BACHELOR PROJECT REPORT: MULTIPLE OBJECT TRACKING WITH POINT PATTERNS

Simon Giebenhain[*]
Department of Computer Science
University of Konstanz
simon.giebenhain@uni-konstanz.de

January 18, 2020

ABSTRACT

Multiple Object Tracking (MOT) has been a subject of research for many years. This paper introduces a new kind of MOT problem, working on unlabeled point detections made of each objects unique point pattern. Next to position, also the orientation of objects becomes of interest. The problem was inspired by trying to track birds based on three-dimensional detections of markers on the their backs. Having access to occlusion free three-dimensional detections makes assigning detections to objects easy. Therefore this task is handled with a simple MOT framework, which can be used with any Single Object Tracker (SOT). A Kalman Filter (KF) based SOT with a few additions tailored for the specific MOT problem is introduced. Experiments show that these additions to the regular KF greatly improve MOT performance. Additionally a neural network based approach is described, which performs worse than the KF approach, but shows promising noise resistance.

1 Introduction

Multiple Object Tracking(MOT) refers to the task of identifying and maintaining some desired properties of objects of interest over time. This paper introduces two Multiple Object Tracking methods which work on unlabeled point detections. Each object is equipped with a point pattern consisting of multiple single markers. The markers are what is being detected. In this paper the desired properties are location and orientation. The work is tailored towards MOT of birds in three-dimensional space, with patterns being defined by the position of four markers. A more precise formulation of the problem is given in section 3.
Usually MOT is performed on video data as the most common use-cases are sports tracking, tracking of pedestrians and in autonomous driving. While this paper technically deals with MOT, this setting is quite different, since precise three-dimensional point detections instead of videos are used, alleviating occlusion problems. Together with a frame rate of 30 frames per second the detection-to-track-assignment, which is the main challenge of the classical MOT problem, becomes easy. The difficulty shifts towards accurate Single Object Tracking (SOT), especially the precise tracking of orientation remains challenging.
Therefore, I propose two approaches for SOT, which can be embedded in a simple MOT-framework to track multiple objects at once. This framework is described in section 4.
Section 4.1 introduces the first SOT approach, which is based on a Kalman Filter (KF). The fundamental difficulty of this SOT approach is to estimate the assignment of the detections to the markers, as knowing these correspondences is necessary for the KF to work.
Section 4.2 introduces a recurrent neural network based approach for SOT. I formulate a general architecture, in which an embedding neural network transforms the input of each time step and feeds it into an Long-Short-Term-Memory network (LSTM). Multiple possible encoder architectures are considered, as this is the most crucial part of the network. The detections at each time step are point clouds of varying number of points, and therefore the encoder has to be permutation invariant as well as resistant to noise, missing detections and false positive detections.

[*]The work was done as a Bachelors Project supervised by Prof. Bastian Goldlücke. My code and videos displaying my tracking results can be accessed via GitHub

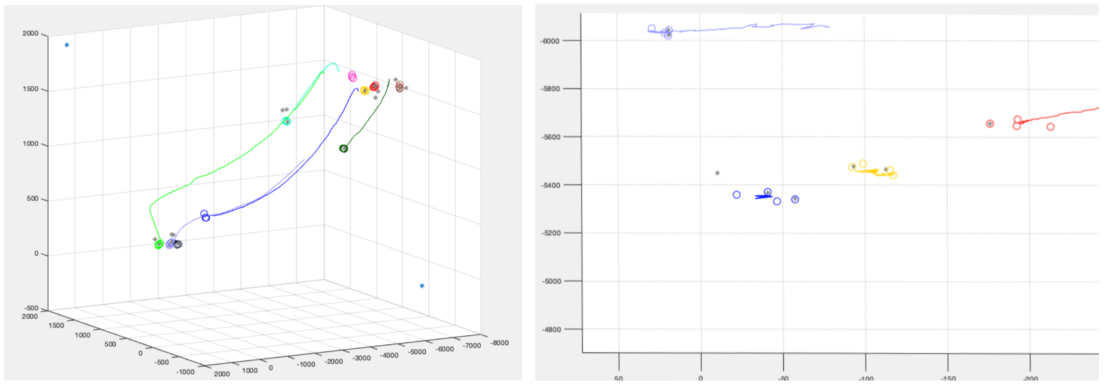

(a) Flying birds (b) Detailed top-down-view

Figure 1: **Detections and tracking results** - Gray stars represent detections. Colorful circles show the points where markers of tracked birds are expected, based on tracked position and orientation. The lines show past movement of birds. (a) Shows some birds flying towards the ground, while others are stationary. (b) Gives a detailed view of four birds. One can see a false positive detection next to the dark blue bird. Lots of detections are missing, e.g. only one of the four markers of the red bird are currently visible.

I validate my design choices for both approaches by running experiments on data with varying degrees of noise, missing detections and false positive detections. In order to conduct these experiments, I had to generate data with known ground truth. The data generation process is described in section 5. A quantitative comparison of both approaches and a short discussion is given in section 5.3. The experiments showed that the KF approach currently achieves superior tracking performance and is more flexible than the neural network, as new data could require retraining the complete neural network.
Multiple ideas for improving both methods are presented in section 6.

1.1 Data Capturing Setup

This section explains how the data is being captured. Images of the data are shown in figure 1. The data was captured inside of a large room with a 2D array of infrared cameras filming downwards from above. Due to a multitude of cameras the detections are precise in three-dimensional space and occlusion does not occur. Each bird is equipped with a tiny backpack which has space for 4 infrared reflecting markers, positioned in a unique pattern. There can be small elevations on the backpack, meaning that the specified pattern can be three-dimensional instead of just lying in one plane. Using these patterns enables to predict the orientation and position of the birds in three-dimensional space. All individual markers are the same, meaning that it is impossible to distinguish between single detections. Although the room was painted such that a minimal amount of infrared light gets reflected from anything else than the infrared markers, there is still a substantial amount of false positive detections. Similarly markers are often not detected by the cameras. These kinds of noise pose the main problem to be addressed in this paper.

2 Related Work

Most of the MOT research focuses on tracking pedestrians or vehicles, as surveillance and autonomous vehicles are two driving industry applications for MOT. These approaches usually apply a tracking-by-detection paradigm, where an object detector locates objects of interest in each frame and the MOT algorithm associates these detections to the maintained tracks. This procedure is called detection-to-track-assignment. This way the algorithm can focus on the information nearby the detected objects and discard seemingly irrelevant information. The detection-to-track-assignment is the most crucial point of these approaches and commonly performed with visual features drawn from the detected bounding boxes. [Leal-Taixé et al., 2017] found that the most defining feature of the best performing models on the MOT15 benchmark ([Leal-Taixé et al., 2015]) and MOT16 benchmark ([Milan et al., 2016a]) is a good affinity measure using appearance features.

[Milan et al., 2016b] present a very general, recurrent neural network based model for MOT, which is trained end-to-end and does not use any information besides the location. They argue that applying deep learning to MOT is far from trivial. In MOT the tracked states are continuous, whereas the detection-to-track-assignment is a discrete assignment problem.

