3D Trajectory Reconstruction for Animal Data

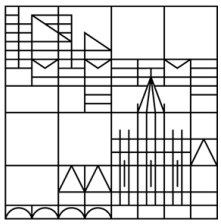
Master Thesis

written by

Giray Tandoğan

at the

Universität Konstanz



Faculty of Sciences

Department of Computer and Information Science

1. Reviewer: Prof. Dr. Oliver Deussen

2. Reviewer: Prof. Dr. Bastian Goldlücke

Konstanz, 2022

Abstract

There are many use cases where trajectory matching can be useful. One example of this can be studying 3D movements of animals for collective behavior studies by using 2 or more cameras which can provide 2D trajectory information of freely moving animals. These 2D trajectories from each view can be used for many purposes such as reconstructing and visualizing the collected data in 3D. However, the collected 2D trajectory in each view should be matched with their corresponding 2D trajectories on other cameras to be able to reconstruct recorded 3D motion of animals. In this thesis, a 2D trajectory matching method and a 3D trajectory reconstruction method are implemented by using epipolar geometry.

Keywords: epipolar geometry, trajectory matching

Acknowledgement

Firstly, I would like thank my examiner and reviewer Prof. Dr. Oliver Deussen for his understanding and guidance. Also, I would like to express my sincere gratitude to my supervisor Hemal Naik and my co-supervisor Thejasvi Beleyur. Their understanding and support were incredibly valuable for me.

Secondly, I would like to thank my parents, my brother and my sister. Their support made it possible for me to come this far. They helped and assisted me in challenging times with every power that they have.

Lastly, during both challenging and good times in Konstanz, I had my friends with me. I would like to thank specifically Kerem Dönmez, Gülsüm Ünal, Deniz Dölek, Carla & Anna Schuy, Gökhan İder, and Selçuk Çabuk for their friendship, support, and perspectives.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In a real-world scenario, animal behavior researchers record animals by using multiple cameras. However, the recorded data is in 2D and the corresponding points may not be know. In order to be able to reconstruct 3D data, the corresponding points on cameras should be know. This paper covers the case where there are 2 cameras recording animals, and provides 2D trajectories on each camera. The implemented methods in this paper solve the issues of finding corresponding 2D trajectories and reconstructing them in 3D by using Epipolar Geometry [6]. A real-world, with 2 camera views, correspondence matching scenario example can be seen in Figure 2.

1.1 Overview

Assuming the 2D trajectory information of objects is provided and matched, it is possible to reconstruct their corresponding 3D positions. If the reason for reconstruction is to visualize multiple objects in 3D, depth information is essential to be able to represent the original 3D recorded data. If two or more cameras are used, depending on their relative positions and orientations, 2D points can be reconstructed back into 3D space. A common technique is to reconstruct 2D points into 3D space is to use epipolar geometry. An example of epipolar geometry can be seen in Figure 1.

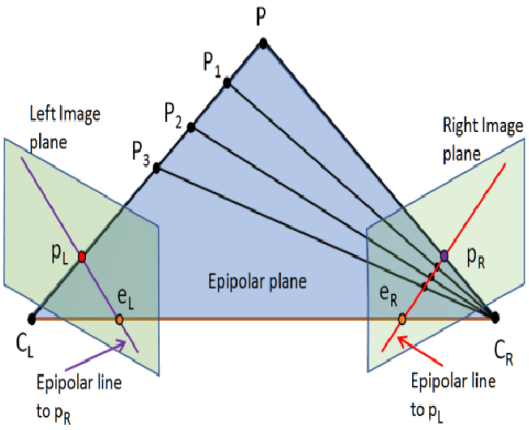


Figure 1: Epipolar geometry of stereo vision. Directly taken from the works of [4].

As it can be seen in Figure 1, a point P is in the field of view (FOV) of both cameras. The projection of point P on the left image plane is p_L and on the right image p_R. However, if the point P is moved to P_1, P_2 and P_3, it would be at the same location on the left image plane which is p_L even though it is moved on the ray in 3D space. As it is mentioned before, we would need accurate matching in order to able to reconstruct objects in 3D space.

Thus, as Figure 1 shows, depending on the matched points, the depth estimation changes. Therefore, correct matching is required to be able to accurately reconstruct 3D points. If we do not know the exact location of point P on the right image plane, we can assume that it should be on the *epipolar line*[6]. How to get the location of e_R and other details will be discussed in Section 2. Therefore, if a point's location on both image planes is known, it is possible to estimate the position of the points in 3D. This process can be called "reconstruction" of corresponding 2D points belonging to a single object in 3D space. In order to get the correct pairs on image planes, we can use points' locations and calculate their distance to the *epipolar line*s. If 2D temporal trajectory data are given for both of the views without the information of which 2D trajectory is corresponding to which 2D trajectory on the other plane, they should be "matched" accurate representation in 3D space. It is possible to give a random depth value to a 2D point on the left image plane which should be in the FOV of the other camera, which would put it on a line from C_L to P in 3D

space. We can take the created 3D point and project it to the other image plane and calculate the *epipolar line*. After that, we would have a line called *epipolar line* which can help us to find the right pair [4][7][8]. In order to find the pair, we can measure the perpendicular distance to the *epipolar line* for all points on the right image plane and a 2D point with the shortest perpendicular distance to the *epipolar line* which goes through e_R and p_R would be the corresponding point of P_L on the right image plane. It is important to note that, if there are multiple objects in the 3D space, there can be other points that are very close to an *epipolar line* which may cause wrong pair identification. This means, just measuring distances to *epipolar line*s may not be sufficient to match correct points. A mismatched point would create high 3D estimation errors. Therefore, when trying to match 2D trajectories with their corresponding 2D trajectories on other images plane, we can try to match multiple points of those 2D trajectories at different frames.

In order to achieve better results from the pairing process, Kalman Filter [9] is used. The main idea is to be able to make an estimation for the next state based on the previously matched frames and pairs. In other words, after a certain amount of frames are matched and objects are paired, the estimated positions of the objects can be used for a forecast calculation. If the forecast or estimated position is in the range of a certain distance threshold, the object is paired. The distance threshold is used between the forecast position for the current frame and the current best candidate from the epipolar line process. Different ways of using Kalman Filter are implemented and will be discussed in Section 2.

In this paper, I implemented a method for matching the corresponding 2D trajectories with another method that reconstructs 3D trajectories based on matched 2D trajectories. Also, different ways of using Kalman Filter for improving the pairing results are implemented. Synthetic data and starling data are used for the implementation, testing, and results.

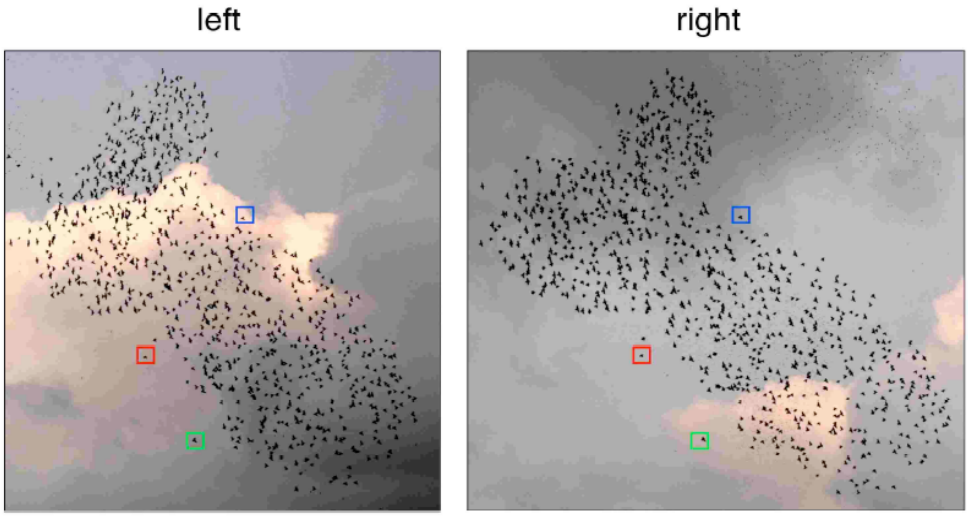


Figure 2: A real-world example scenario for matching process with two different camera views. Corresponding starlings are marked with the same color rectangles. Directly taken from the works of [2].

