Multi-View 2D and 3D Trajectory Optimization

Author: Giray Tandoğan
University of Konstanz
Konstanz, Germany

Supervisor: Hemal Naik
Max Planck Institute of Animal Behavior,
Technical University of Munich, and University of Konstanz

Co-supervisor: Thejasvi Beleyur
Acoustic and Functional Ecology, and
Max-Planck Institute for Ornithology

ABSTRACT

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

KEYWORDS

epipolar geometry, trajectory optimization

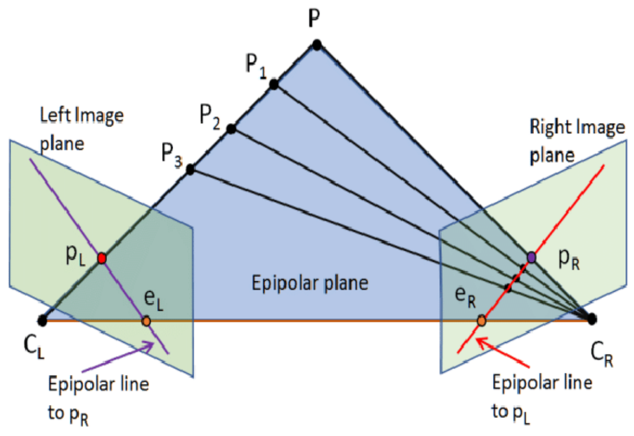

Figure 1: Epipolar geometry of stereo vision. [3]

1 INTRODUCTION

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, the 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 rotations, 2D points can be reflected back into 3D space. A common technique which can be used to reconstruct 2D points into 3D space is to use epipolar geometry. An example of epipolar geometry can be seen at Figure 4.

As it can be seen at Figure 4, a point P is in the field of view (FOV) of both cameras. The location of point P on 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 same location on the left image plane which is p_L even though it is moved in 3D space. As it is mentioned before, we would need accurate depth information in order to able to reconstruct object in 3D space. Thus, we can use the information on the right image plane which would provide a depth information if we know the exact position on the right image plane. If we do not know exact location on of point P on the right image plane, we can assume that it should be on a line which goes through e_R and p_r. How to get the location of e_R and other details will be discussed at section 3. Therefore, if a point's location on both image planes are known, it is possible to estimate the position of the points in 3D. This process can be called "reconstruction" of a 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 lines. If 2D temporal trajectory data 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 reflect 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 [3][2][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 lines may not be sufficient to match correct points. Mismatched point would create high 3D estimation errors. Therefore, when trying to match 2D trajectories with their corresponding 2D trajectories on other images planes, we can try to match multiple points of the those 2D trajectories at different frames.

In this paper, I implemented a method for matching the corresponding 2D trajectories with another method which reconstructs 3D trajectories based on matched 2D trajectories.

2 RELATED WORK

2.1 Epipolar Arrangement of Satellite Imagery by Projection Trajectory Simplification [7]

In their paper, an efficient method for automatically generating approximate epipolar images from linear push-broom satellite imagery is presented. This algorithm's core technique is a projection trajectory simplification strategy, in which a pair of straight lines is used instead of the projection trajectory approach to approximate

