $m \in [1, M_i]$ and $k \in [1, M_j]$ be the running index over observations of trajectory i and j. M_i and M_j denote the total number of available observations. Let us redefine a trajectory as

$$f_i := \left\{ \boldsymbol{x}_{i,m} \in \mathbb{R}^d,\ t_{i,m} \in \mathbb{R}\ :\ t_{i,m} < t_{i,m+1} \right\}_{m=1}^{M_i}$$

where $t_{i,m}$ is the timestamp when state $\boldsymbol{x}_{i,m}$ was observed. The issues that arise from this when comparing a trajectory pair i and j can be summarized as:

(a) $t_{i,m} \neq t_{j,m} : \forall m \wedge M_i \neq M_j$. This means that observations of both trajectories are not aligned and cannot be treated as a fix sized vector. Therefore an element-wise comparision is not possible.

(b) $t_{i,m} \neq t_{j,k} : \forall m, k$. This is an extension to the previous statement. It states that there is no guarantee that an alignment exists even for a subset of observations of two trajectories. This can be caused by trackers working at different frequencies.

(c) $t_{i,1} > t_{j,M_j}$. Two trajectories may cover entirely different time intervals and there is no guarantee that a common window exists where a comparision is possible. This is because new trajectories may be created or terminated at any point, e.g. because of occlusion, objects entering or leaving the observation area.

In the discrete case distances between two trajectories are only computed when an observation for either of them was made. We address problems (a) and (b) by using linear interpolation between two consecutive observations. Hence the distance of an observation $\boldsymbol{x}_{i,m}$ of the ith trajectory at $t_{i,m}$ to the jth trajectory is

$$d_{ij,m} = \|\boldsymbol{x}_{i,m} - (a\boldsymbol{x}_{j,k+1} + (1-a)\boldsymbol{x}_{j,k})\|$$

with

$$a = \frac{t_{i,m} - t_{j,k}}{t_{i,k+1} - t_{j,k}}$$

and $t_{j,k} \leq t_{i,m} < t_{j,k+1}$ for $k \in [1, M_j)$. The effort of finding the appropriate time interval $[t_{j,k}, t_{j,k+1})$ can be reduced in practise by computing distances on the fly as soon as a new observation of a trajectory is made. Let $\boldsymbol{x}_{i,m+1}$

be this new observation then $t_{i,m+1}$ and $t_{i,m}$ create a new time interval to which all currently existing observations can be compared to. Since $t_{j,M_j} < t_{i,m+1}\ \forall j \neq i$ it is sufficient to check whether the last observation of the remaining trajectories falls into this new interval and update the set of their distances accordingly. Therefore for every observation $\boldsymbol{x}_{i,m}$ we also maintain a set of distances

$$\{d_{ij,m} : \forall j \neq i \wedge t_{j,k} \leq t_{i,m} < t_{j,k+1} : k \in [1, M_j)\}$$

to all other trajectories that have a time interval where $\boldsymbol{x}_{i,m}$ falls into. The variance σ_{ij} from equation (3.3) is now computed over a set of distances

$$\left\{d_{ij,m}, d_{ji,k} : \forall m, k : \max\{t_{i,1}, t_{j,1}\} \leq t_{i,m}, t_{j,k} \leq \min\{t_{i,M_i}, t_{j,M_j}\}\right\} \quad (3.5)$$

and weights will be applied depending on the velocity

$$v_{ij,m} = v_{i,m} = \frac{\boldsymbol{x}_{i,m} - \boldsymbol{x}_{i,m-1}}{t_{i,m} - t_{i,m-1}}$$

at an observation. It should be clear that the computational cost for creating the affinity matrix not only depends on the number of trajectories but also on their number of observation. This would quickly become infeasible without pruning data. Therefore we simply throw away observations that are older then certain threshold (e.g. 10 sec) and remove the trajectory if it becomes to short. However there might be more clever ways to approach this, for example a stationary feature could easily be described by only using its first and last observation.

Now what remains is to deal with case (c) which occures when we maintain trajectories that have not received an update in a while and trying to compare them with newly created trajectories. As we can see from (3.5) this set will then be empty and choosing an appropriate affinity is not a simple task. So why would we keep outdated trajectories in the first place? In fact old trajectories do not add any information to the clustering process different from when they were *up-to-date*. Since they will not receive any new observations their affinity to other trajectories will not change and it suffices to maintain their last up-to-date object hypothesis. Hence it is save to remove these trajectories from the clustering process. However one should make a descision when to mark a trajectory as *outdated*. If we have trajectories from sources of different frequencies we can still encounter issues stated by (c). In case the number of uncomparable trajectories is small enough, setting their affinity to a constant $a_{ij} = 0.5$ works fine. It should neither emphasize high nor low affinity.

This leaves us with the interpretation of the parameters λ and γ from (3.2) and (3.4). If we consider the affinity as a smooth indicator of whether a pair of trajectories belong to same object ($a_{ij} \to 1$) or not ($a_{ij} \to 0$)

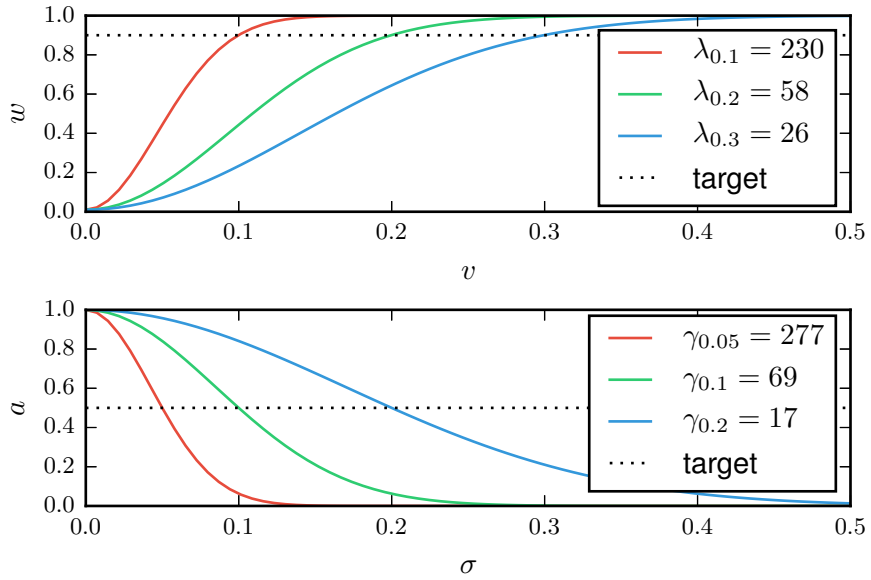

Figure 3.2: Top: parameter selection of λ_v for weights w based on target velocity v. Bottom: paramter selection of γ_σ for affinity a based on target standard deviation σ.

then equation (3.4) can also be seen as a *soft* threshold mapping standard deviation σ to affinity (see Figure 3.2 bottom). This means one could define γ as

$$\gamma_\sigma = -\frac{1}{\sigma^2} \log 0.5$$

depending on the standard deviation at which to switch between high and low affinity. Similiar one can specify λ as

$$\lambda_v = -\frac{1}{v^2} \log(1 - 0.9)$$

depending on the velocity v at which to consider a feature to be moving (see Figure 3.2 top).