1.2 Related Work

1.3 Overview

This paper's implementation idea was oriented around those possible needs and use cases related to animal behavior research. For instance, experimenters may setup several cameras, and record the movements of the animals, such as starlings. The data collected from those cameras would be a 2D trajectory data. However, experimenters may want to work with 3D trajectory data in order to be able to study the movements of the animals. In order to achieve this, they would need to find the corresponding points to able to reconstruct the 3D movement data. The previous research for animal behavior and corresponding points matching have certain problems such as, they are for very specific cases or source code not available or the methods are manual or semi-manual. This paper's motivation is to provide a solution that eliminates the problems of previous works which were mentioned. The source code can be found in Appendix A. There are also some works that are quite close to the cover of this paper and they will be discussed in the related work section. The idea is to show the differences from their implementation or show the

similarities to the methods applied in this paper with different sets of data and varying approaches to the problems of corresponding point matching.

1.3.1 "AcinoSet: A 3D Pose Estimation Dataset and Baseline Models for Cheetahs in the Wild" [1]

The authors [1] provide a data set called "AcinoSet" which consists of cheetah data. They implement a 3D trajectory reconstruction method with the support of Extended Kalman Filter. Their idea is to analyze cheetah movements and reconstruct them in 3D which can provide a research base for robotic movements in the future [1].

Cheetah data can be viewed as challenging data because cheetahs are the fastest land mammals. Therefore, the data provides very fast and consistent movement data. The data is collected by using six cameras which are calibrated. The data provides 7500 hand-labeled 2D key points which were later used for reconstruction purposes. They also use multi-view triangulation for reconstruction. In addition, they also use "Full Trajectory Estimation" (FTE) for the feedback control. The data has pose parameters such as: head, neck, front and back torso, tail base and mid, shoulders, knees, back positions and angles [1]

They use collected data from multiple view synchronized cameras and make 2D marker-less pose estimations. Later, they implement 3D pose tracking step which consists of Triangulation, EKF and FTE. After, they implement 3D trajectory estimation. Their workflow can be seen in Figure 3

In my implementation, for a more realistic world results, starling data is used. Also, in my paper's implementation Standard Kalman Filter instead of Extended Kalman Filter. Therefore, my paper provides results for some comparison with Extended Kalman Filter and Standard Kalman Filter. Moreover, Joska, D et al. [1] use cheetah data which mostly moves in one direction. However, starling and synthetic data which are used for the results of my paper, move in many directions at different speeds, causing more challenging pairing process. Also, they use 2D

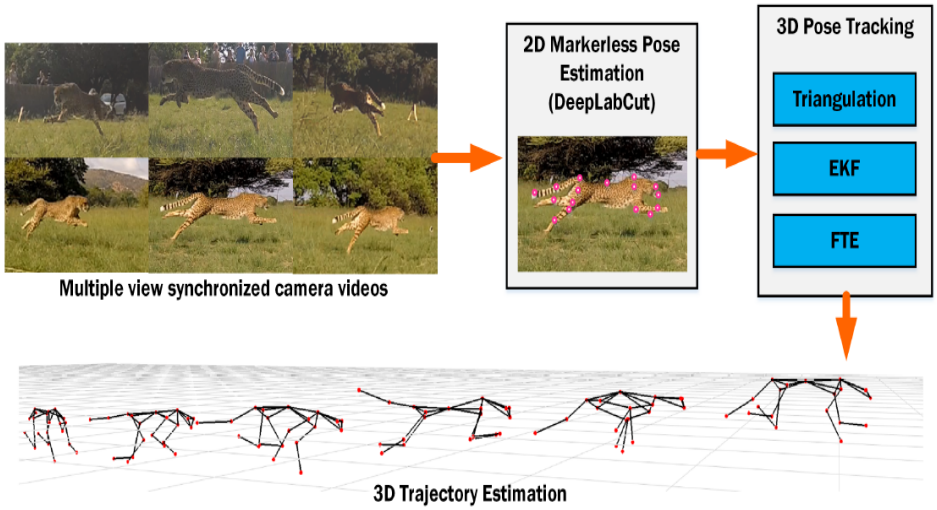


Figure 3: Diagram of the workflow for authors' [1] applied methods. Directly taken from the works of [1].

key-points, representing different body parts of the cheetahs. Therefore, they do not implement a pairing process for the cheetahs because they position information of the eyes, noise, and etc. In starling data which was used for my implementation, every starling is represented with a single dot. This creates a more challenging process since the pairing is done with less information.

1.3.2 "STARFLAG" Project [2]

Their work provides an overview of working with large amount of animals from a very distant observation point. The cameras were 25 meters apart they were about 100 meters away from the recorded birds [2]. Authors [2] used starlings flocks and this is a challenging case for animal behaviour studies.

In their implementation, they also used epipolar lines to find corresponding points. This is a common technique in the field and it performs well. They go through the techniques that they have used in a detailed manner. The authors [2] also implemented a 3D reconstruction method similar to my paper's implementation.

However, in my implementation, methods work with 2 calibrated cameras whereas their methodology works with 3 cameras. Although using 3 cameras is a good approach in order to solve the problems of corresponding point matching, it can

be costly. The setup requirements may matter for the researchers in the field of biology because the funding may be limited and 3 cameras requires more sources. In addition, 3 cameras solution can be useful for the distanced observations because the animals can be present in the field of views of all 3 cameras. However, this is not the case for up close recordings. For instance, bat behaviours can be recorded in caves. On the other hand, caves may not provide enough space for the positioning of 3 or more cameras where bats can be present in the field of view of the 3 cameras at the same time. My implementation uses 2 cameras, therefore, providing a low-cost and more flexible solution. The problems of finding the corresponding points are tried to be solved with the usage of Kalman Filter, similar to the implementation of Subsection 1.3.1. The further problems are tried to be solved with using the previous data of corresponding points. This process is called Re-run and will be introduced and discussed in the following chapters.

1.3.3 "Software techniques for two-and three-dimensional kinematic measurements of biological and biomimetic systems" [3]

The authors [3] mention about the previous work done in the field of the corresponding point matching for animal behavior. They mention that the previous works struggle to satisfy the requirements of other fields and are hard to modify for specific cases. Therefore, they try to provide a method that would require less effort to modify for specific cases. They also use epipolar geometry to identify the corresponding points and use the 3D reconstruction of the paired objects, similar to the cover of my paper. However, their software "DLTdv3" [3] requires some manual input from the user in order to work in some cases. They also mention that completely automatic object identification and 3D correspondence matching cases are possible but limited to controlled situations. The implementation requires the input from the user as a label or click on specific objects' landmarks such as the tip of a wing.