The number of tracked objects is arbitrary and death and birth of tracks have to be handled. These circumstances make it hard to design a architecture which can handle this variety of data. Their results reflects this reasoning, as the version of their model which uses an LSTM architecture for the detection-to-track-assignment performs slightly worse than the version using the Hungarian Algorithm (HA) for the detection-to-track-assignment. For this reason I choose to use the HA.

The work of [Weng and Kitani, 2019] uses a regular KF for tracking objects in three-dimensional space, where the state is composed of position, velocity, heading angle, and lengths describing the size of the bounding box of an object. They achieve state-of-the-art-results on the 3d MOT challenge on the KITTI dataset ([Geiger et al., 2012]), while being the fastest method with a large margin among the top algorithms. Their input are the detected bounding boxes from [Shi et al., 2018], which achieves the state-of-the-art results on the the 3d object detection challenge from the KITTI dataset, despite only using LiDAR point clouds and no RGB images at all. Using these state-of-the-art detections might explain their superiority in the 3d MOT challenge. Their work supports the choice of following a KF approach to tackle the problem at hand.

Differences to regular MOT The MOT variant discussed in this paper naturally follows a tracking-by-detection scheme, as 3d detections are the only available source of information. However the task at hand differs substantially from the regular MOT setting. When tracking pedestrians the scenes are usually very crowded and troubled by occlusion. Then an affinity model relying on visual appearance is crucial. The point detections in this work have high precision and are three-dimensional, which resolves occlusion problems. Therefore the detection-to-track-assignment is relatively easy most of the time and the MOT problem can be tackled by multiple SOTs and the HA for assigning the detections to the tracks.
However even a perfect detection-to-track-assignment does not guarantee a good tracking performance, as multiple detections are assigned to each object, without knowing which detection corresponds to which marker of the pattern. In order to estimate the position and orientation correctly, an exact detection-to-marker-assignment is required.
The task is complicated further by frequent missing detections and detections, which do not correspond to one of the markers, so called false positive detections. Missing detections especially become a problem when less than two detections are present for multiple frames, as no unique orientation can be determined. Not identifying a false positive detection as such, can potentially disrupt the orientation estimation. Therefore the applied SOT method has to be robust against such noise. Additionally, a strong motion model and using temporal information effectively seem to be crucial.

3 Problem Formulation and Notation

This section introduces the notation used throughout the rest of the paper while stating the problem mathematically. The task is to estimate the position and orientation at every time step for every object, based on a sequence of noisy detections of the unique point patterns over time.

In this formulation the patterns of the occurring birds have to be known beforehand. Let N be the number of known patterns, and let N_t be the number of present objects in frame t.
Each pattern is made up out of K markers. A pattern defines the arrangement of the K markers, which are d-dimensional points, in space. The pattern of the n-th object is denoted by

$$m^{(n)} = \begin{pmatrix} m_1^{(n)} \\ \vdots \\ m_K^{(n)} \end{pmatrix} \in \mathbb{R}^{Kxd}, \text{ where the } k\text{-th row holds the } k\text{-th marker }, m_k^{(n)} \in \mathbb{R}^d$$

Let T be the number of total time steps, then the input is a sequence of detected point clouds denoted by $\left(\mathcal{X}^{(t)}\right)_{t=1}^{T}$, where $\mathcal{X}^{(t)} \subset \mathbb{R}^d$. It is important to note here that $\mathcal{X}^{(t)}$ is a set of d-dimensional points, meaning that we have no prior information which detection belongs to which marker of which object. While $\mathcal{X}^{(t)}$ conceptually is a set, the detections are stored in such a way that there is an arbitrary ordering to them. I will reference the corresponding but ordered detections with $X^{(t)}$.

For each object the values to estimate are the

$$\text{positions } \left(p_1^{(t)}\right)_{t=1}^{T}, \ldots, \left(p_N^{(t)}\right)_{t=1}^{T} \text{ and rotations } \left(r_1^{(t)}\right)_{t=1}^{T}, \ldots, \left(r_N^{(t)}\right)_{t=1}^{T}$$

For $d = 3$ there are many common way to represent rotations. Next to the canonical representation as a rotation matrix, there are for example Euler angles and quaternions. Euler angles are a minimal representation since it only requires 3 parameters. However, they exhibit singularities, which correspond to a gimbal lock configuration, where large changes in the Euler angles are required to represent small changes in the actual rotation, [Hertzberg et al., 2011]. Therefore, Euler angles are generally considered inferior to quaternions in controlling and tracking tasks.

The task is to come up with the

$$\text{position estimates } \left(\hat{p}_1^{(t)}\right)_{t=1}^{T}, \ldots, \left(\hat{p}_N^{(t)}\right)_{t=1}^{T} \text{ and rotation estimates } \left(\hat{r}_1^{(t)}\right)_{t=1}^{T}, \ldots, \left(\hat{r}_N^{(t)}\right)_{t=1}^{T}$$

which minimize some desired error. The Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) metrics introduced in [Bernardin and Stiefelhagen, 2008] could be of interest. The MOTP metric has to be adapted in order to account for the whole patterns:

$$\text{Pose-MOTP} := \frac{1}{T \cdot \sum_{t=1}^{T} N_t \cdot K} \sum_{t=1}^{T} \sum_{n=1}^{N_t} \sum_{k=1}^{K} \left\| \text{Rot}(r_n^{(t)}) m_k^{(n)} + p_n^{(t)} - \text{Rot}(\hat{r}_n^{(t)}) m_k^{(n)} + \hat{p}_n^{(t)} \right\|_2 \tag{1}$$

Here the function $\text{Rot}(r)$ maps the rotation representation r to a rotation matrix. Note that the MOTP metric disregards missed tracks, false positive tracks and ID-switches. These details are skipped in the formula above for simplicity reasons.

I constrain the problem to the specificities of the bird data by fixing the dimensionality $d = 3$ and the number of markers per pattern $K = 4$. Having $d = 3$ also seems to be the most applicable case in general.

4 My Approaches

I followed two major approaches to solve the specific MOT problem described above, a KF approach and a modern deep learning approach. Both approaches are embedded in the same MOT framework, a tracking-by-detection framework which uses multiple SOTs combined with some simple heuristics to mange multiple tracks. The framework is depicted in figure 2. In the following I describe the components of this framework in more detail.

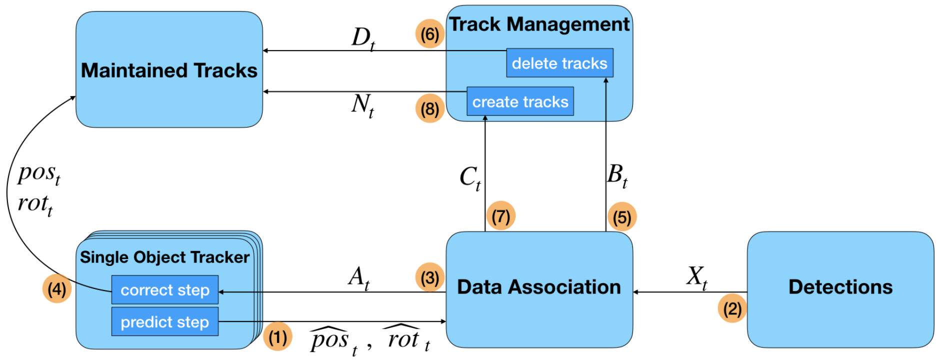


Figure 2: **MOT Framework** - (1) Each SOT predicts location and rotation for frame t. (2) & (3) The detections of frame t are associated with the predicted positions using the HA with the possibility for non-assignments (4) The SOT get the assigned detections as input to correct their internal state and make a more accurate estimation for the current position and rotation. These estimates are stored. (5) & (6) The track management unit gets informed which tracks where unassigned and decides whether to discontinue the track or not. (7) & (8) The track management unit gets all unassigned detections. If it detects the pattern of a untracked bird, a new track is initialized.