3.2 Divisive k-Means Clustering

We have succesfully constructed an affinity matrix $\boldsymbol{A}$ with each entry $a_{ij} \in [0, 1] : \forall i \neq j \in [1, N]$ and $a_{ii} = 1$ describing the similarity between a pair of feature trajectories. We now seek to assign labels $l_i \in \{1, 2, ..., K\}$ to each trajectory where each label represents the hypothesis of a single object. In other words we want to cluster our trajectories based on an affinity matrix. The first problem we face is to determine K, that is finding the total number of clusters within the data. Although this is a very general problem and literature offers several approaches to solve this, it is not trivial.

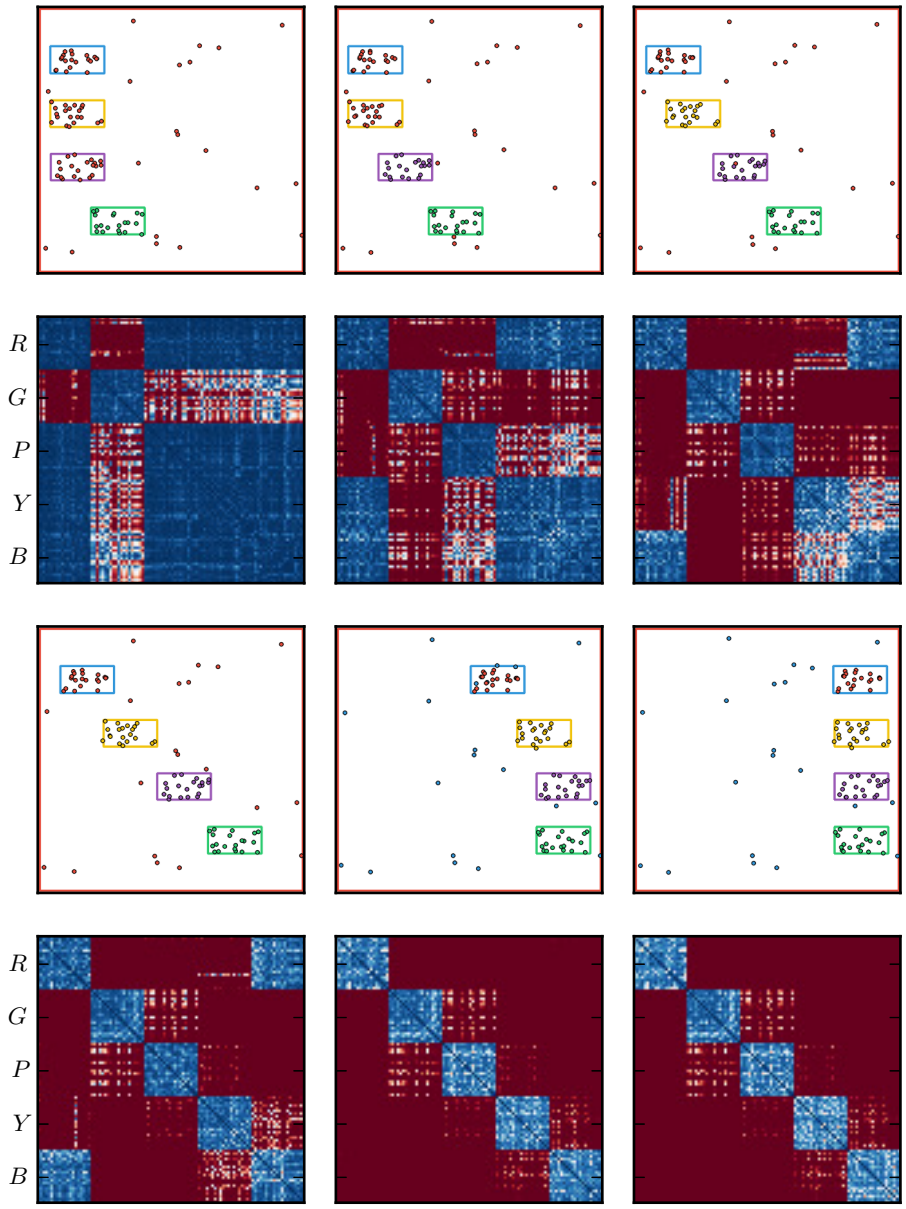

Figure 3.3: Featuring a simulated scene of 5 objects $\{R, G, P, Y, B\}$ (as red, green, purple, yellow, blue) where G, P, Y, B are objects moving from left to right with same constant speed but with different starting times. R is the stationary background. Each object yields 20 trajectories. The first and third row show the scene at $t = \{1, 2, 3, 4, 7, 9\}$, the second and fourth row presents the corresponding affinity matrix (blue $\rightarrow$ 1, red $\rightarrow$ 0) of trajectories sorted by objects.

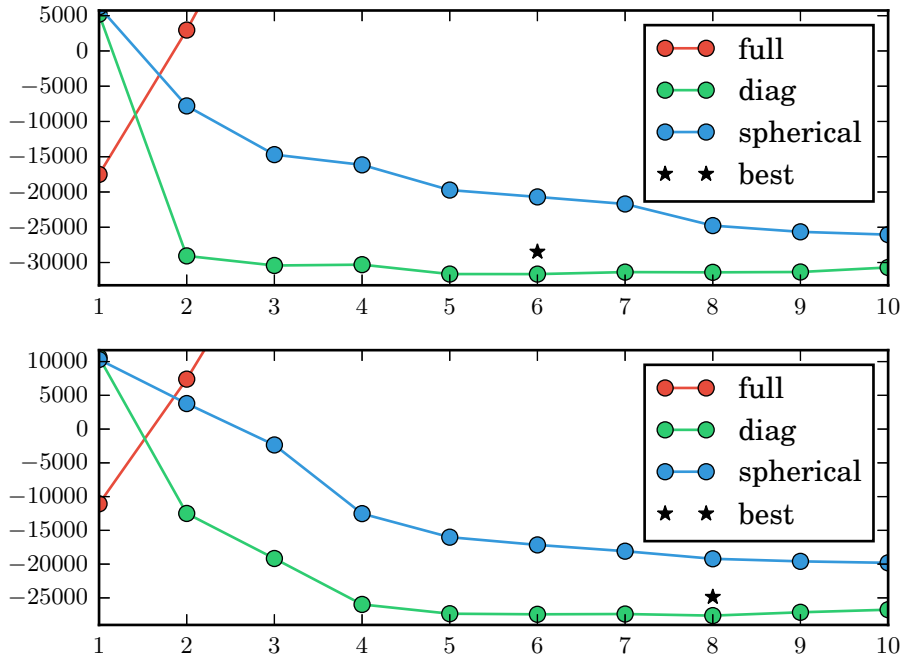

Figure 3.4: Bayesian information criterion (y-axis) for Gaussian Mixture Models with different types of parametrizations for the covariance matrices and variing number of components (x-axis). Fitting was performed on the affinity matrix obtained from the scene shown in Figure 3.3 at time $t = 1$ (top) and $t = 3$ (bottom). The star denotes the minimal BIC and hence the suggested model parametrization.

One approach is to fit several different models to the data, evaluate the Bayesian information criterion (BIC) [11] and select the model where BIC is minimal. BIC uses the likelihood of the data under the selected model to estimate the quality of the fit. Since adding more parameters to the model can easily improve the likelihood, it also leads to overfitting. Therefore BIC uses a penalty for the number of parameters of a model. *Scikit-learn* [10] provids easy access to an implementation for fitting Gaussian Mixture Models to the data. Figure 3.4 shows the resulting BIC for different Gaussian Mixture Models depending on the type of parametrization for the covariance matrices (full, diagonal and spherical) and the number of components. The used data is from a simulation scenario shown in Figure 3.3 at two different points in time $t = 1$ (top) and $t = 3$ (bottom). Expected values for the number of clusters are $K = 2$ in the first case, and $K = 4$ in the later one. As we can see the minimum BIC suggests more clusters then expected.