In my implementation, the manual inputs are limited to the configurations such as

threshold values. Therefore, provides an automatic system in order to identify the corresponding points. My current implementation only requires the 2D trajectories data and camera parameters as inputs. This can be particularly beneficial for animal behavior researchers as there can be many different real world scenarios. Also, authors [3] use MATLAB for their software, whereas, my paper's implementation uses Python and it can provide research groups the flexibility of customizing the software for their needs. Nevertheless, it should be mentioned that they provide a user interface which makes the user experience better.

Chapter 2

Methods

2.1 Overview

In this section methods which are used for the creation of results will be explained.

In order to be able to reconstruct 3D trajectories from 2D trajectories which are from 2 different frame cameras, where it is assumed 2D trajectories are known on both of the cameras but needs to be matched for 3D reconstruction, the following steps are used:

1. A point from the first camera at a given frame is projected to the second camera by using Fundamental Matrix [6].

2. The epipolar lines on both cameras are calculated. An example of these points can be seen in Figure 1 as P_L and P_R

3. Perpendicular distances of the possible candidate points to the epipolar line on the second camera are calculated. An illustration of this can be seen in Figure 5.

4. A point which has the smallest distance to the epipolar line is chosen as the best candidate where there is a predefined threshold value for the smallest distance. An example of this process can be seen in Figure 5

5. The point which is the best candidate on the image plane of the second camera is re-projected to the image plane of the first camera for two-way authentication.

6. If the best candidate on the first camera is the same point which was chosen at step 1, this is considered as possible corresponding points.

7. Possibly corresponding pair points are triangulated to calculate an estimated 3D point.

8. Kalman Filter prediction availability is checked.

   (a) If there is enough data for Kalman Filter calculation, calculate the distance of a predicted point from previous frames. Enough data is a reference to the threshold values. For the implementation, these are taken as 10 and 5 which means: in the last ten frames there should be at least five 3D estimated points to be able to make a forecast for the next frame.

   (b) If the distance between the forecasted point and the estimated point at step 7 is less than 30 centimeters, the points are paired. An example of this scenario can be seen in Figure 7.

9. If there is not enough data for Kalman forecast, the distance between epipolar lines and 2D points are checked for a threshold of 50 pixels. If the distances satisfy the threshold value, the points are paired.

This process is done for all frames, matching 2D points on both cameras where calibration of the cameras and 2D trajectories are known. A summarized version of the steps can be seen in Figure 4. In the end, the matched points' 2D trajectory id numbers are counted to find the most common matched trajectory ids on first and second cameras. This will provide matched corresponding 2D trajectories. By using epipolar geometry, where calibration information of the cameras is required, matched 2D trajectories are reconstructed in the 3D space of the world coordinate system. OpenCV (Open Source Computer Vision Library) [5] is used for the implementation part of the methods.

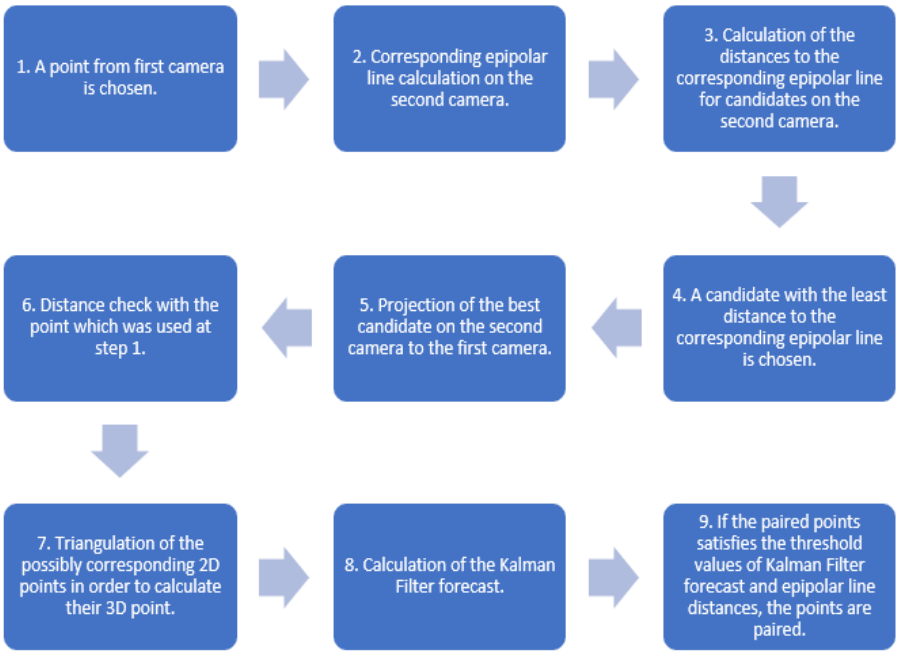


Figure 4: The workflow of applied pairing method.

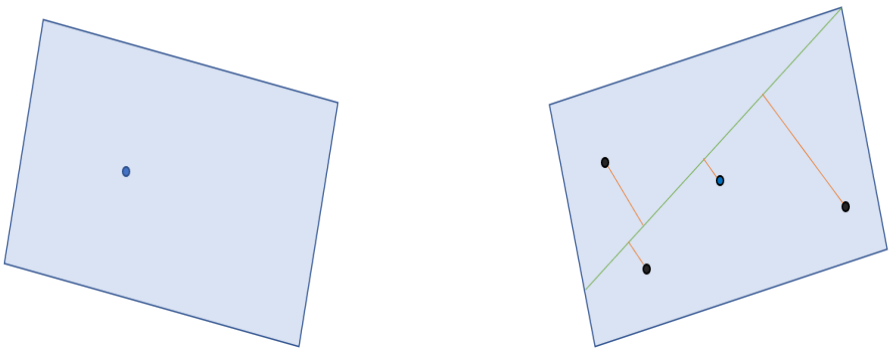


Figure 5: An example of epipolar distance pairing. A point from the left view (blue dot on the left camera plane) is chosen and its corresponding epipolar line (represented with green line) is calculated by using fundamental matrix on the second camera plane on the right. All of the possible candidates' (black dots on the right camera plane) distances (represented with orange lines) to the epipolar line are calculated. Corresponding point on the second camera is represented with black outer and blue inner circle.

2.2 Corresponding 2D Trajectory Matching

In this subsection, the details of the previously mentioned steps for 2D corresponding trajectory matching will be explained. For the epipole points calculation, we need to know the location of the cameras in 3D space, to be able to draw a line that goes through the image planes of both cameras. Epipole points are basically projected points of a camera on the other camera's image plane. Also, the projection of points from 3D to 2D will be frequently used in this implementation. In order to project a 3D point in the world coordinate system, on an image plane, the following steps are used.

The following formulas ((2.1) to (2.9)) are taken from [5] and explained at this part. First, a camera's intrinsic matrix A is needed which is consists of focal lengths of a camera f_x and f_y and principal points c_x and c_y.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Second, extrinsic parameters of a camera, rotation and translation matrices R and t respectively, are needed where P_W represents a 3D point in world coordinate system which will be transformed as P_c to 3D camera coordinate system.

$$P_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_W, \tag{2.2}$$

therefore,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.3}$$

with $x' = X_c/Z_c$ and $y' = Y_c/Z_c$,

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.4}$$

x' and y' are the 2D representations, however, with real cameras, there can be distortion effects and intrinsic parameters of the camera should be used to get the final 2D projection values. In order to be able to represent distortion effects, x'' and y'' are needed where they represent projection values after distortion. Finally, intrinsic parameters can be used with x'' and y'' to find final values u and v. The distortion parameters are categorized into radial coefficients $(k_1, k_2, k_3, k_4, k_5, k_6)$, tangential distortion coefficients (p_1, p_2), and thin prism distortion coefficients $(s1, s2, s3, s4)$ [5] [10].