Single Object Tracker For each present object there is a SOT, which makes predictions for the position and orientation of the object in the next frame. These predictions are used by the *data association unit* to assign the detections to the objects. The SOTr then gets its assigned detections as input, in order to make a more accurate estimation for the position and rotation for the current frame and to correct its internal state.
This unit is formulated generic, such that multiple methods can be used as SOT. Section 4.1 describes a KF based SOT and section 4.2 describes a neural network based approach.

Data Association or Detection-to-Track-Assignment Unit Due to the fact that the tracking takes place in three-dimensional space, the detection-to-track-assignment is relatively simple, as empirical results suggest (see Section 5). Therefore this work is limited to a simple data association scheme based on the HA. Additionally to the standard HA there is a fixed cost for not assigning a detection to one of the K markers of an object and for leaving one of the K markers of an object unassigned.

Track Management Unit This unit handles the task of creating new tracks and deleting tracks which have been lost. The latter task is done in the simplest way possible. Tracks are deleted, if they were not assigned any detection for more than fixed number of frames.
The creation of new tracks has to be done carefully in order to prevent ID-switches. Therefore, a new track is only initialized when a pattern of one of the objects, which are not currently tracked, exactly matches a subset of the unassigned detections. For larger K it would be sufficient if a subset of the pattern matches a subset of the detections almost perfectly. However since $K = 4$ for the bird data this would lead to falsely creating new tracks frequently.
In this case the algorithm deciding whether new tracks are created can be described as follows: All unassigned detections are clustered using an agglomerative, hierarchical clustering scheme with the euclidean distance as a metric. Clusters are formed by cutting the dendrogram where links have larger distances than the largest distance between points in a pattern. all clusters with 4 or more points are considered as candidates. Then a pattern matching algorithm developed in [Umeyama, 1991] is iteratively applied to the cluster paired with one of the patterns of an untracked object. The pattern matching algorithm finds the similarity transformation (rotation, translation and scaling), that minimizes the mean squared error between the transformed detections and the pattern. For this method to work, the correspondences between detected points and points of the pattern have to be known. This assignment is estimated with a similar method as described in section 4.1.3, just without any prior knowledge of the position and orientation. After ordering the detections correctly the pattern matching algorithm can be applied. If the returned mean squared error is below a certain threshold, the pattern is considered a match and a new track is initialized with the calculated position and orientation. The threshold is chosen very low, such that new tracks are only initialized with very high confidence.

4.1 Kalman Filter Approach

This section describes a SOT, based on a KF approach, which can be used in the MOT framework described above. The KF is a well known method across many different fields and is frequently applied to tracking problems and constitutes an interesting baseline to the MOT problem specified above. The KFr is an algorithm that aims at uncovering the underlying state of a system, given only a sequence of noisy measurements originating from the system. A description of the system and measurements in a probabilistic settings gives rise to a graphical model depicted in figure 3. The KF follows a predict-correct mechanism. It predicts the state for the next time step as a probability distribution. Once a new measurement is available the probability distribution estimating the state is corrected with the new information. If the reader is not familiar with the KF and its variants, I refer the reader to [Prince, 2012, Chapter 19].

If the underlying system is a Linear Dynamic System (LDS), only only Gaussian noise is present and the dynamics and noise parameters are known the KF can be proven to be the optimal estimator for the system state with respect to the mean squared error.
The system at hand is not linear, due to the inherent non-linearity of rotations. For non-linear dynamic systems there are extensions of the KF, such as the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Particle Filters. For tracking orientation, which is commonly called Attitude Estimation in the literature, there is a vast variety of approaches which extend the basic KF to handle the non-linearities and intricacies of the orientation representation correctly. A summary of such approaches is given in [Crassidis et al., 2007]. My approach uses a simple EKF and normalizes the quaternions in the measurement function. The details of my KF approach are given in the remainder of this section. Choosing an EKF over the UKF is justified by the work of [LaViola, 2003], which found that the performance of the EKF and UKF are very similar for estimating quaternions in a noisy setting, whereas the UKF is computationally more expensive.

4.1.1 The State Space and State Evolution

The state space holds all relevant information describing the current state of the system. The state $x^{(t)}$ holds information about the current

$$\text{position } p^{(t)} = \left(p_x^{(t)}, p_y^{(t)}, p_z^{(t)}\right) \in \mathbb{R}^3 \text{ and orientation } q^{(t)} = \left(q_w^{(t)}, q_x^{(t)}, q_y^{(t)}, q_z^{(t)}\right) \in \mathbb{R}^4,$$

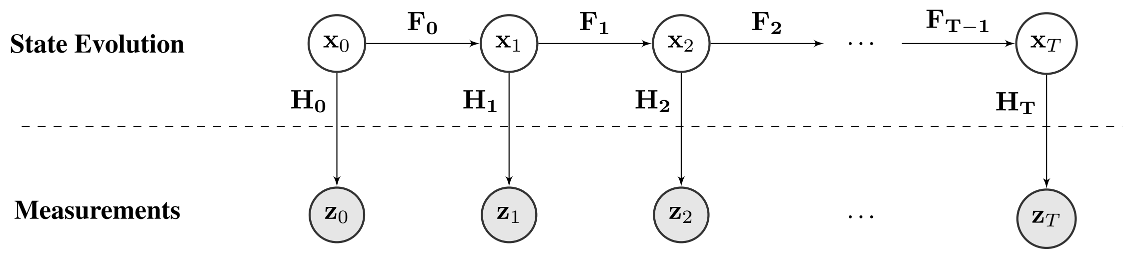


Figure 3: **The KF as a Graphical Model** - In the vanilla KF $x^{(t)} \in \mathbb{R}^n$denotes the detection at time t, $z^{(t)} \in \mathbb{R}^m$ the measurement at t, $F^{(t)} \in \mathbb{R}^{n \times n}$ models the dynamics in the state space and $H^{(t)} \in \mathbb{R}^{m \times n}$ is the measurement function. The random variables are assumed to be gaussian distributed with $x^{(t+1)} \sim \mathcal{N}(Fx^{(t)}, Q^{(t)})$ and $z^{(t)} \sim \mathcal{N}(Hx^{(t)}, R^{(t)})$.

represented as a unit quaternion. Furthermore, the state holds information about the rate of change of $p^{(t)}$ and $q^{(t)}$. The exact form depends on the used motion models.
Assuming a constant velocity model for the position of the birds the state additionally estimates the current

$$\text{velocity } v^{(t)} = \left(v_x^{(t)}, v_y^{(t)}, v_z^{(t)}\right) \in \mathbb{R}^3.$$

A brownian motion model for the quaternions is assumed, meaning that no additional information about the change of the quaternions is present in the state. This is justified by the complication that the function mapping from the state space to the measurement space is not known beforehand, the correction step of the KF might introduce large changes in the estimate for the quaternion. When using a for example a constant velocity model, these disruptions become even worse and impact the future frames more heavily than a brownian motion model.