Another approach comes from spectral clustering [15] and uses the eigengab heuristic to determine the number of clusters. Our affinity matrix $\boldsymbol{A}$ can be interpreted as the adjacency matrix of a fully connected graph, where each trajectory resembles a vertex in the graph and the entries a_{ij} are weights of

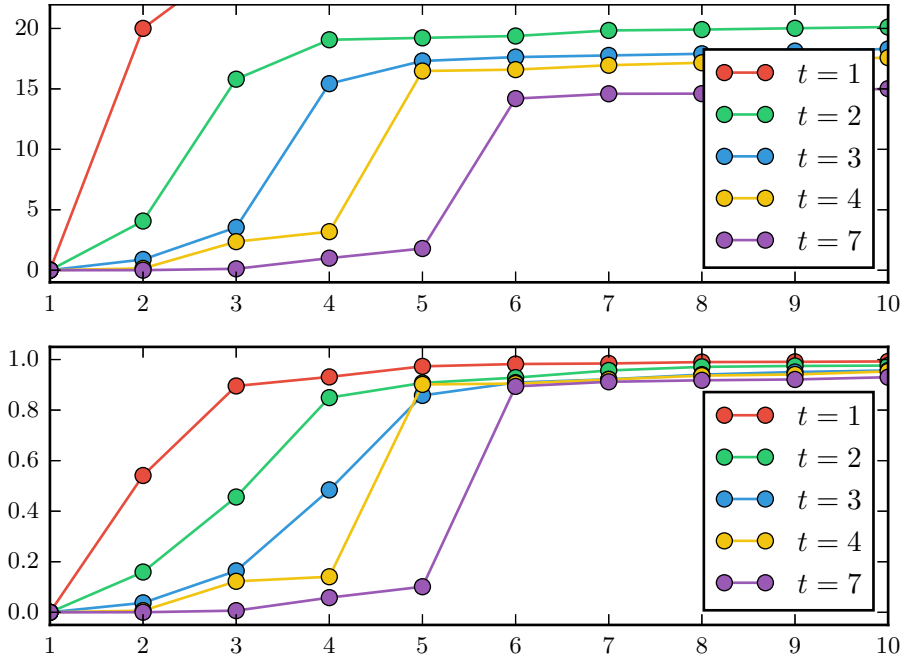

Figure 3.5: Shows the first 10 eigenvalues (y-axis) of the graph laplacian L (top) and the normalized laplacian L_{norm} (bottom) computed based on the affinity matrix obtained from the scene shown in Figure 3.3 at different points in time. The x-axis represents the indices of the corresponding eigenvalues.

an edge connecting trajectories i and j. One can compute the graph Laplacian as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is the degree matrix with diagonal elements

$$d_{ii} = \sum_{j=1}^{N} a_{ij}.$$

The graph Laplacian has several interesting properties that one can utilize. For further reading on this and spectral clustering the work of [15] is highly recommended. In particular what we are interested in is that

- the smallest eigenvalue of $\boldsymbol{L}$ is 0, the corresponding eigenvector is a constant one vector,
- $\boldsymbol{L}$ has N non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$,
- the number of K eigenvalues $\lambda_{1:K} = 0$ corresponds to the number of connected components in the graph.

The last statement leads to the so called eigengab heuristic which suggests that K should be chosen such that $\lambda_K - \lambda_{K+1}$ is large, hence $\lambda_{1:K} \ll \lambda_{K+1}$. Unfortunately for fully connected graphs as in our case this heuristic

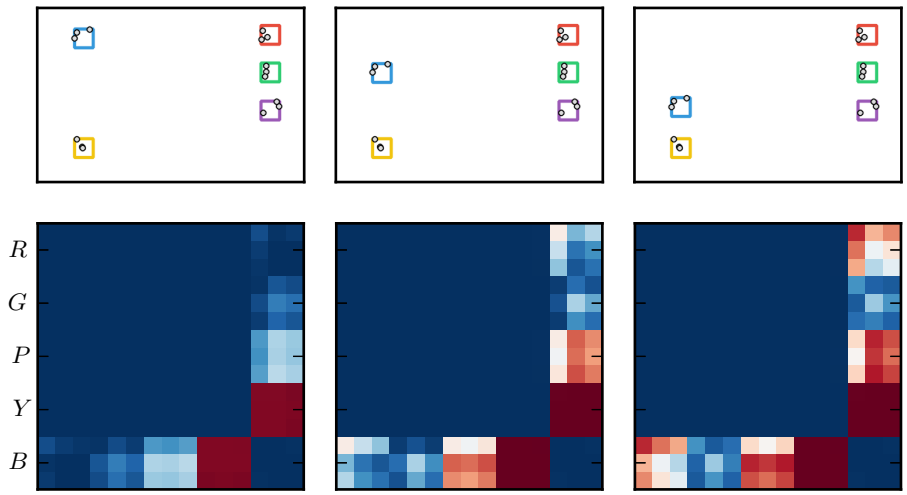

Figure 3.6: Featuring a simulated scene of 5 objects $\{R, G, P, Y, B\}$ (as red, green, purple, yellow, blue) where R, G, P, Y are stationary and B moves downwards (top row, from left to right). Each object yields 3 trajectories. The bottom row shows the affinity matrix (blue $\to$ 1, red $\to$ 0) of these trajectories sorted by objects (R: first 3 entries, G: next 3, ...).

seems to fail when the components are not clearly separated. Figure 3.5 shows the first 10 eigenvalues of the Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ (top) and the normalized Laplacian $\boldsymbol{L}_{\text{norm}} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$ (bottom) for the same scene used to investigate BIC (Figure 3.3) at $t = \{1, 2, 3, 4, 7\}$. The expected value for K at the given timestamps are $\{2, 3, 4, 5, 5\}$. As we can see the transition between consecutive eigenvalues is rather smooth in most cases and therefore it seems as there exists some connection between different objects that prevents a clear separation. In fact the variance criteria from the previous section 3.1 has an interesting feature that seems responsible for this.

To make things clear let us take a look at a different simulated scene in Figure 3.6. There are 5 objects (R, G, P, Y, B) and each yields 3 perfect trajectories without any noise. Only a single object B is moving downwards, the others do not move at all. One might suspect that, since only B moves and R, G, P, Y have the same stationary trajectory except for their location, the affinity of B to all others should be equally small. A closer look at the affinity matrix however reveals that only affinity between B and Y is small and is particularly high to G. This is very different from equal. Why is that?

If we think again in terms of rigid bodies then the motion of B could also indicate a rotation around G and having high affinities in this case is a good thing. This also shows that the chosen criteria is not entirely independent

to relative position and movement. If the motion vector and distance vector are parallel (B and Y) small distance variations lead to significent affinity changes. More specifically only the projected portion of the motion vector onto the distance vector affects the variance. This leads to the conclusion that different feature trajectories of two objects may have different affinities depending on where the features are located. If we see R, G, P, Y as one object, e.g. the background, than we need to take into account that some parts of this object may still have strong connections to other objects (G to B). A weak connection however is what separates it from others (Y to B).

The divisive k-means approach tries to solve this issue by splitting the graph into two similar parts and removing all edges connecting both clusters. This step is repeated for each cluster if there still exists a decent amount of dissimilarity within it. In general k-means minimizes

$$\min_{\boldsymbol{l}} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{3.6}$$

for the label vector $\boldsymbol{l} \in \{1, 2, ..., K\}^N$ and $\mathcal{C}_k = \{i | l_i = k\}$ as the set of cluster indices.