Therefore,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix} \tag{2.5}$$

and,

$$r^2 = x'^2 + y'^2 \tag{2.6}$$

to get x'' and y'',

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x' \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1x'y' + p_2(r^2+2x'^2) + s_1r^2 + s_2r^4 \\ y' \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_2x'y' + p_1(r^2+2y'^2) + s_3r^2 + s_4r^4 \end{bmatrix} \tag{2.7}$$

and to get final values u and v by using intrinsic parameters,

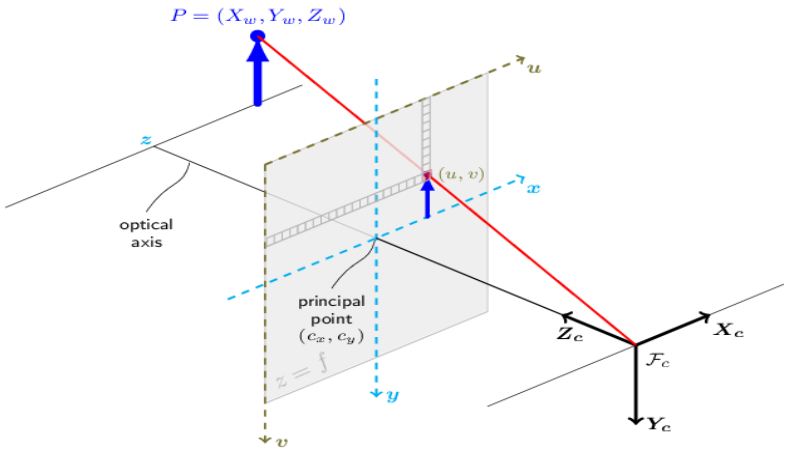


Figure 6: Pinhole camera model. Directly taken from the works of [5].

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix} \tag{2.8}$$

u and v values are projected values on the image plane of the camera. An illustration of this can be seen at Figure 6. [5]

For the previous implementation, the equations above are used to find epipole points. After, finding epipole points by projecting the camera centers into each other's image planes, 2D candidate points at each frame from Camera 1 should be projected onto the image plane of Camera 2. To do this, first, the points on the image plane of Camera 1 should be reversed to the step (2.2). This process can be reversible if intrinsic and extrinsic parameters of the camera are known and in the implementation part, these parameters are known. An arbitrary point Z_c should be defined at known depth because during the recording of the objects, the depth values (Z_c) of the objects are lost. During the implementation, a predefined depth value is given. Any depth value, within a reasonable distance, can be used and it will not affect the outcome as the reason was mentioned before in Chapter 1, moving the point p to p_1, p_2 or p_3 would reflect the same epipolar line on the right image plane. As it can be seen at (2.5) and (2.4) after giving a value for Z_c and since rotation matrix R and translation matrix t are known, X_w, Y_w and Z_w can be calculated representing a 3D point P_W in world coordinate system. Since the Z_c value was defined by the user,

it is not the original P_W, and will be referred to as P'_W. After a 2D point from the first image, plane is reconstructed in the 3D world coordinate system as P'_W, it can be reflected on the image plane of the second camera by following the same steps until 2.8. A 2D line on the second image plane which is going through the projected point and the epipole point on the image plane of the second camera is drawn. An example of this can be seen at Figure 1, on the right image a red line can be seen going through epipole point e_R and P_R.

With the updated implementation, the Fundamental Matrix is used in order to find the corresponding points. This approach provides more accurate and faster results.

Perpendicular distances to the epipolar line of all 2D points on the image plane of the second camera are calculated. A point with the shortest distance is chosen as the best candidate for the corresponding point on the first camera. After this process, a two-way matching method is implemented. For two-way matching: the best candidate is also projected to the image plane of the first camera. If it is matched with the same point which was projected on the image plane of the second camera to find the best candidate, these points are considered as two-way matched points.

This process is repeated for all points on the image plane of the first camera at all frames until every point has a match. However, in some cases, some of the points may not be matched and discarded if their distances to epipolar line are too far or they are not two-way confirmed. The reason why some points may be eliminated is due to their distances to the epipolar line can be noise with 2D points. The effect of noise will be discussed in the Results and Discussion sections in this paper. Since all of the provided 2D points have a predefined 2D trajectory id, finding the most commonly matched 2D trajectory ids for their 2D points at all frames, is the last step to match corresponding 2D trajectories.

2.3 Kalman Filter

Kalman filtering uses a series of input data, which may contain noise, error, etc., and produces estimations for the given states [11]. An example of this can be measuring the position of a moving object, such as a bird. If we assume that the bird has a GPS tracker, the position observations may include some errors and noise across multiple readings. Kalman filtering can use these observations to make more accurate estimations for the position of the object, in this case, the bird.

In addition, Kalman Filter can be used for forecasting purposes. In this paper, Kalman Filter is used for forecasting purposes, and for the implementation of standard Kalman Filter, OpenCV [12] [13] is used. The idea is to use Kalman filtering to make better predictions while pairing the objects. As it is explained in 2.1, the Kalman filter uses the previously paired objects' positions in order to make a forecast for the current frame. This provides filtering for the mis-pairings. An illustration of how Kalman Filter is used can be seen in Figure 7.

One problem of the epipolar matching is that there are some cases that contain multiple good candidates. This means the perpendicular distances of the candidates to the epipolar line are quite close and small. This causes wrong pairings of the objects and reduces the accuracy. To be able to make a better prediction of which object is which on the camera views, the possibly corresponding objects are filtered by the Kalman filtering. The results of this applied method will be discussed in the Results [3] chapter. In comparison to the Related Work 1.3.1, authors [1] use Extended Kalman Filter, whereas, this paper uses Standard Kalman Filter. For some of the parameters required for settings of Kalman Filter, the example from OpenCV [13] is used with little modifications.

2.4 3D Trajectory Reconstruction

After corresponding 2D trajectories on both image planes are matched, the next task is to reconstruct them to 3D world coordinate system with the purpose of accurately recreating recorded objects in 3D space. For this purpose epipolar geometry from [6]