With these assumptions the equations governing the state evolution are as follows:

$$p^{(t+1)} = p^{(t)} + v^{(t)} + \eta_p^{(t+1)} \qquad (\eta_p \sim \mathcal{N}(0, \sigma_p \cdot I_3)) \tag{2}$$

$$v^{(t+1)} = v^{(t)} + \eta_v^{(t+1)} \qquad (\eta_v \sim \mathcal{N}(0, \sigma_v \cdot I_3)) \tag{3}$$

$$q^{(t+1)} = q^{(t)} + \eta_q^{(t+1)} \qquad (\eta_q \sim \mathcal{N}(0, \sigma_q \cdot I_4)) \tag{4}$$

, where σ_p, σ_v and σ_q are parameters determining the level of noise in the state evolution for the position, velocity and quaternion respectively. Putting this in a vectorized form gives the linear state transition function F as a matrix for the KF:

$$x^{(t+1)} = Fx^{(t)} = \begin{pmatrix} 1&0&0&1&0&0&0&0&0&0\\ 0&1&0&0&1&0&0&0&0&0\\ 0&0&1&0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \begin{pmatrix} p_x^{(t)}\\ p_y^{(t)}\\ p_z^{(t)}\\ v_x^{(t)}\\ v_y^{(t)}\\ v_z^{(t)}\\ q_w^{(t)}\\ q_x^{(t)}\\ q_y^{(t)}\\ q_z^{(t)} \end{pmatrix} + \eta \tag{5}$$

, where $\eta \sim \mathcal{N}(0, diag(\sigma_p, \sigma_p, \sigma_p, \sigma_v, \sigma_v, \sigma_v, \sigma_q, \sigma_q, \sigma_q, \sigma_q))$.

4.1.2 The Measurement Space and Measurement Function

The measurement function mapping from the state space to the measurement space is discussed next. In an ideal world, where no detections can be lost and the order of the detections is always the same, the relation of the measurement to the state is described by the following equation:

$$\tilde{H}(x^{(t)}) + \zeta = \begin{pmatrix} \text{Rot}(q^{(t)})m_1 + p^{(t)} \\ \text{Rot}(q^{(t)})m_2 + p^{(t)} \\ \text{Rot}(q^{(t)})m_3 + p^{(t)} \\ \text{Rot}(q^{(t)})m_4 + p^{(t)} \end{pmatrix} + \zeta = \tilde{z}^{(t)} \in \mathbb{R}^{12} \qquad \text{, where } \zeta \sim \mathcal{N}(0, \varrho \cdot I_{12}) \tag{6}$$

Since this measurement function is not linear and the quaternion has to be of unit norm the standard KF cannot be applied anymore. There are several variants of the EKF and UKF which deal with these issues. [Crassidis et al., 2007] gives an overview the most common techniques. Interesting new approaches like [Hertzberg et al., 2011] impose the manifold structure of the state space on the KF algorithm. I did not pursue any of the more elaborate approaches mentioned, yet, since the relation between the state and measurement space are more complicated, as explained next. Therefore the performance limiting factor is not the correct handling of the quaternions, but rather the detection-to-marker assignment. Instead I chose to include the normalization of the quaternions in the function Rot(.), mapping q to a rotation matrix. Note that this leaves the actual estimate for the quaternions unconstrained. Despite this, empirical results show that the the individual components of the quaternion estimates do not diverge further away from zero than 2.5. In future work investigating the performance gain of implementing a more suitable variant of the EKF or UKF would be interesting.

The actual measurement function needs to take the visibility and ordering of the detections into account. An algorithm to perform this detection-to-marker-assignment is discussed in section 4.1.3. As stated in the problem formulation in section 3 detections are ordered in an arbitrary fashion, meaning that $z_{\lfloor i/3 \rfloor}$ doesn't necessarily correspond to the i-th marker. The correct function mapping from the system state to the measurements can then be expressed as follows:

$$H(x^{(t)}) + \zeta = A^{(\pi^t)} \begin{pmatrix} \text{Rot}(q^{(t)})m_1 + p^{(t)} \\ \text{Rot}(q^{(t)})m_2 + p^{(t)} \\ \text{Rot}(q^{(t)})m_3 + p^{(t)} \\ \text{Rot}(q^{(t)})m_4 + p^{(t)} \end{pmatrix} + \zeta = z^{(t)} \tag{7}$$

, where $A^{(\pi^t)}$ is responsible for ordering and removing markers if they are not currently visible. Let $\pi \in \{1, \ldots, K\}^1 \cup \cdots \cup \{1, \ldots, K\}^K$ be a tuple dictating the ordering of the detections, where π_i is the index of i-th marker detected. In this definition I omit the possibility to classify all detections as false positives.
For a tuple π of length l, $A^{(\pi)} \in \mathbb{R}^{l \cdot 3 \times d \cdot K}$ can be defined as follows. The 3×3 block at block-index $i \in \{1, \ldots l\}, j \in \{1, \ldots, K\}$ is defined as

$$A^{(\pi)}_{i,j} := \begin{cases} I_3, & \text{if } \pi_i = j \\ 0_3, & \text{otherwise} \end{cases} \tag{8}$$

.

4.1.3 Detection to Marker Assignment

The detection-to-marker-assignment is a crucial part of the proposed KF, as it estimates $\pi^{(t)}$. Wrong estimates of $\pi^{(t)}$ produce a false measurement function, which breaks down to whole concept of the KF. The position estimate is not strongly effected by such errors, since the rotation uses the position estimate as origin. The orientation estimate however suffers strongly from errors in estimating $\pi^{(t)}$. Therefore the detection-to-marker assignment is designed to give the best possible estimate without regards of the runtime.

The estimation is done by energy minimization of a handcrafted energy E, which combines rotation and translation invariant information about the pattern of the object, as well as the prior knowledge of the rotation and translation as predicted by the KF.
Let there be D point detections written as matrix of detections $X = (X_1, \ldots, X_D)^T$. Let $m = (m_1, m_2, m_3, m_4)^T$ be the pattern of the object of interest. Then any non-empty assignment π of markers to detections can then be described by:

$$\pi \in \{1, \ldots, 4\}^1 \cup \cdots \cup \{1, \ldots, 4\}^D =: \Pi \tag{9}$$

, where $\pi_i = j$ means that the i-th detection, X_i, is assigned the j-th marker, m_j, of the pattern. Let M be the length of π. Allowing for $M < D$ means that detections can remain unassigned, which corresponds to classifying them as false positive detections, which frequently appear in practice.