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \boldsymbol{x}_i$$

is the mean of all rows $\boldsymbol{x}_i = (a_{i,1:N})^T$ of matrix $\boldsymbol{A}$ assigned to the kth cluster. Hence equation (3.6) attempts to minimize the variance within all clusters. We stop further dividing clusters if the minimum affinity within the cluster after the edge removal step is greater than 0.4.

3.3 Label Propagation

Finally we need to combine the hypotheses obtained from clustering $\hat{\mathcal{C}}$ with previously observed hypotheses $\mathcal{C}$. To accomplish this we create a single correlation matrix $\boldsymbol{M} \in \mathbb{R}^{K \times K}$ with entries

$$m_{kl} = \sum_{i}^{N} p(c_k|x_i) p(\hat{c}_l|x_i) p(x_i)$$

where $p(c_k|x_i)$ denotes the probability of the ith trajectory belonging to cluster c_k, which is the currently assumed object hypothesis, $p(\hat{c}_l|x_i)$ the probability that x_i belongs to $\hat{c}_l$, which is the updated hypothesis obtained from clustering. $p(x_i)$ is assumed to be uniform, so $p(x_i) = N^{-1}$. When $p(c_k|x_i) = 1$ if $x_i \in c_k$ and else $p(c_k|x_i) = 0$, this basically counts the number of trajectories that are in c_k and $\hat{c}_l$. From $\boldsymbol{M}$ we K-times pick the indices k, l with the maximum value and set row and column $m_{k,1:K} = m_{1:K,l} = -1$ to prevent from picking there again. Every pick assigns all trajectories from cluster $\hat{c}_l$ to c_k.

Chapter 4

Experiments

This chapter investigates the theoretical capabilities of the proposed method by discussing the results based on simulation data. The remainder presents a real world application using the Lucas-Kanade-Tomasi Tracker as a source of feature trajectories from RGB-D data.

In order to evaluate clustering methods, [7] suggests several criterias. One of them is the normalized mutual information (NMI) which is defined as

$$\mathrm{NMI} = \frac{2I(\mathcal{C};\mathcal{G})}{H(\mathcal{C}) + H(\mathcal{G})}. \tag{4.1}$$

Let $\mathcal{G} = \{g_1, g_2, ..., g_L\}$ be the set of L clusters denoting the ground truth, where each g_l is a cluster of trajectories that form the true object hypothesis. Let $\mathcal{C} = \{c_1, c_2, ..., c_K\}$ be the set of K object hypothesis obtained from the proposed method. In general the mutual information $I(X;Y)$ of two random variables expresses the amount of dependence between both variables. If X and Y are independend and therefore $p(x,y) = p(x)p(y)$, then $I(X;Y) = 0$. In our case mutual information is computed as

$$I(\mathcal{C};\mathcal{G}) = \sum_{k=1}^{K}\sum_{l=1}^{L} p(c_k \cap g_l) \log \frac{p(c_k \cap g_l)}{p(c_k)p(g_l)}$$

where

$$p(c_k) = \sum_{i=1}^{N} p(c_k|x_i)p(x_i) = \frac{1}{N}|c_k|$$

is the probability of trajectories x_i being in the kth cluster (and $p(g_l)$ accordingly) and

$$p(c_k \cap g_l) = \sum_{i=1}^{N} p(c_k|x_i)p(g_l|x_i)p(x_i) = \frac{1}{N}|c_k \cap g_l|$$

is the probability of trajectories being in both clusters. The minimum of $I(\mathcal{C};\mathcal{G})$ is 0 if the hypotheses are random with respect to the expected clusters. The maximum mutual information is reached when the obtained hypothesis exactly match the expectation. However the maximum does not change when the perfect results are further subdivided into smaller clusters and therefore this value is also reached if every cluster consists of only a single trajectory ($K = N$). To fix this mutual information is normalized using the average entropy of the obtained and expected clusters $H(\mathcal{C}) + H(\mathcal{G}))/2$ with

$$H(\mathcal{C}) = -\sum_{k=1}^{K} p(c_k) \log p(c_k)$$

and similiar for $H(\mathcal{G})$. Since the entropy has its maximum at $K = N$ this normalization penalizes oversegmentation and ensures an upper bound of 1, hence $\mathrm{NMI}(\mathcal{C};\mathcal{G}) \in [0, 1]$.

The ground truth of simulation and the real world datasets is fixed for each trajectory. Every trajectory is assigned to exactly one true hypothesis independend of the time. This means even if the trajectories are similiar at a certain point in time (e.g., the background and an object that did not start to move, yet), the ground truth assumes different objects over the entire lifetime of both trajectories. Since we evaluate the NMI for every obtained update on the object hypotheses, we only consider trajectories in the ground truth that are currently active. Hence, the set we use for ground truth only contains trajectories we receive from clustering.

4.1 Simulation

In principal, the simulation can be seen just as another source of feature trajectories that we send as input to our processing pipeline. The major difference however is that we know the exact properties of each trajectory and what the corresponding true object hypothesis is. This not only enables us to quickly create new scenarios to review problems that occured with a real application, it is also a great way to qualitatively assess the effect of parameter changes and algorithm adjustments. In the following we will discuss the effect of the parameter γ of equation (3.4) with respect to noise of the trajectories and the effect of using weights based on velocities. In a last scenario we show that the neither translation nor rotation is a problem and how suddenly spawning new trajectories affects the results.

In order to create a new scenario we first define a set of objects $\mathcal{O} = \{R, G, P, Y, B, ...\}$, where each object defines a rectangular region. A remark on the notation: letters R, G, P, Y, B are chosen such that they denote the color of the corresponding rectangles in the images (R: red, G: green, P: purple, Y: yellow, B: blue). For each object o we also define a set of