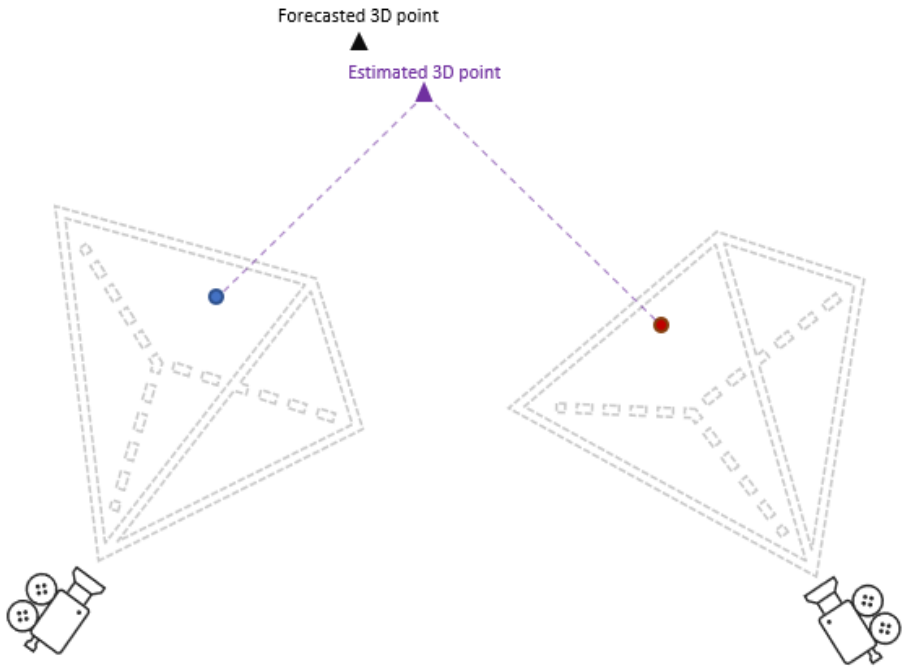


Figure 7: An example scenario where Kalman Filter forecast is used. Black triangle: A forecasted 3D point which was calculated by using Kalman Filter. Blue dot: A 2D point on the first camera. Red dot: Blue dot's corresponding 2D point on the second camera. Purple triangle: 3D triangulation of the two corresponding points (blue and red).

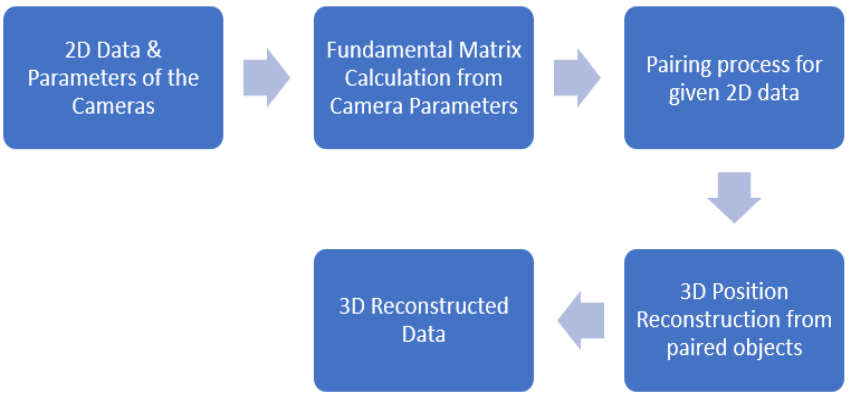


Figure 8: Diagram of the workflow for the applied methods.

is used. The following formulas ((2.11) to (2.15)) are taken from [6] and explained at this part. The required inputs are 3x4 projection matrices of the cameras and 2D trajectory points which are undistorted. A projection matrix of a camera is the matrix which projects a point in 3D world coordinate system to the image plane of a camera and can be defined by using $[R|t]$ and intrinsic matrix A as follows:

$$P = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{2.9}$$

and,

$$x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.10}$$

In this implementation, there are two cameras, therefore, P, P', x and x' are defined. There are many ways to find corresponding 3D point of x and x'. One example is drawing lines from camera centers which going through x and x' and their intersection would be their corresponding 3D position. Although this is a simple approach, the problem is that, in 3D space, most of the time lines never intersect due to noise. This requires another approach to reconstruct 3D points. OpenCV's triangulation function called "triangulatePoints" [5] which is used for the implementation, uses a linear triangulation method called Homogeneous Method (DLT)[6]. For this method, a 3D point representations of x and x' are needed, can be defined as:

$$\mathbf{X} = \begin{bmatrix} XYZ1 \end{bmatrix}^T \tag{2.11}$$

There should be an $\mathbf{X}$ where it satisfies $x = P\mathbf{X}$ and $x' = P'\mathbf{X}$. First, they eliminate homogeneous scale factor by using a cross product to get three equations for each

image point. An example is $x\mathrm{x}(P\mathbf{X}) = 0$ which gives:

$$x(p^{3T}\mathbf{X}) - (p^{1T}\mathbf{X}) = 0, \tag{2.12}$$

$$y(p^{3T}\mathbf{X}) - (p^{2T}\mathbf{X}) = 0, \tag{2.13}$$

$$x(p^{2T}\mathbf{X}) - y(p^{1T}\mathbf{X}) = 0 \tag{2.14}$$

where p^{iT} are the rows of P. For $\mathbf{X}$, in its components, these equations are *linear*. An equation of $A\mathbf{X} = 0$ can be composed as follows:

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \tag{2.15}$$

They demonstrate that the solution is the unit singular vector corresponding to the smallest singular value of A [6].

2.5 Iterative Information Process

After the results of the objects pairing are done, the accuracy of pairing can be improved by using the previous iterations. The idea is to use the information gathered at previous runs. In order to achieve this, the information from the previous runs is used for the next runs.

As it is mentioned at 2.1, Kalman Filter forecasting requires data from the last 10 frames and at least 5 of those frames should contain an estimated 3D position so that the current frame's position can be forecasted and compared to the estimation from pairing process. However, if the the data from previous iteration(s) are used, the next 10 frames can be used. In order to achieve this, if enough data cannot be found from the previous 10 frames, the next 10 frames are checked from the previous iteration(s). If there is enough data from the next 10 frames, this information can be used reversely and a forecast can be made for the current frame's pairing process.

This methodology will be referred to as "Re-run" (RR) for the Results 3 section.

2.5.1 Auxiliary: 2D Trajectory Classification

This method is implemented for a case where 2D trajectories may not be available and the only data provided is the 2D positions of detected objects in each frame. In this case, before corresponding 2D trajectory matching, 2D trajectories should be calculated/classified. A simple approach would be using a modified version of k-nearest neighbors algorithm (k-NN). It works by getting a point p_i at given frame f_n and looking for closest point at frame f_{n+1}. This process can be repeated for all points at frame f_n until all of the candidates at f_{n+1} are matched by assigning candidate points trajectory ids to their matched point's trajectory id at f_n, where it is assumed all of the points never leave the FOV of cameras and the number of 2D trajectories is known. Further implementation should be able to deal with objects leaving the FOV by assigning them a new 2D trajectory ids. This implementation is observed to be working well when none of the objects have an overlap when they are projected on image planes. However, it is observed that when there is an overlap of possible candidates at frame f_{n+1}, or possible candidates with similar distances at frame f_{n+1}, wrong 2D trajectory assignments occur which leads to temporal errors which means for the rest of frames wrong trajectory ids are assigned to the 2D points.

The two-way confirmation method, which was mentioned in section 2.2 can be a possible solution for these kinds of problems. When there are overlapping candidates or candidates which have similar distances smaller than the predefined threshold value, the information on the other camera can be used. For this, the current 2D point's p_i at f_n trajectory t_k needs be known. The previous points from $f_{n'}$ until f_n of t_k can be projected on the second camera's image plane, and the corresponding 2D trajectory method with two-way confirmation can be used. In order to achieve this, the 2D trajectories on the second camera should also be classified, from $f_{n'}$ to f_{n+2}. If there are no overlaps on the second camera at given frames, the results are expected to be accurate. Therefore, the classified 2D trajectories on the second camera should

be matched with corresponding trajectories with the points of the 2D trajectory t_k of the first camera with the method in Section 2.2. At f_{n+2} the 2D point of the matched corresponding 2D trajectory on the second camera, is projected on the first camera to find the best candidate by using epipolar line method which was mentioned at 2.2, and the best candidate's trajectory is classified as t_k at frame fn_{n+2}. By doing this, the potential wrong classification at f_{n+1} is ignored but the temporal errors can be solved if the matching process is accurate because the assigned point is at frame f_{n+2}. This method has the potential of solving the problems which were mentioned, if the trajectories on the second camera at frames from $f_{n'}$ to f_{n+2} can be correctly assigned.