The rotation and translation invariant part of the energy, $E_1(\pi)$ is defined as the mean squared error between the pairwise distances of the pattern m and the detections X brought into order by π:

$$E_1(\pi) = \frac{1}{\binom{M}{2}} \sum_{i=1}^{M} \sum_{j=i+1}^{M} \left(\|m_i - m_j\|_2 - \left\|X_{\pi_i} - X_{\pi_j}\right\|_2 \right)^2 \tag{10}$$

The second part of the energy, $E_2(\pi)$, makes use of the KF predictions for position, $\hat{p}$, and orientation, $\hat{q}$. Assuming that these predictions are good, which is not always the case in practice due to the simplified motion model, the rotated and translated pattern should be close to the detections ordered by π:

$$E_2(\pi) = \frac{1}{M} \sum_{i=1}^{M} \|\mathrm{Rot}(\hat{q}) m_i + \hat{p} - X_{\pi_i}\|_2 \tag{11}$$

Finally since $E_1(\pi)$ and $E_2(\pi)$ become small for small M, not assigning a detection to marker is punished with a fixed cost $c_{\text{nonAssignment}}$. The energy $E(\pi)$ is then composed of a weight sum of the three parts specified above:

$$E(\pi) = \lambda_1 \cdot E_1(\pi) + \lambda_2 \cdot E_2(\pi) + (D - M) \cdot c_{\text{nonAssignment}} \tag{12}$$

$E_1(\pi)$ is crucial in scenarios where the predictions are bad, $E_2(\pi)$ enforces temporal consistency and punishing non-assignments rewards longer π, which carry more information than short π.
Balancing the influences by choosing the parameters $\lambda_1, \lambda_2, c_{\text{nonAssignment}}$ was done in an empirical fashion with a very coarse grid search.

The estimate $\hat{\pi} = \text{argmin}_{\pi \in \Pi} E(\pi)$ is obtained by a simple exhaustive search over Π.

This exhaustive search heavily exploits the fact that $K = 4$ and would not be practically possible for large K. In such cases an Iterative Closest Point(ICP) [Zhang, 1994] algorithm could be used instead. Then it would be possible to only keep the matches with the highest certainty and discard the rest, since not all points are needed to estimated the rotation and translation and wrong matches could disrupt the KF. ICP is solving a problem very similar to what required in the detection-to-marker-assignment. Therefore it would be interesting to see how ICP compares to method described above, although I could not find information whether ICP works well with a small number of points, as local minima could potentially become a problem. [Pomerleau et al., 2014] would be a good point to start searching for the right ICP variant for this specific problem.

4.1.4 Some Heuristics

There are two notable heuristics I added to the KF approach.
Both are validated with experiments in section 5.2.1.

False Positive Filtering The detection-to-track-assignment is handled with the HA and a fixed cost for non-assignments. Having a fixed cost however is harmful, as the cost has to be high enough in order to allow for the erroneous prediction of the simplified motion model. Therefore if some or all actual detections are missing, false positive detections could be assigned to a track and if none ore only one true positive detection is assigned, it becomes hard to identify false positives, which leads to jumps in the predicted position and orientation and can cause to lose a track in the worst case.
To filter such false positives the non-assignment cost is scaled with velocity an object is currently moving, meaning that faster moving objects can still endure prediction errors of the motion model, while stationary objects get assigned less false positives.

Adaptive Noise Similarly having a constant measurement noise scale ϱ might be fitting for the regular KF setting, but as the measurement function is only estimated in this case, as described in the previous section, adapting the measurement noise to the certainty of the estimation of π seems reasonable. Therefore the covariance matrix of the measurement noise is scaled by $E(\pi)$. This means that frames with a certain detection-to-marker-assignment will have a stronger influence on the state estimation.

4.2 Deep Learning Approach

The recent success of deep learning in many different areas also motivates to consider neural network for this task. The models described in the remainder of this section are SOT models and can be used in the MOT framework described in section 4.

Potential benefits I believe that the main benefits of a Deep Learning approach compared to the KF approach are twofold:
First, a more complex motion model allows more accurate position predictions. While the relatively simple motion model of the KF might work well for predictions for the next time step, it especially breaks down when detections are missing for a few frames. A more complex motion model can accurately anticipate the next few time steps and therefore fill holes where detections are missing completely for multiple consecutive frames. This can potentially reduce the number of tracks being lost, which is important, since initializing new tracks can only be done in frames where the pattern can be determined with very high certainty, which occurs rarely.
Since there is some degree of non-determinism in the behavior of most objects, like in birds or pedestrians, predicting future positions is complicated. In order to tackle this problem, commonly multi-modal position predictions are used. Combining multi-modal predictions and long-term predictions is however far from trivial. [Zyner et al., 2018] present a possible solution for driver trajectory predictions in intersections. Whether such an approach is feasible for birds remains questionable, as the topology of the intersection strongly constrains possible trajectories.

Secondly, missed and false positive detections are a problem for the KF approach, as the simplified motion model does not carry over detailed information about the orientation to the detection-to-marker-assignment. A neural network can develop more complicated methods, which can draw more information from previous frames. The temporal information together with the position of the non-missing markers should be enough to reconstruct the positions of the missing markers, in order to estimate the rotation accurately.

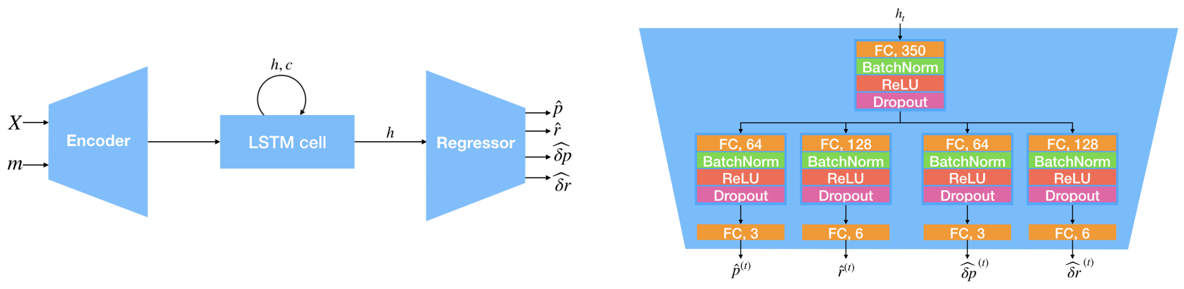
Deep Learning specific Challanges While Deep Learning looks like a promising method, designing an effective architecture is not trivial. In the following I list some of the problems that render the Deep Learning approach challenging:

- Working on point clouds is problematic, as their input order should not be taken into account. Designing architectures which are invariant to the input order of detected points is tricky and described in section 4.2.2.
- Furthermore the network has to be capable of accepting a varying number inputs, as the amount of detections varies from frame to frame. This is also addressed in section 4.2.2
- Additionally the network has to perform well with every possible pattern, which is potentially of interest. In this case the neural network should work on all patterns used when recording the birds and on all patterns to be used in the future, as frequent retraining is tedious.
- Analytically estimating rotation and translation between two point clouds can be done with ICP or [Umeyama, 1991], if the correspondences are known. However, this task alone is already non-trivial for neural networks, as described in section section 4.2.3.
  Alternatively one could avoid estimating rotations directly, by just regressing the marker positions directly and calculating the rotations and translation in a post-processing step with [Umeyama, 1991]. I haven't followed this approach yet.
- Where to insert the temporal information as well as how to incorporate the pattern of the tracked object are both not obvious. The overall proposed architecture will be discussed next.