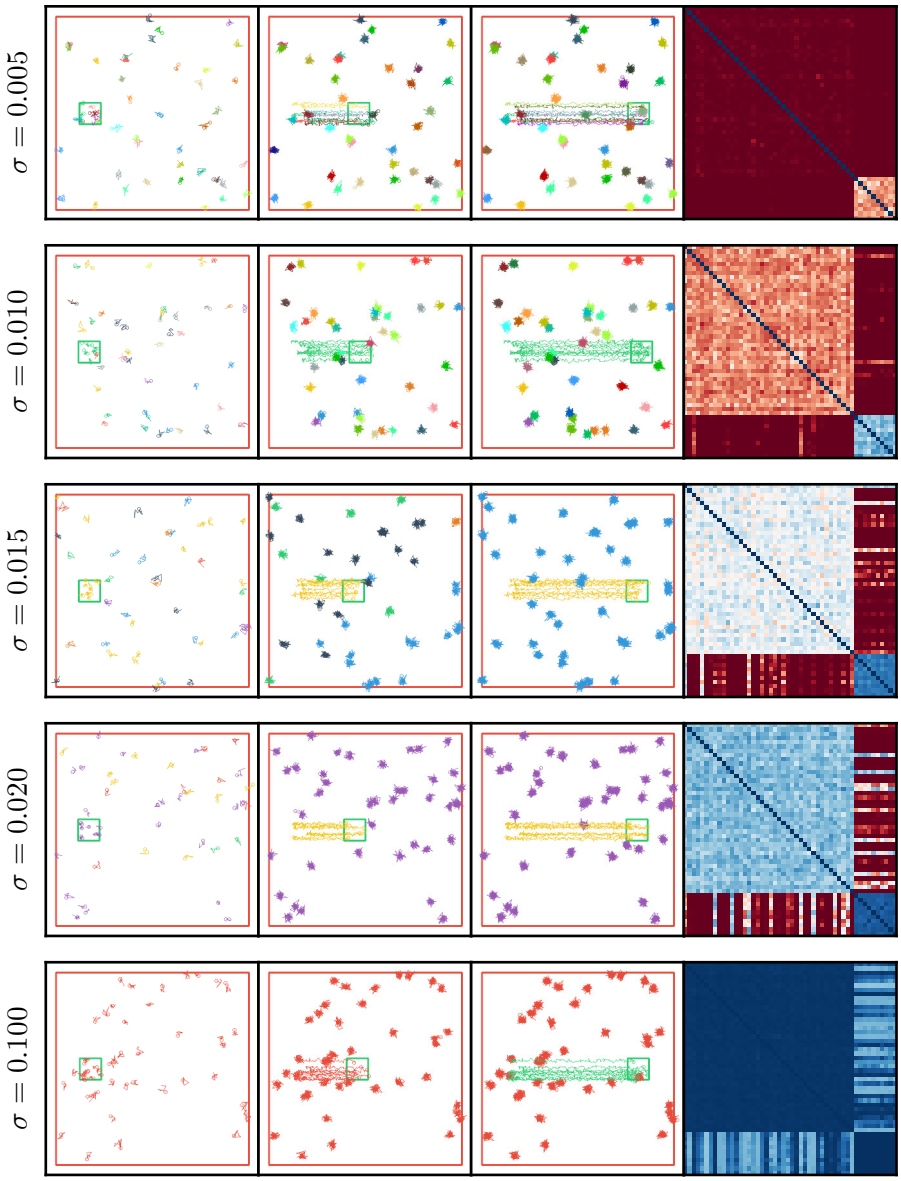

Figure 4.1: Shows the effect of choosing γ based on different values for σ. The scene consists of two objects $\mathcal{O} = \{R, G\}$, where R resembles the background with 40 stationary features and Gaussian noise $\sigma_R = 0.01$. G is an object moving from left to right with 10 features and $\sigma_G = 0.005$. The first three pictures of each row present a snapshot of the scene at time $t = \{1, 5, 10\}$. The colors of the trajectories indicate the resulting hypotheses. The last column of each row shows the affinity matrix corresponding to the scene in the middle picture ($t = 5$)

key-states $\{\boldsymbol{s}_{o,1}, \boldsymbol{s}_{o,2}, ...\boldsymbol{s}_{o,M}\}$ where $\boldsymbol{s}_{o,1} = (p_x, p_y, \phi)^T$ is the position and orientation at the beginning of the simulation ($t = 0$) and $\boldsymbol{s}_{o,M}$ at the end ($t = T$). The simulation runs with a frequency of 15Hz and interpolates linearly between two consecutive key-states to obtain the current state $\boldsymbol{s}_{o,t}$ of each object.

The initial state $\boldsymbol{s}_{o,1}$ and the specified rectangular region are used to define the initial states $\boldsymbol{x}_{i,1}$ of the features for each object. All new features are always spawned randomly relative to $\boldsymbol{s}_{o,1}$ and within the region. With each simulation update all feature states will be transformed based on the current state $\boldsymbol{s}_{o,t}$ of their corresponding object. These transformed states create new observations for each feature as

$$\boldsymbol{x}_{i,t} = \boldsymbol{T}_{\boldsymbol{s}_{o,t}} \boldsymbol{x}_{i,1} + \epsilon$$

with $\epsilon \sim N(0, \sigma_o)$ being normal distributed noise.

The effect of γ

At the end of section 3.1 we have seen that a more convenient way of choosing γ is to set it with respect to a desired standard deviation σ

$$\gamma_\sigma = -\frac{1}{\sigma^2} \log 0.5$$

at which we want to switch between high and low affinities. Figure 4.1 shows the results for different values of σ on a simple scenario that consists of only two objects $\mathcal{O} = \{R, G\}$, where R resembles the background with 40 stationary features and Gaussian noise $\sigma_R = 0.01$. G is an object moving from left to right with 10 features and $\sigma_G = 0.005$. All features are spawned initially and do not die. Each row of Figure 4.1 shows the scene at timestamps $t = 1, 5, 10$. The color of the trajectories denotes the estimated object hypotheses where the same color corresponds to the same hypothesis. The last image in each row shows the affinity matrix at $t = 5$ (hence, corresponding to middle image of the three scene snapshots).

From the first row we see that if σ is too small, each trajectory is assigned to a separate object hypothesis and we have a classical case of oversegmentation. In the second row the trajectories of G are already placed in the correct hypothesis but the background is still incorrect. In the middle row it becomes better but it still takes quite some observations until all trajectories in the background are assigned to the same hypothesis. Row four shows almost perfect results except at the very beginning and in the last row the object G has to move almost entirely across the scene to be separated from the background. A plot of the NMI in Figure 4.2 presents these results more precisesly and shows exactly how long it takes to reach a perfect solution if possible.

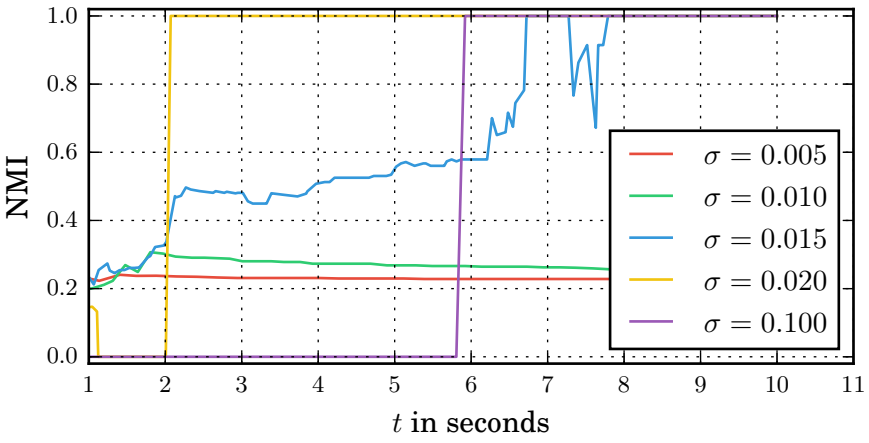

Figure 4.2: Normalized mutual information over time corresponding to the scene from Figure 4.1. Shows the results of the effect for choosing γ based on different values for σ.

The best choice for γ in this scenario is with respect to $\sigma = 0.02$. In fact this result is not very surprising if we take another look at the decomposition of the variance in equation (3.1) as

$$\mathrm{Var}(\hat{X} - X) = \mathrm{Var}(\hat{X}) + \mathrm{Var}(X) - 2\mathrm{Cov}(\hat{X}, X)$$

which becomes smaller when $\mathrm{Cov}(\hat{X}, X) > 0$. Since a positive covariance between two random variables indicates similiar behavior and the opposite for negative values, defining the switch between high and low affinity at the point where $\mathrm{Cov}(\hat{X}, X) = 0$ makes perfectly sense. In this case we get

$$\mathrm{Var}(\hat{X} - X) = \mathrm{Var}(\hat{X}) + \mathrm{Var}(X) = \sigma_{\hat{X}}^2 + \sigma_X^2$$

which suggests to redefine the choice for γ as

$$\gamma_{ij} = -\frac{1}{\sigma_i^2 + \sigma_j^2} \log 0.5$$

if the values for σ_i and σ_j of the ith and jth trajectory are known in advance. Yet, adjusting γ depending on the trajectory pair that is being compared has not been investigated any further and we will leave this topic to future work. Hence we set

$$\sigma = 2\max\{\sigma_i\}$$

for all known standard deviations. The obvious limitation to this is the assumption that the error of a trajectory can be approximated by Gaussian noise.