There can be some cases where the second camera also does not have optimal classification conditions. In this case, using more than 2 cameras can provide an opportunity for a "chain" of confirmation for assigning points to accurate trajectories to increase the robustness.

Chapter 3

Results

3.1 Data Preparation

3.1.1 Synthetic Data

In order to test the implementations, random synthetic data is created. Real-world data is available, however, it is thought that to develop and test the methods with synthetic data simplifies the progression of finding potential errors, challenges and helps in systematically exploring the capabilities of the implemented methods and their algorithms. Random data for objects are defined with a minimum of 2 meters in front of the cameras without leaving the FOV of both cameras while keeping their distance to the cameras around 4 meters. For their movement, a sine function is used with adding a random value and division of frames per second (FPS) to achieve an average speed of around 10m/s which is a similar average speed of bats [14]. The created ground truth 3D data is projected onto image planes of the cameras. Camera 1 and Camera 2 were 10 degrees rotated towards each other with a 2 meters distance in between them, with Go Pro Hero 3 White parameters are assigned to them. The reason why Go Pro Hero 3 was used is that their parameters were publicly available. The results were calculated with random sets of ground truth data for the object for each of the results. For each, there were 600 frames at 30FPS. The number of created objects is mentioned in the results. An example of a created test setup with

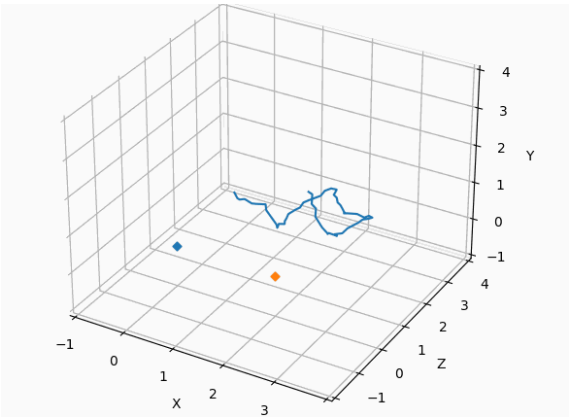


Figure 9: An example image of the implementation setup. Blue and orange diamonds represent the cameras. Blue line represents the 3D trajectory of a synthetic object.

2 cameras and an object's 3D trajectory can be seen in Figure 9.

The real-world scenarios may have errors in 2D object detection and in order to imitate that Gaussian noise is used. Therefore, sigma indicates the standard deviation value used for Gaussian noise which has zero mean distribution in pixels. For starling data results, there was no Gaussian noise process. The tests were done by using varying number of objects as it can be seen in Figure 10 and Table 1 and Table 3. Implementation is tested with 600 frames at 30FPS.

3.1.2 Starlings Data

Starling data has 3D tracking of 12 to 15 starlings information. The data was gathered in the "Barn" facility located in Radolfzell, Germany. The data is a subset from scientific experiments conducted at MPI-AB[1]. The barn had equipment of 30 cameras which were capable of tracking retro-reflective markers as small as 1mm accuracy. A motion capture system was used in order to collect the 3D motion of the starlings. The data has a realistic movement of the birds. They are close to each other and their movement trajectories overlap with each other in different cameras. For this paper's implementation, these 3D data are projected to 2 camera planes, creating 2D data. As you will see in the results, for some of the test cases, different noise levels were added to the projected data. The 3D data was produced by using

[1]Primary contribution from Nora Carlson, Mate Nagy.

marker tracking, and this causes the projection of 3D to 2D to be accurate.

Starlings data consists of 3D data that was gathered from behaviour experiment on starlings in controlled conditions. For the purposes of this paper's work, these 3D data are used and 2D trajectories are created by projecting the ground truth 3D point on camera views. This way reconstruction accuracy can be compared directly with the ground truth. This aims to imitate a set of 2D data collected from cameras. The 2D converted data has the properties of object id, x, y, frame values. These data are later used for the testing, and object ids with their representative data are evaluated and paired.

The reason why starling data was used for test cases is that it is challenging to be able to get ground truth 3D motion data of animals in the wild. Therefore, this causes the evaluation of the applied methods to be challenging. However, starling data still provides close data to a real-world scenario.

3.2 Evaluation of Corresponding 2D Trajectory Matching

As mentioned at 2.2, corresponding points on both of the image planes should be matched first in order to be able to match corresponding 2D trajectories. For this process a measure which shows successful results of corresponding points matching (SRCPM) percentage is shown in Figure 10. At each frame, corresponding points are matched with their best corresponding candidates. The percentage of the results indicates how successful was the implemented method after matching 2D points.

For Tables 1 and 3, pairing accuracy indicates how many of the pairing was done successfully. Used data ratio (UDR) indicates how many of the frames are used pairing. Some of the frames of the objects are not used. The reason why is that some candidates do not satisfy any of the pairing conditions and there are eliminated from the pairing process. In table 1 Kalman filter and Fundamental Matrix are used in order to show the difference of the methods which are used for the results. Kalman filter and Fundamental Matrix methods are used in the belief of

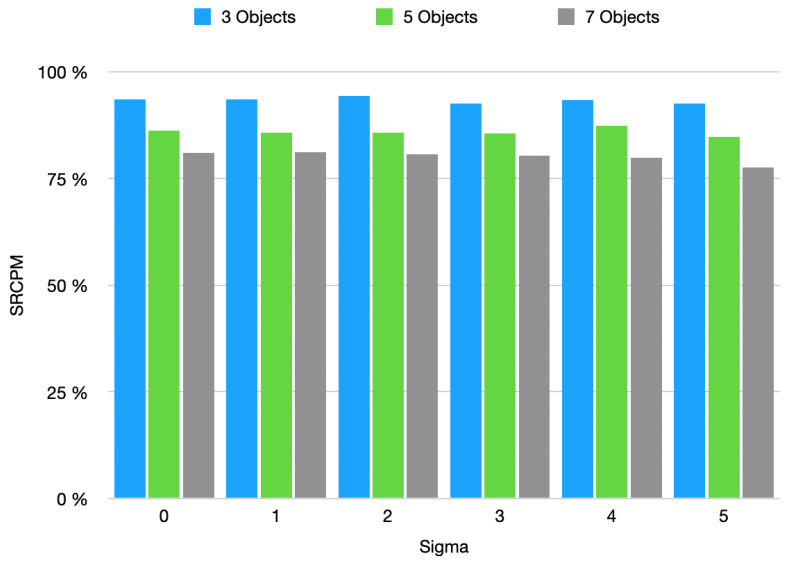


Figure 10: Successful results of corresponding points matching (SRCPM) percentage and used noise (σ) values during the corresponding 2D trajectory matching. The results were gathered with only using Epipolar Line Distance Pairing.

improving the previous results of Table 10. Kalman filter and Fundamental Matrix methodologies are not used for the previous results of Table 10. The comparison of previously and currently implemented methods can be seen in Table 1. Starling data is used as a test case scenario for the real world. Table 5 shows the animal id with pairing accuracy (PA) and used data ratio (UDR). Re-run (RR) is implemented after the "FM, ELDP" results, therefore, that means it takes the results of the implementation without Kalman Filter as an additional input. This creates a feedback loop that it can use the information from the previous run. This especially provides the advantage of forecasting by looking at the next frame pairing results from the previous run, and Kalman Filter makes a state forecast in a reverse approach.