4.2.1 Proposed Architecture

The general architecture consist of an encoder neural network which maps the detected point cloud and the pattern of the tracked object to a more suitable, high dimensional representation, which is fed into the LSTM network. Note that since the pattern is not changing while tracking, it does not need to be fed into the network every step. For example It would be possible to initialize the hidden state from the pattern. The final hidden state of the LSTM is then mapped to position and orientation estimates, as well as, predicted position and orientation change for the next frame. Figure 4(a) shows the architecture on a high level. The exact form of the encoder is described in section 4.2.2 and is a less obvious choice, while using a LSTM to transfer temporal information and an MLP network as regressor is quite natural.
Yet another approach would be to split up the task into two subtasks. The first would be classifying the detections into which marker they correspond to or whether they are false positive detections. This would correspond to a detection-to-marker-assignment. Then one could bring the detections into order and remove the false positives. A second network would then operate on the ordered detections, which makes the task of the second network significantly

easier. I believe however, that imposing this structure onto the neural network is inferior opposed to leaving the neural network all freedom. Therefore I did not pursue this method.



(a) General Architecture (b) Regressor Architecture

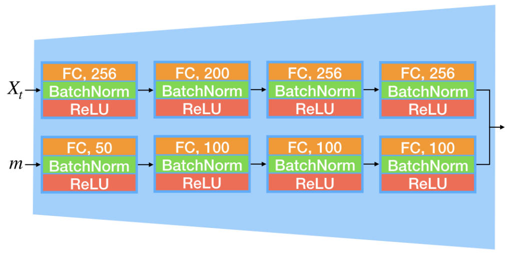
Figure 4: The neural network gets the detections at each frame, X_t and pattern$m$ as input and predicts the current position $\hat{p}_t$ and rotation $\hat{r_t}$. Additionally the change in position and orientation to the next frame, $\widehat{\delta p_t}$ and $\widehat{\delta r_t}$, are predicted. These can be used for predicting the state in the next frame and for the detection-to-track-assignment. (a) The LSTM cell has 600 hidden units in all experiments. (b) The input is the hidden state of the LSTM at time step t. The numbers indicate the used number of hidden units. The dropout rate was set to 0.25. Using an additional fully connected layer in the regressor did not make a real difference in performance.

4.2.2 Encoder Architecture

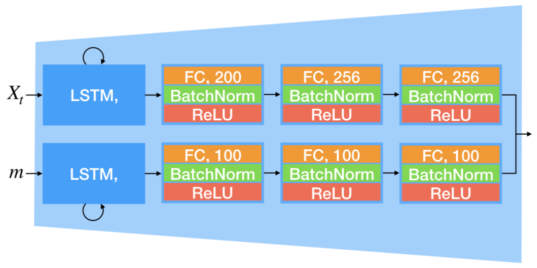
As the detections come in an arbitrary ordering the encoder network has to be invariant to permutations of the inputs, i.e. it has operate on point clouds. This exact challenge is discussed in [Qi et al., 2016]. The authors designed an effective and efficient deep learning architecture, which works on point clouds and therefore is permutation invariant. Their approach is on par with other approaches, like Voluminetric CNNs [Zhirong Wu et al., 2015] and Multi-view CNNs [Su et al., 2015], while being computationally much more efficient. Voxelization and rendering images from multiple views does not seem to make sense for 4 points anyways.
Another alternative discussed in the paper is treating the detections as a sequential signal and training a Recurrent Neural Network (RNN) on randomly permuted sequences, hoping that the network will become permutation invariant. They found, that this approach is worse than their proposed architecture. The paper "Order Matters" [Vinyals et al., 2015] gives various examples where input (and output) order in set2seq (seq2set) problems made a significant difference on performance, implicating that RNNs in general cannot be trained to become order invariant. However they suggest that for short sequences (in their experiment they used a of length 5) the RNN seemed to less affected by the ordering.
Both papers did not mention MLPs, as they cannot deal with variable sequence length. In this problem however one can pad the assigned detections with zeros and introduce an indicator variable masking the padded points. Then there are always 4 assigned detections as input and an MLP can be applied.
The architectures are depicted in figure 5. In section 5.2.2 I will compare the performance of a PointNet-sytle encoder, a LSTM encoder and a MLP encoder.

4.2.3 Representing Rotations

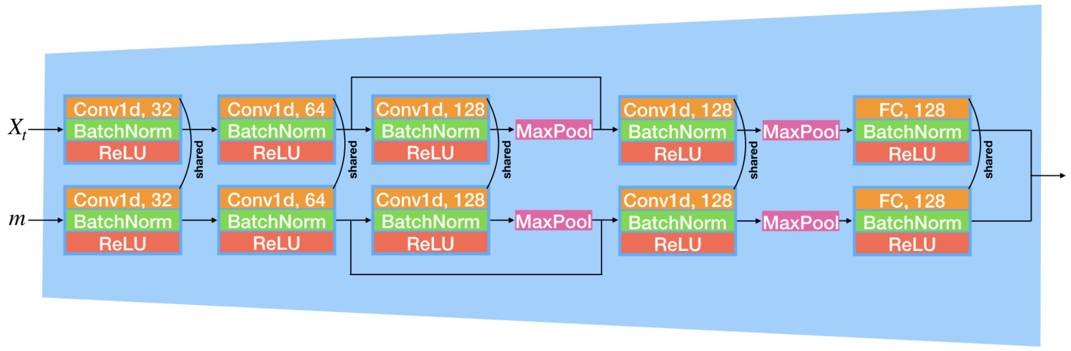
Choosing a suited rotation representation method for neural networks is essential. Most common representations as Euler angles and quaternions suffer from certain discontinuities, as described in [Zhou et al., 2018]. The paper proposes an alternative continuous 6-dimensional representation for rotations in 3-dimensional space, which alleviates the problems of discontinuous representations. My empirical results support these findings, see section 5.2.2. The authors argue that instead of directly regressing the rotation matrix, which would not suffer from discontinuities, and then applying a Gram-Schmidt process to orthogonalize it, it is possible to regress 6 parameters, which can be mapped



(a) MLP Encoder



(b) LSTM Encoder



(c) PointNet-style Enocder

Figure 5: For all architectures the detections are padded with zeros and a variable indicating the padded entries, such that $X_t \in \mathbb{R}^{6\times4}$. For (a) and (b) the detections are shuffled such that the network learns to become permutation invariant. For (a) moving the passed zeros to the back was greatly beneficial. (c) is a similar architecture to the PointNet [Qi et al., 2016] architecture, except that no matrix transform is applied as translation rotation is of interest. Therefore the detection order has no effect on the network. The features after the first pooling operation hold information about the whole point cloud. These are concatenated with the local features after the second 1d convolution in order to combine both local and global information. Whether sharing weights makes sense has to be tested, as the only one part has to deal with noise. Furthermore the pattern-network-part always gets the same pattern meaning that a simple MLP network would be sufficient.

to a rotation matrix with the following simple mapping:

$$\begin{aligned}
\text{Rot}\begin{pmatrix} | & | \\ a_1 & a_2 \\ | & | \end{pmatrix} &:= \begin{pmatrix} | & | & | \\ b_1 & b_2 & b_3 \\ | & | & | \end{pmatrix} \qquad \text{, where} \\
b_1 &:= \frac{a_1}{\|a_1\|_2} \\
b_2 &:= \frac{a_2 - \langle b_1, a_2\rangle b_1}{\|a_2 - \langle b_1, a_2\rangle b_1\|_2} \\
b_3 &:= b_1 \times b_2
\end{aligned}$$