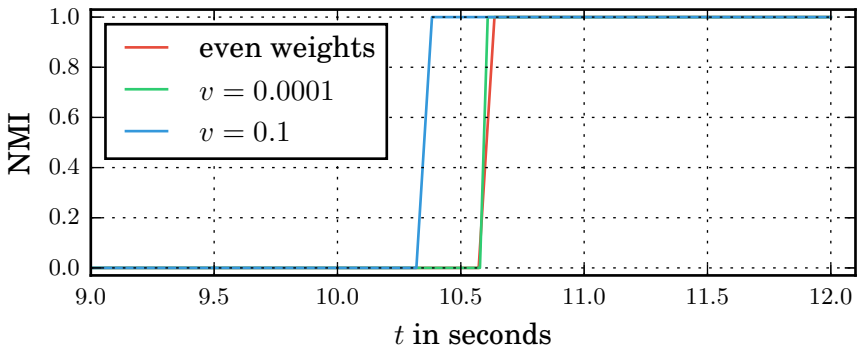

Figure 4.3: Normalized mutual information over time for investigating the effect of weights based on velocity.

Assigning Weights Based on Feature Velocity

The idea of using weights when computing the distance variance is to add some dynamic to the process when an object starts to move after a longer period at rest. To evaluate the effect of this we slight change the previously used scenario from Figure 4.1 so that object G does not move for 10 seconds and then moves to the other side of the scene with constant velocity $v_G = 0.3$. Since great noise would also put high weights on the background trajectories we set $\sigma_R = \sigma_G = 0.0005$. Figure 4.3 shows the NMI of this setup for placing equal weights on all distances, setting λ to very small velocity and for a good choice of $v = 0.1$. Because of how the weighting scheme is designed (3.2), choosing a small threshold velocity v is not very different from setting equal weights. Selecting a good value for v provides the expected increased dynamic although the effect is quite small.

Spawning New Trajectories

In the last scenario we will take a closer look at what happens if we regularly initialize new trajectories. This investigation is motivated by the fact that for real applications it is quite common for trajectories to die due to occlusion or other problems causing the tracker to loose the feature. To compensate for this loss it is practical to regularly search for new features and create new trajectories.

Figure 4.4 depicts the test scenario of three objects $\mathcal{O} = \{R, G, P\}$, where again R is the set of 20 background features with $\sigma_R = 0.005$. G is an object moving from right to left, containing 10 features with $\sigma_G = 0.005$ and spinning around its center in changing directions (CCW $\rightarrow$ CW $\rightarrow$ CCW). P resembles an elongated rectangle with 10 features and $\sigma_P = 0.001$ that slowly rotates by 90 degrees. Every 2 seconds, 20% of the existing features of each object are randomly removed and replaced by new ones. We set γ with respect to $\sigma = 0.01$ and use even weights.

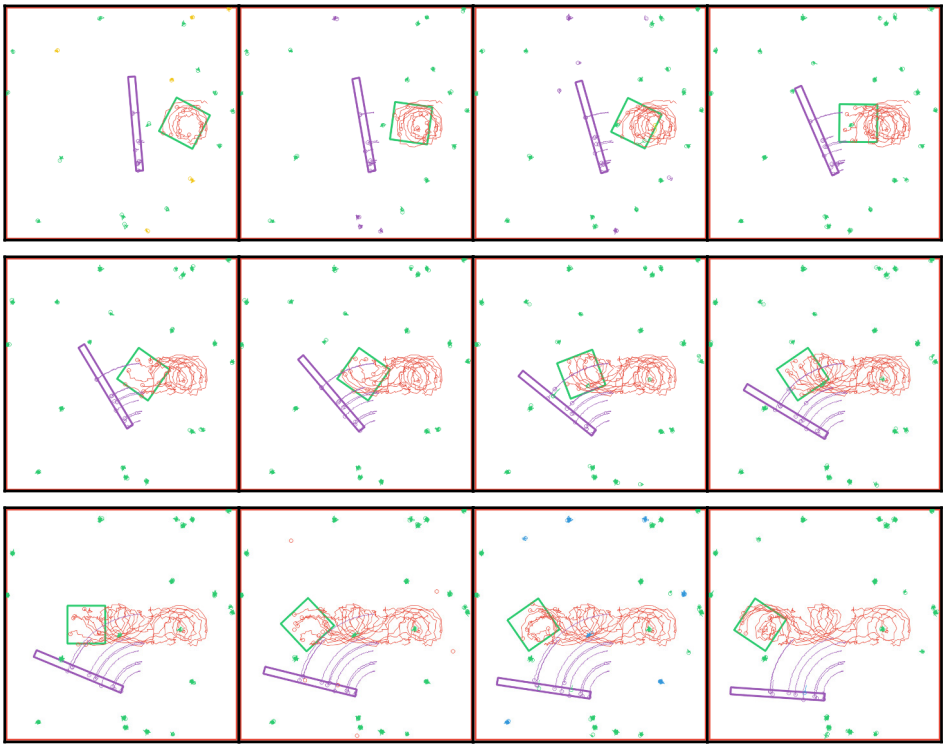

Figure 4.4: Shows a simulation scenario of three objects $\mathcal{O} = \{R, G, P\}$, where R is the set of 20 background features with $\sigma_R = 0.005$. G is an object moving from right to left, containing 10 features with $\sigma_G = 0.005$ and spinning around its center. P rotates to the left by 90 degrees, contains 10 features with $\sigma_P = 0.001$. Every 2 seconds, 20% of the existing features of each object are randomly replaced by new ones. Colors of the tracks indicate the estimated hypotheses.

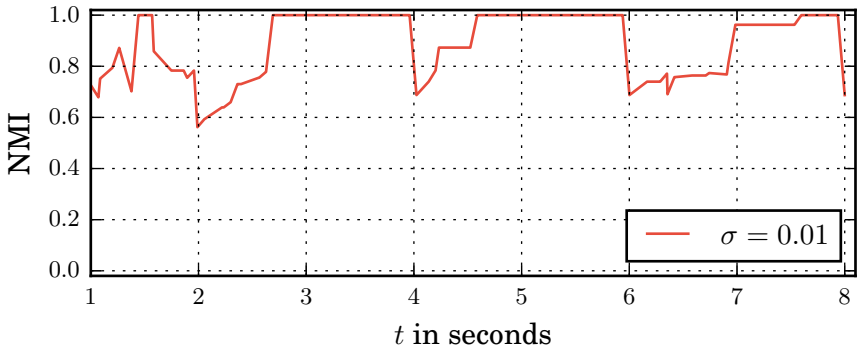

Figure 4.5: Normalized mutual information over time corresponding to the scene shown in Figure 4.4. The drops of the NMI are caused when new features are initialized.

In general the results as shown in Figure 4.4 look satisfying and most of the trajectories are assigned to the correct hypothesis. Another look at Figure 4.5 however reveals a drop of the NMI at $t = \{2, 4, 6\}$, which is exactly when new trajectories are spawned. This is because new trajectories have high affinities to all existing ones since their common observations are very small. When a split of data is performed by the clustering step, these features tend to be placed within the greater of both clusters and therefore usually get assigned to the background first. They get reassigned to the correct hypothesis as soon as more observations are available. Hence, the NMI in Figure 4.5 slowly moves back up to 1.