For Tables 3, 5 different sets of data and methods are used. In order to make it more simple for the readers, the contents of the sets are provided in Tables 2 and 4. For Tables 3, 5, the numbers used for Pairing Accuracy and UDR are normalized to the range of 1.0 and 0.0 where 1.0 indicates 100% performance.

Number of Objects	Applied Methods	Pairing Accuracy
3	ELDP	0.92
5	ELDP	0.85
7	ELDP	0.80
3	KF, FM, ELDP	0.98
5	KF, FM, ELDP	0.97
7	KF, FM, ELDP	0.96

Table 1: Results comparison. KF: Kalman Filter, FM: Fundamental Matrix, ELDP: Epipolar Line Distance Pairing. Gaussian noise's Sigma (σ) value is taken as 3 for all of the results.

Set Name	Methods
Set 1	FM, ELDP
Set 2	KF, FM, ELDP
Set 3	KF, FM, ELDP, RR

Table 2: KF: Kalman Filter, FM: Fundamental Matrix, ELDP: Epipolar Line Distance Pairing, RR: Re-run.

Num	GNS	Applied Methods	Pairing Accuracy	UDR
20	0	Set 1	1.0	0.98
20	0	Set 2	1.0	0.64
20	0	Set 3	1.0	0.83
20	3	Set 1	0.87	0.86
20	3	Set 2	0.88	0.59
20	3	Set 3	0.96	0.51
20	5	Set 1	0.80	0.81
20	5	Set 2	0.79	0.58
20	5	Set 3	0.92	0.39

Table 3: Num: Number of synthetic objects, GNS: Gaussian noise Sigma (σ) value, UDR: Used Data Ratio. Same 20 object synthetic data is used for the results.

3.3 Evaluation of 3D Trajectories

During the 3D trajectory construction, errors are calculated in millimeters when comparing them to the ground truth values. If the pairing of corresponding points are accurate, then the reconstruction should be accurate by default. However, noise and camera calibration can cause errors during the reconstruction. At zero Sigma (σ), the minimum estimation error average was 0.0354mm and the maximum was 0.0367mm for the synthetic data which was used in Table 1. For Figure 3, at 0 Sigma (σ) average estimation error was 0.021mm, at 3 Sigma (σ) average error was

13mm, at 5 Sigma (σ) average error was 21mm. For the starlings data which is used for the result in Table 5, average 3D estimation error was 0.035mm.

Group Name of the Starlings	Animal IDs
Starlings 1	ST01a, ST02a, ST03a
Starlings 2	ST06b, ST07b, ST08a
Starlings 3	ST09b, ST10b, ST11b
Starlings 4	ST12a, ST14b, ST15b

Table 4: Animal IDs: Identification number of starlings from the starling data set.

Group	Applied Methods	PA	UDR	Group	Applied Methods	PA	UDR
Starlings 1	Set 1	1.0	0.90	Starlings 2	Set 1	1.0	1.0
Starlings 1	Set 2	1.0	0.90	Starlings 2	Set 2	1.0	0.99
Starlings 1	Set 3	1.0	0.90	Starlings 2	Set 3	1.0	1.0
Starlings 3	Set 1	1.0	1.0	Starlings 4	Set 1	1.0	0.94
Starlings 3	Set 2	1.0	0.99	Starlings 4	Set 2	1.0	0.90
Starlings 3	Set 3	1.0	1.0	Starlings 4	Set 3	1.0	0.94

Table 5: ID: Identification number of starlings from the starling data set, PA: Pairing Accuracy, UDR: Used Data Ratio.

3.4 Disclaimers

During the implementation and gathering of the results of starling and synthetic data, versions of Python 3.9, and OpenCV-python 4.5.2.54 are used. Further details can be found in Appendix Chapter A. The implemented methods are not tested with other versions. Therefore, with other versions, the results may differ.

In addition, the settings and the parameters during the implementations were heavily optimized with trial and error for the given data sets. The results may differ with other kinds of data. Therefore, further optimization methods and approaches may be needed.

Chapter 4

Discussion

4.1 Overview

In this section, the results of the implemented methods, the potential improvements and issues will be discussed.

4.2 Results

Applied methodologies showed varying results across synthetic and starling data. It should be noted that the synthetic data acts as if it is a "stress test". In starling data, the birds do not move at all times, and they do not make very sudden changes constantly. In synthetic data, the process of pairing the corresponding object is, therefore, more challenging as they are in constant movement and there are sudden direction changes. Also, this creates a further challenge for Kalman Filter testing due to the fact that Kalman Filter shows poorer results with an object which has very sudden state changes.

The first implementation tests were done only with Epipolar Line Distance Pairing (ELDP). Its results in Table 1 show that the accuracy was enough to identify the corresponding objects. However, later, the data was also tested with the updated version which was using Fundamental Matrix (FM). The previous approach was dis-

cussed in Section 2.2. The observation was that the Fundamental Matrix approach, alone, was enough to improve the results. However, Table 1 shows the results with Kalman Filter (KF) and Fundamental Matrix (FM). The results improvement is significant enough to move forward with an even more complicated data set (20 objects, Table 3) and a more realistic data set (starlings).

Table 3 shows that KF improves the accuracy considerably when it is used with previous pairing results. For the iteration of the pairing results, only FM and ELDP are used. When KF is used without previous iteration pairing results, it still shows an improvement but a smaller one. The variations or combinations can be changed and used in different ways and can provide better results and will be mentioned as a possible improvement to the applied methods. The cost of using KF with RR is that the UDR gets lower as it eliminates some of the pairing results from the previous runs with more confidence. On the other hand, this approach can still be useful since it provided results with more accuracy. Therefore, it can be used for improving the confidence of the results by cross-checking with other results. KF proves to be useful even when the noise levels are increased which is a quite good thing considering the noise level causes more 3D reconstruction errors which leads to a harder process of forecasting the positions of the objects.

For more realistic results, starlings data was used and the results can be seen in Table 5. The results were very accurate due to the fact that the starlings do not move all the time and stay idle for long periods of time. However, the results show that the applied methods can be further implemented in real-world scenarios even if the movements of the animals are more complex. The performance of the Set 3 methods did not show poor performance with UDR, unlike synthetic data results. This is probably due to the starling data is being more realistic. The synthetic data objects move fast, in many directions, and can have sudden direction changes. This creates a very challenging process for KF forecasts. In starling data, this is not the case most of the time, and therefore UDRs were high for the starling data.

UDR also seems to be constant or very similar in Table 4. This is due to methods working with very high accuracy with given data. Also, in some cases, starling leaves

the FOV of the cameras. If a starling leaves the FOV of one camera and it is in the FOV of the other camera, the bird is still tried to be paired, however, when the algorithm cannot find a proper candidate from all possible candidates, the bird is not paired and it leads to a lower UDR.

Chapter 5

Conclusion

5.1 Issues

Results show that the noise levels can have a significant effect on the accuracy of pairing corresponding points. However, the Gaussian noise on the synthetic data may not be a very good indicator for the real-world results. In fact, the results may indicate an underestimation of the applied methodologies in this paper. Therefore, applied methods may perform better with other real-world data as well. Starling data was recorded in a real-world scenario with real cameras and the applied methods showed promising results. However, this also does not show that methods will perform with the same accuracy in all real-world cases. For instance, starling data has its own quirks. Starlings are mostly idle and there are a few flight sequences, so, more real-world scenarios should be tested with more data-sets for further thoughts on the success of the applied methods.