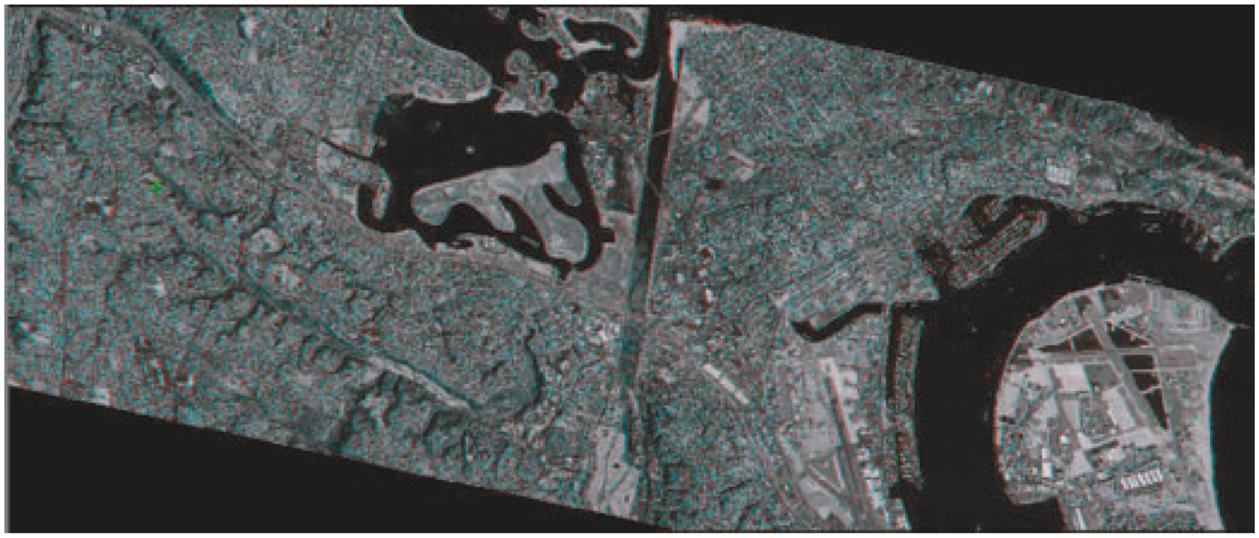

Figure 2: Stereo anaglyph of the generated epipolar image provided by using images from IKONOS satellite. [7]

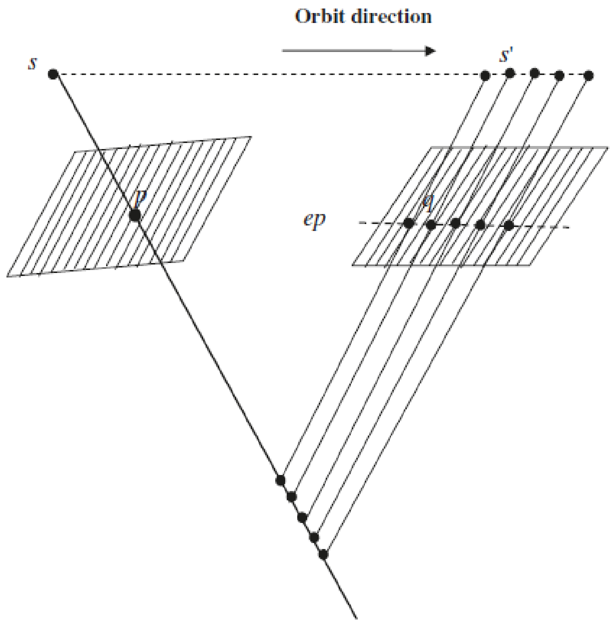

Figure 3: Linear pushbroom satellite images. [7]

present each pair of conjugate epipolar curves defined by the projection trajectory approach with respect to geometric characteristics of epipolar curves in the satellite image scene. This method allows for high efficiency approximate epipolar image generation from satellite stereo-images.

First, an alternative linear epipolarity model is used, in which straight lines are used to present each pair of conjugate epipolar curves in satellite image scenes based on their geometric properties; second, a practical algorithm for epipolar arrangement of satellite images is provided, that is, generating approximate epipolar images of satellite stereo-images using projection trajectory simplification. Third, by testing the proposed algorithm on various satellite images, to determine its feasibility and robustness; and fourth, to determine the optimal number of selected sample points used for line fitting, in order to achieve a balance between their algorithm's efficiency and accuracy.

In my implementation, the cameras are positioned stationary, whereas, their source of data is satellites which move in the orbit. This results with pushbroom satellite imagery as it can be seen in Figure 3. This creates a more challenging task to be able to match corresponding points from multiple views. In this paper, the aim is to reconstruct 3D trajectories which requires multiple matching 2D points on temporal frames. However, their method aims to automate generation of approximate epipolar images from linear pushbroom satellite imagery. This process can provide anaglyph images, used create a stereoscopic 3D effect which can create an illusion of depth to a flat image. An example of this can be seen at Figure 2.

3 METHODS

At this section methods which are used for the creation of results will be explained. To start with, OpenCV (Open Source Computer Vision Library) is used for the implementation part of the methods. It has more than 2500 optimized algorithms are included in the library, which includes both classic and state-of-the-art computer vision and machine learning algorithms [2][1].