5 Results

5.1 Generating Data

As there is no labeled data of the birds available, I had to generate data myself, in order to evaluate the performance of the models quantitatively. However generating realistic bird behavior from scratch is difficult, I took two different

approaches. First I generated random 3d trajectories and orientation trajectories by composing functions from multiple simple functions. The generate behavior is far from actual bird behavior, however I tried to come up with a wide variety of data in order to test the generality of the models. I denote this data set by "artificial" data. Additionally I generated data based on the KF predictions in the real detections. I smoothened the predictions and only used parts without obvious errors, like discontinuities or extreme motion or rotation. This data is denoted by "birdlike".
Equally important as the generate underlying behavior is the simulation of the detections and introducing realistic measurement errors. I came up with two random processes to simulate the presence of false positive detections as well as missed detections. These random processes are solely inspired by observing the real detections and the following two observations: First missing detections do not occur uniform at random. It seems that rather some markers go missing for multiple frames at a time, before producing reliable detections again. This could be explained by bird covering a marker with a feather for some time. Second false positive detections also do not appear uniformly at random. There seem to be some points in the room which sometimes reflect the infrared light for certain periods.
Using these stochastic processes allows to specify the amount of noise present.

5.2 Experimental Results

5.2.1 Evaluating the Kalman Filter Approach

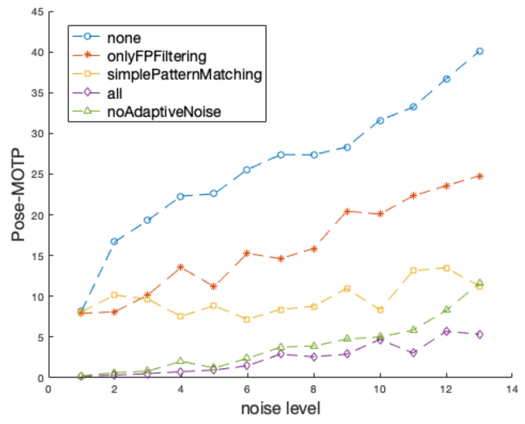
Figure 6 shows the MOTA and Pose-MOTP metrics on a birdlike dataset with 10 birds present. Therefore, this experiment is a bit biased, as the result of the KF were used to obtain the underlying trajectories. However the noise levels were simulated with the two stochastic processes, as described in the previous section, meaning that this experiment is still fit to investigate the noise resistance of the different KF versions. The videos found in GitHub also show the MOT results on the real detections. The results seem to be of high quality, indicating that even the most simple motion model combined with the HA is sufficient to master the detection-to-track-assignment. In the following I explain the effects of the three investigated additions to the most simple KF version.

Proposed detection-to-marker-assignment The only version using this detection-to-marker-assignment, as described in section 4.1.3, are *noAdaptiveNoise* and *all*. It is clear to see that the Pose-MOTP error is drastically lower when using this method instead of a simple HA on distances of the detections to predicted marker locations. When comparing the Pose-MOTP of *simplePatternMatching* and *all*, which only differ in the detection-to-marker-assignment, one observes that the simple detection-to-track-assignment never really manages to predict the correct pose, even in low noise levels. The effect on the MOTA is less obvious.
False positive filtering This heuristic, as explained in section 4.1.4, is used in all versions except in *all*, and is the only difference between *all* and *onlyFPFiltering*. Clearly both the Pose-MOTP and especially the MOTA are strongly positively influenced by this heuristic.
Adaptive Noise The effect of this heuristic, as described in section 4.1.4, can be seen when comparing the version *noAdaptiveNoise* and *all*, which only differ in the use of adaptive noise. Both the Pose-MOTP but especially the MOTA are positively influenced by this method.

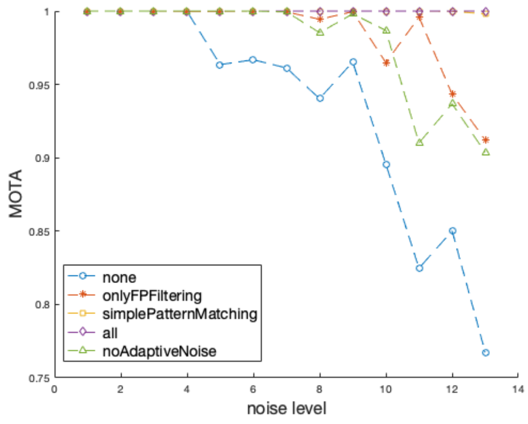
5.2.2 Evaluating the Deep Learning Approach

Training Procedure In order to ease the training of the neural network, a simplified task is considered. The training data consists of 50.000 sequences of 100 frames each. The test dataset hold 5.000 sequences of length 100. However each sequence is scaled an shifted such that the mean of each sequence is 0 and the standard deviation over the norms of each position in each sequence is 1. This way the training data densely covers the vicinity of the origin and makes training a lot easier. This simplification has the drawback that the neural network cannot be easily applied to a long sequence anchored at an arbitrary position in the room.
Furthermore the neural network is only evaluated as a SOT. Therefore predicting $\widehat{\delta p}$ and $\widehat{\delta r}$ is skipped.
The networks are trained for 100 epochs with a batch size of 64. The Adam optimizer with an initial learning rate of 0.001 and a momentum of 0.9 is used. The learning rate is reduced with a factor of 0.5 upon hitting a plateau. Here a patience of 3 and cooldown of 10 epochs is used. Furthermore weight decay with a weight decay of 0.0002 is applied. The resulting performance can be found in table 1.

Comparing Encoder Architectures The results in table 1 clearly show that the MLP encoder works best in all settings. It is interesting to note that the LSTM encoder works quite well in a noise free setting, but is heavily affected by increasing noise levels. Despite the promising results of [Qi et al., 2016] the PointNet encoder did not work well. It has to be noted that the training pose error was comparable to the other models. However I was not able to overcome this severe overfitting, despite introducing more dropout, a higher weight decay and fewer parameters. I cannot explain this behavior and further investigation of this issue is necessary.



(a) Pose-MOTP



(b) MOTA

Figure 6: **KF performance on different noise levels** - These results compare the MOTA and Pose-MOTP for different noise levels and different versions of the KF. *none* refers to the most simple Kalman Filter with no additions and the HA for the detection-to-marker-assignment. *onlyFPFiltering* is the same as *none* with the addition of false positive filtering as described in section 4.1.4. *simplePatternMatching* has both false positive filtering and adaptive noise, as describe in section 4.1.4, but also the same simple detection-to-marker-assignment as *none*. *noAdaptiveNoise* uses the detection-to-marker-assignment described in section 4.1.3, false positive filtering but no adaptive noise. *all* uses all the additions. Note the y-axis range for figure (b).