The pairing process is currently slow due to the lack of optimization of the code. It can take a long for a user to analyze a few animals in thousands of frames. However, for instance, there are some use cases where flocks of birds are analyzed and there are thousands of birds with a high density. For these use cases, the results may take more than several hours or days. Thus, an optimization is currently necessary for further applications.

In some cases, the pairing process of the corresponding points may perform poorly due to some specific cases, noise, or other variables. For some of these problems, using an additional camera or maybe more can solve the issues of the poor performance in certain frames. However, the current application of the applied methods does not support using more than 2 cameras. This aspect of the applied algorithms can be improved for scenarios where more than 2 cameras are available.

5.2 Possible Improvements

5.2.1 Reactive Epipolar Line Matching

During the pairing process, the epipolar line distance pairing method is used. An example of this can be seen in Figure 11. However, the implementation works with a constant pixel threshold value. In some cases, the threshold value becomes unnecessary because it is too high compared to the average of paired corresponding points. Therefore, a reactive epipolar line distance threshold can be used for a better functioning threshold value. In order to do this, after the first pairing process, the distances to the epipolar lines can be analyzed similar to the iterative approach Re-run. After that, a new and possibly better-performing threshold value can be determined.

This addition to the implementation may increase the accuracy of the pairing. Also, since it can be an automated process, it can increase the adaptiveness of the methods applied in this paper can improve for different kinds of data sets.

5.2.2 Further Iterations

The results with the Re-run approach show promising results. The combination of different methods and order of the applied methods can be changed and better results may appear. Especially, further iterations by using previous results can be beneficial and may increase the accuracy of pairing.

In addition, this concept can be improved beyond the current application. Using

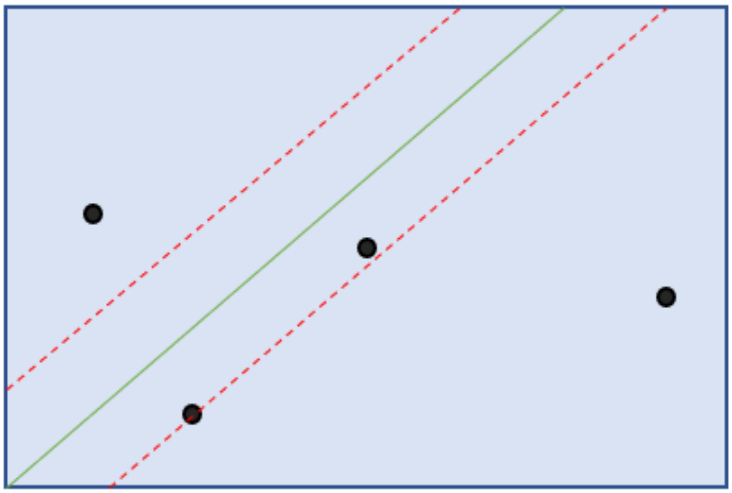


Figure 11: An example of epipolar distance pairing. Black dots: All possible candidates. Green line: Epipolar line. Dashed red lines: The distance threshold for the possible candidates.

previously paired data may also assist the epipolar line distance matching by calibrating the threshold values used, resulting in a reactive threshold value for the pairing.

Furthermore, using Kalman Filter for finding the outliers in the previously paired point can improve the next pairing processes due to the elimination of the possibly incorrect pairings. Kalman Filter can be used for forecasting the next state, therefore, this process can be used for the analysis of the previously paired points. Any point that is not within the boundaries of the desired threshold value of the user, can be eliminated or removed from the results. This approach also provides reverse forecasting so the possibility of improving the pairing accuracy of corresponding points seems promising if applied.

5.3 Conclusion

In this paper, I implemented a different combination of methods for 2D trajectory matching and 3D trajectory reconstruction. The results showed that noise may affect the reconstruction process and may cause reconstruction errors, and the number of objects in the field of the views may decrease the performance of 2D trajectory

matching. With both synthetic data and starlings data, the implemented methods achieved the purpose as they successfully matched 2D trajectories and reconstructed them in a 3D world coordinate system when there is not too much noise. The results without noise were even more promising.

The results and the evaluation showed that the applied methods can be further tested with other data sets and real-world data. The accuracy of corresponding pair matching was improved by using Kalman Filter. Also, it was shown that using previously matched points as input to Re-run the Kalman Filter increases the matching accuracy. There are still possible improvements to the applied methods and even higher accuracy levels seem to be possible.

This paper's implementation tries to solve some aspects of the related works by providing source code, not requiring manual user interaction, and can be implemented with only using 2 cameras. Therefore, this paper's implementation can be interesting to the researchers who work with birds, fishes, bats, starlings or even groups of other moving objects.

Appendix A

Appendix

A.1 Used packages for the implementation

1. python: 3.9
2. numpy: 1.20.2
3. pandas: 1.2.4
4. matplotlib: 3.4.2
5. opencv-python: 4.5.2.54
6. scipy: 1.6.3

A.2 Source Code

https://github.com/Giraytd/tproject

Bibliography

[1] Joska, D. *et al.* Acinoset: A 3d pose estimation dataset and baseline models for cheetahs in the wild. *arXiv preprint arXiv:2103.13282* (2021).

[2] Cavagna, A. *et al.* The starflag handbook on collective animal behaviour: Part i, empirical methods. *arXiv preprint arXiv:0802.1668* (2008).

[3] Hedrick, T. L. Software techniques for two-and three-dimensional kinematic measurements of biological and biomimetic systems. *Bioinspiration & biomimetics* **3**, 034001 (2008).

[4] Chotrov, D., Uzunova, Z., Yordanov, Y. & Maleshkov, S. Mixed-reality spatial configuration with a zed mini stereoscopic camera. In *Conf. Techno. Edu. Smart World*, vol. 11 (2018).

[5] Camera calibration and 3d reconstruction. *OpenCV* (2021). URL `https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d`.

[6] Hartley, R. & Zisserman, A. *Structure Computation* (Cambridge University Press, 2004), 2 edn.

[7] Opencv. *OpenCV* (2021). URL `https://opencv.org/`.

[8] Zhang, Z. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision* **27**, 161–195 (1998).

[9] Welch, G., Bishop, G. *et al.* An introduction to the kalman filter (1995).

[10] Louhichi, H., Fournel, T., Lavest, J. & Aissia, H. B. Self-calibration of scheimpflug cameras: an easy protocol. *Measurement Science and Technology* **18**, 2616 (2007).

[11] Wikipedia. Kalman filter — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Kalman%20filter&oldid=1063755317` (2022). [Online; accessed 14-January-2022].

[12] OpenCV. Kalman filter class reference. `https://docs.opencv.org/3.4/dd/d6a/classcv_1_1KalmanFilter.html` (2022). [Online; accessed 14-January-2022].

[13] OpenCV, A. O. A. P., Alexander Alekhin. Kalman.py. `https://github.com/opencv/opencv/blob/master/samples/python/kalman.py` (2022). [Online; accessed 14-January-2022].

[14] Theriault, D. *et al.* Reconstruction and analysis of 3d trajectories of brazilian free-tailed bats in flight. In *20th Int. Conf. on Pattern Recognition*, 1–4 (2010).