An approach to solve this problem could incorporate a prior belief based on the spatial distance to existing trajectories. This would result in new trajectories to takeover the hypothesis of nearby features as long as the number of observations is too small. This is another aspect we will leave to future work.

4.2 Example Application

Let us now turn to a more practical application. For this purpose we fall back on the publicly available implementation of the popular Lucas-Kanade feature tracker [1] provided by the *OpenCV Library* [2]. Shi-Tomasi corners [12] will be used as features in the color image. A very simple way to extend this tracking method by additional spatial information is to use an RGB-D sensor such as Microsoft Kinect and project the positions of the tracked features from the color image into Euclidean space. This is very easy since color and depth images are already aligned on hardware side which means for every color pixel we also have a 3D point.

A common way to detect tracking errors caused by occlusion is to perform a consistency check of the forward and backward flow for each feature [14]. Let $x_{i,t}$ be the position in pixel coordinates of the ith feature in the current image. We obtain the position $x_{i,t+1}$ in the new image by applying the Lucas-Kanade flow estimation between images at t and $t+1$. We can now also apply this method backwards by tracking $x_{i,t+1}$ from image $t+1$ to t in order to obtain $\hat{x}_{i,t}$. Idealy this would result in the same position. As suggested by [14] we stop tracking if

$$\|w_i + \hat{w}_i\|^2 < 0.5(\|w_i\|^2 + \|\hat{w}_i\|^2) + 0.5$$

where $w_i = x_{i,t+1} - x_{i,t}$ denotes the forward motion and $\hat{w}_i = \hat{x}_{i,t} - x_{i,t}$ the backward motion of the ith feature. Another heuristic used to stop tracking is to restrict the length of the motion vector in Euclidean space. This is necessary because color and depth image are not perfectly aligned and points may suddenly jump between the object and the background in depth image. To compensate for the lost features we create new features

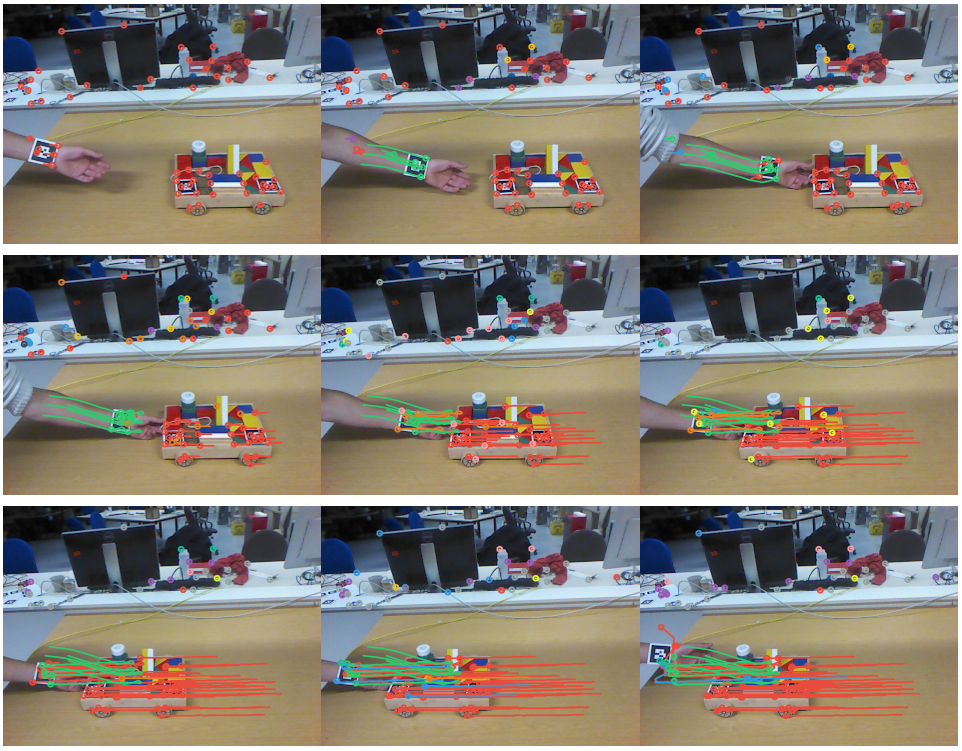

Figure 4.6: Shows the estimated hypotheses at different points in time for the toy truck scene. Trajectories are obtained using the KLT tracker.

after certain number of frames at the 100 best positions in the current image. In order to prevent spawning new features at same locations every time, we subdivide the image into equal sized grid cells and only keep the newly created features if they fall into an empty cell.

This application was evaluated on three different scenes. Figure 4.6 shows how a toy truck is pulled from the right side of the image to left. New features are initialized after every 15th frame. Figure 4.9 provides the corresponding NMI results. Notice that trajectories of the hand and the truck are assigned to different hypotheses (red and green) if they are observed for a long time. Trajectories that are spawned later in time and therefore lack the observations to discriminate them from other objects are assigned to the bigger cluster. For this reason some parts of the hand show red trajectories instead of green ones. We can observe another source of error in this scene: points in the background tend to be separated into small clusters. This is because the measurement noise of the Kinect raises rapidly with increasing distance.

Figure 4.7 shows a similar scene but this time with two objects moving at the same time with dissimilar motion. The truck moves from right to left while the tower is pushed from the left to the right at the same time. In

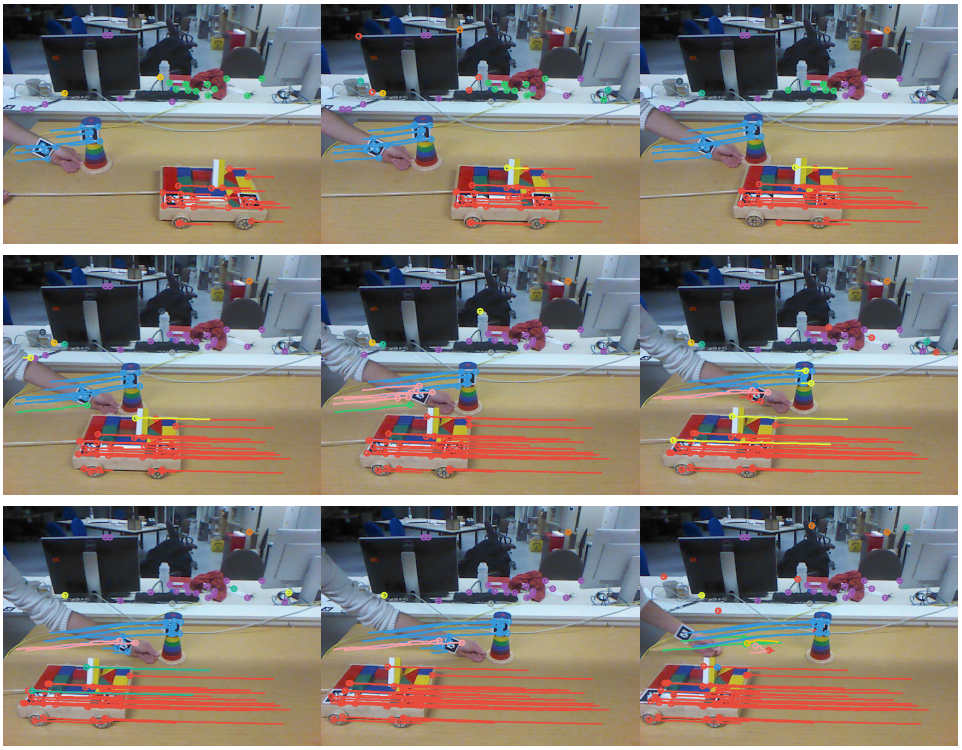

Figure 4.7: Shows the estimated hypotheses at different points in time for the scene of a toy truck and a tower moving in different directions. Trajectories are obtained using the KLT tracker.

this case we initialized new features with only half the rate at every 30th frame (hence, every second). The results are similar to the previous scene. Tower and truck are nicely separated (blue and red). Notice the two yellow trajectories on the truck whos feature is located close to the border. While motion looks good in the color image, their corresponding 3D point however sometimes jumps between table and truck. Such outliers are currently not handled well. Figure 4.10 gives the results of the NMI evaluation.