Comparing Rotation Representation The results in table 1 suggest that representing rotations as unit quaternions introduces issues in neural networks and matches with the findings of [Zhou et al., 2018]. It is interesting to see that the pose error does not drastically increase with increasing noise. This suggests that not the noise is the main problem but rather the rotation representation itself.

model type	model + rotation representation	no noise	medium noise	high noise
NN	MLPencoder + 6d	**0.032**	**0.042**	**0.077**
NN	MLPencoder + quat	0.112	0.108	0.150
NN	LSTMencoder + 6d	0.057	0.164	0.243
NN	PointNetEncoder + 6d	0.468	0.710	0.696
KF	proposedKF + quat	**0.008**	**0.011**	**0.031**

Table 1: **Comparing pose error of different models on artificial data** - The experiments presented here were conducted on artificial data of a single object. The used metric is the pose error, similar two the Pose-MOTP defined in section 3, but in a SOT setting. All models were evaluated on different noise levels. The evaluated neural network models only differ in the encoder architecture, as discussed in 5 and the best neural network performing model was also evaluated with internally representing rotations as quaternions, in order to show the dominance of the 6d rotation representation explained in 4.2.3. The proposed KF was evaluated on the same data.

5.3 Comparing both approaches

The results of the MLP encoder network and the KF approach in both tables 1 and 2 suggest that the neural network is inferior to he KF approach, despite the fact that the data was generate in a form, which rendered training the neural network easier, as describe in section 5.2.2. However, the neural network seems to be less influenced by increasing noise levels. The neural network is already inaccurate in a noise free setting. Thus there might be some flaws in the architecture, as it should be possible to achiever near perfect results in a noise free setting. Once these issues have been resolved, the higher resistance to noise could make the neural network perform better than the KF.
The KF also performed better on the birdlike data, as the trajectories are simpler than in the artificial data. This is due to the simplified motion model of the KF.

Another advantage of the KF is its generality. It makes no assumption about the behavior or the patterns of the objects. Therefore only a few parameters might have to be changed in order to apply the model to a completely new setting. Although I have not investigated the generality of the neural network by evaluating it on different kinds of data than training it on, for certain changes to the setup a complete retraining might be necessary.
One drawback of the KF is the assumption that K is small, as the detection-to-track-assignment consumes exponential time in K.

model	medium noise	high noise
proposedKF	**0.0025**	**0.0055**
MLPencoder + 6d	0.0348	0.045

Table 2: **Comparing pose error on birdlike data** - The experiments followed the same setup as in table 1. However the numbers cannot be directly compared, as the data had different scale due to the normalization procedure.

6 Future Work

The KF already produced decent results, despite not treating the quaternions correctly, only being an EKF and using a very simple motion model. In future I will investigate slightly more sophisticated motion models and use a version of the EKF which treats the quaternions appropriately. Another interesting line of research is estimating the parameters of the KF, as they have to hand tuned as of now. Whether one of the common approaches, like Expectation Maximization for parameter estimation, is applicable to this setting remains to be seen. Not knowing and only estimating the measurement function might introduce problems.
The deep learning approach is showing promising results in terms of noise resistance, but still makes mistakes in a noise free setting. This suggests that the architecture is not optimal yet. In the following I list some ideas how to improve the neural network. The experiments showed that the rotation representation can be the limiting factor. Therefore directly regressing the marker positions is an alternative approach which bypasses representing rotations. As regressing points is successfully done in practice, e.g. human joints or bounding boxes, this approach seems promising.
Furthermore, the neural networks are not capable of tracking for long, as they are only trained on short sequences implying that the object cannot leave far from the origin. Resolving this issue will be of future work.
Another issue is making the encoder permutation invariant. Training a MLP to become permutation invariant seems wasteful. However the PointNet architecture overfit severely. In future I will investigate this issue. However only working with 4 points could explain the bad performance, as PointNet normally is applied to dense meshes.
The fact that all detections of a single bird sometimes go missing for multiple consecutive frames, encourages further investigation of the neural network approach, since a more sophisticated motion model can potentially fill these holes and prevent the track from getting lost.
Yet another possibility is to combine both approaches, by training a neural network to perform the detection-to-track-assignment in the KF approach.

7 Conclusion

In this work, a new kind of MOT was formulated. Experiments showed that the detection-to-track-assignments, which is commonly the most challenging part of MOT, is simple. Therefore, the difficulty lies in the precise tracking of position and orientation for single objects, which are then combined to MOT with a distance based HA for the detection-to-track-assignment and a few simple heuristics. Both introduced model are SOT for this framework. A simple EKF with a few additions to handle specific difficulties quantitatively achieves good results on generate data. A qualitative observation of the MOT results on the real data confirms the good results. The neural network based SOT introduced here still needs improvement, but already shows promising noise resistance. Several improvement ideas for both models are given.

References

[Bernardin and Stiefelhagen, 2008] Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309.

[Crassidis et al., 2007] Crassidis, J. L., Markley, F. L., and Cheng, Y. (2007). Survey of nonlinear attitude estimation methods. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28.

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Hertzberg et al., 2011] Hertzberg, C., Wagner, R., Frese, U., and Schröder, L. (2011). Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *CoRR*, abs/1107.1119.

[LaViola, 2003] LaViola, J. J. (2003). A comparison of unscented and extended kalman filtering for estimating quaternion motion. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 3, pages 2435–2440 vol.3.

[Leal-Taixé et al., 2015] Leal-Taixé, L., Milan, A., Reid, I., Roth, S., and Schindler, K. (2015). MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942 [cs]*. arXiv: 1504.01942.

[Leal-Taixé et al., 2017] Leal-Taixé, L., Milan, A., Schindler, K., Cremers, D., Reid, I. D., and Roth, S. (2017). Tracking the trackers: An analysis of the state of the art in multiple object tracking. *CoRR*, abs/1704.02781.

[Milan et al., 2016a] Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016a). MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*. arXiv: 1603.00831.

[Milan et al., 2016b] Milan, A., Rezatofighi, S. H., Dick, A. R., Schindler, K., and Reid, I. D. (2016b). Online multi-target tracking using recurrent neural networks. *CoRR*, abs/1604.03635.

[Pomerleau et al., 2014] Pomerleau, F., Colas, F., and Siegwart, R. (2014). A review of point cloud registration algorithms for mobile robotics.

[Prince, 2012] Prince, S. (2012). *Computer Vision: Models Learning and Inference*. Cambridge University Press.

[Qi et al., 2016] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593.

[Shi et al., 2018] Shi, S., Wang, X., and Li, H. (2018). Pointrcnn: 3d object proposal generation and detection from point cloud. *CoRR*, abs/1812.04244.

[Su et al., 2015] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. G. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*.

[Umeyama, 1991] Umeyama, S. (1991). *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:376–380.

[Vinyals et al., 2015] Vinyals, O., Bengio, S., and Kudlur, M. (2015). Order matters: Sequence to sequence for sets.

[Weng and Kitani, 2019] Weng, X. and Kitani, K. (2019). A baseline for 3d multi-object tracking. *CoRR*, abs/1907.03961.

[Zhang, 1994] Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces.

[Zhirong Wu et al., 2015] Zhirong Wu, Song, S., Khosla, A., Fisher Yu, Linguang Zhang, Xiaoou Tang, and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920.

[Zhou et al., 2018] Zhou, Y., Barnes, C., Lu, J., Yang, J., and Li, H. (2018). On the continuity of rotation representations in neural networks. *CoRR*, abs/1812.07035.

[Zyner et al., 2018] Zyner, A., Worrall, S., and Nebot, E. M. (2018). Naturalistic driver intention and path prediction using recurrent neural networks. *CoRR*, abs/1807.09995.