In the last scene as shown in Figure 4.8 and the corresponding NMI in Figure 4.11 we see a shelve with drawers and doors. At first the lower door is opened and closed showing rotational motion. Then the drawer is pulled out and back in. In this scene special attention should be given to the lower door. The hypothesis actually changes when the door is fully open. This is because at this point all features are lost and there is no association to the trajectories for closing the door again.

Figure 4.8: Shows the estimated hypotheses at different points in time for a scene with translational and rotational motion. Trajectories are obtained using the KLT tracker.

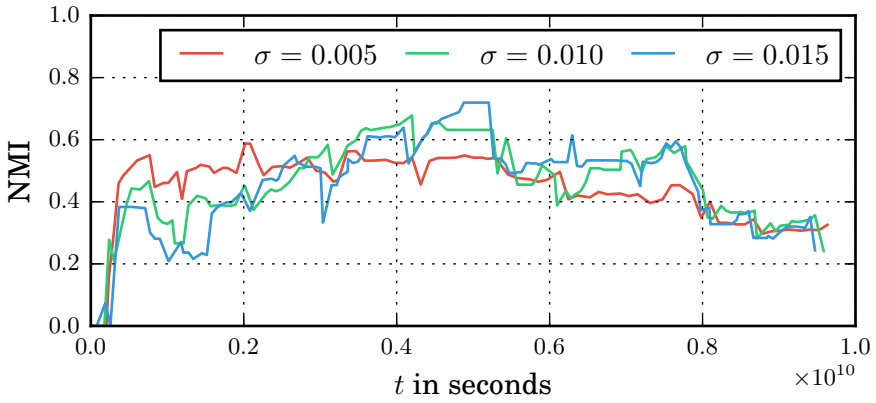

Figure 4.9: Normalized mutual information over time for the toy truck scene shown in Figure 4.6 for different values of σ.

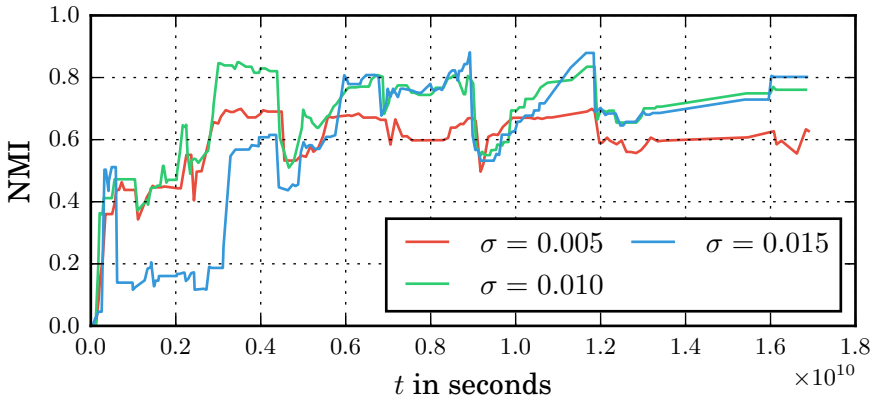

Figure 4.10: Normalized mutual information over time for the toy truck and tower scene shown in Figure 4.7 for different values of σ.

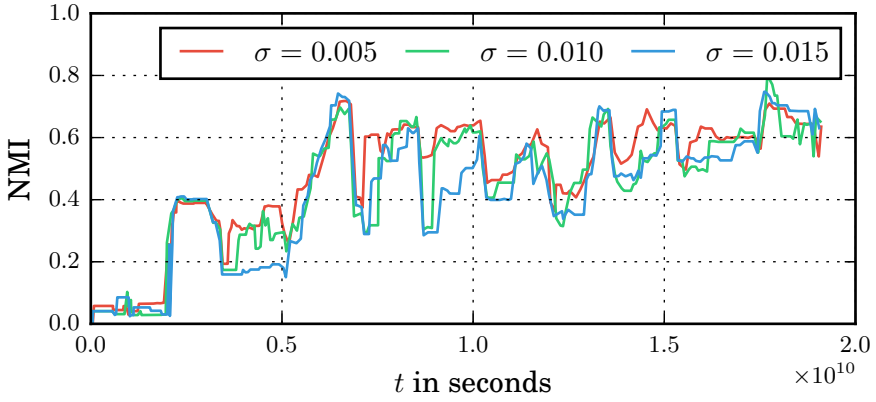

Figure 4.11: Normalized mutual information over time for the drawer scene shown in Figure 4.8 for different values of σ.

Chapter 5

Conclusion

In this thesis we investigated an approach to fuse trajectories from different feature trackers and to create hypotheses about objects under the constraint of rigid body motion. This could eventually become an important piece of a system that leads to a more general way of how perception is addressed in robotic applications.

The presented method shows promising capabilities to identify objects of translational as well as rotational motion for feature trajectories with Gaussian noise. This was verified by various experiments using data from simulation. The results of an example application using the KLT tracker as a single source for trajectories however suggsets that Gaussian noise approximation may not suffice for real trackers. Despite the heuristics that were applied to reduce the number of tracking errors, the simple nature of how we extended the KLT tracker from color image to the depth image still leads to outliers in the trajectories. One way to deal with small outliers is to reduce parameter γ at the price of requiring larger motion to discriminate between objects. This however is not a satisfying solution and therefore should be placed on top of the list of potential improvements.

Another important issue that should be addressed in future work is to improve how newly initialized features are handled. A good starting point would be to incorporate a spatial prior so that new trajectories preferably takeover the hypotheses of their close neighbors. Unfortunately the actual fusion of different types of trackers has only been covered in the conceptual phase. For this reason future work should also be dedicated to an evaluation of using multiple trackers in order to verify results. One potential obstacle might require to address different levels of trajectory noise. If one could guarantee gaussian noise of the feature states, choosing the value for γ depending on the tracker types of the compared pair of trajectories could make a difference.

Bibliography

[1] J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5:1–10, 2001.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *Computer Vision–ECCV 2010*, pages 282–295. Springer, 2010.

[4] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[5] J. Kenney, T. Buckley, and O. Brock. Interactive segmentation for manipulation in unstructured environments. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1377–1382. IEEE, 2009.

[6] K. Koffka. Principles of gestalt psychology. 1935.

[7] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[8] R. M. Martín and O. Brock. *Online Interactive Perception of Articulated Objects with Multi-Level Recursive Estimation Based on Task-Specific Priors*. 2014.

[9] P. Ochs and T. Brox. *Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions*, page 1583–1590. IEEE, 2011.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[12] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[13] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard. Vision-based detection for learning articulation models of cabinet doors and drawers in household environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 362–368. IEEE, 2010.

[14] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Computer Vision–ECCV 2010*, pages 438–451. Springer, 2010.

[15] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[16] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Computer Vision–ECCV 2006*, pages 94–106. Springer, 2006.

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.