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, following steps are used:

(1) The epipole points on both cameras are calculated. An example of these points can be seen at Figure 4 as e_L and e_R
(2) A point from the first camera at a given frame is projected to 3D space with a predefined depth value.
(3) The point which is projected to 3D space is re-projected to the image plane of the second camera.
(4) The epiolar line on the second camera is calculated by using the points:

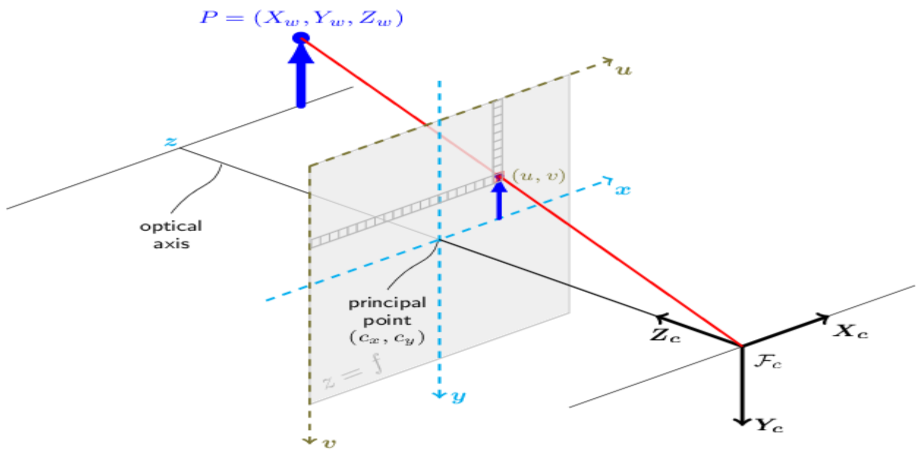

Figure 4: Pinhole camera model. [1]

   (a) The point which is re-projected to the image plane of the second camera.
   (b) The epipole point on the image plane of the second camera.
(5) Perpendicular distances of the points to the epipolar line on the second camera are calculated.
(6) 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.
(7) 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 by again giving a defined depth value to move 3D space and re-projecting it on the image plane of the first camera.
(8) If the best candidate on the first camera is the same point which was chosen at step 2, this is considered as matched corresponding points.

This process is done for all frames, matching 2D points on both cameras. At 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, matched 2D trajectories are reconstructed in 3D space of world coordinate system.

3.1 Corresponding 2D Trajectory Matching

At 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 which goes through image planes of the both cameras. Epipole points are basically projected point of a camera on the other camera's image plane. Also, projection of points from 3D to 2D will be frequently used in this implementation. In order to project a 3D point in world coordinate system, on a image plane, following steps are used.

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.

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

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.

$$P_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} P_W, \tag{2}$$

therefore,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3}$$

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

$$Z_c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = [R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4}$$

$x' andy'$ are the 2D representations, however, with real cameras, there can be distortion effects and intrinsic parameters of the camera should be used to get 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 $(s1, s2, s3, s4)$ [1] [5].

Therefore,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix} \tag{5}$$

and,

$$r^2 = x'^2 + y'^2 \tag{6}$$

to get x'' and y'',

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x' \frac{1+k_1 r^2 + k_2 r^4 + k_3 r^6}{1+k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2(r^2 + 2x'^2) + s_1 r^2 + s_2 r^4 \\ y' \frac{1+k_1 r^2 + k_2 r^4 + k_3 r^6}{1+k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_2 x' y' + p_1(r^2 + 2y'^2) + s_3 r^2 + s_4 r^4 \end{bmatrix} \tag{7}$$

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

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix} \tag{8}$$

u and v values are projected values on the image plane of the camera. An illustration of this can be seen at Figure X. [1]

For the implementation, the equations above are used to find epipole points. After, finding epipole points by projecting the cameras into each others image planes, 2D candidate points at each frame from Camera 1 should be projected on to image plane of Camera 2. To do this, first, the points on image plane of Camera 1 should be reversed to the step (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. At this point, a depth value Z_c should be defined, 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 reasonable distance, can be used and it should not affect the outcome as the reason was mentioned before at Section 1, moving the point p to p_1, p_2 or p_3 would reflect to the same epipolar line on the right image plane. As it can be seen at (5) and (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, and it is not the original P_W, and will be referred as P'_W. After a 2D point from the first image plane is reflected into 3D world coordinate system as P'_W, it can be reflected on image plane of the second camera by following the same steps until 8. A 2D line on the second image plane which is going through 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 4, on the right image a red line can be seen going through epipole point e_R and P_R. Please note, there is an easier way of finding epipolar line by using fundamental matrix, however, it is left as a future work.

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 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 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 is too far or they are not two-way confirmed. The reason why some points may be eliminated due to their distances to the epipolar line can be noise with 2D points. The effect of noise will be discussed at 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.

3.2 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 is used. 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:

$$P = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{9}$$

and,

$$x = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10}$$

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 which is used for the implementation, uses a linear triangulation method called Homogeneous Method (DLT)[4]. For this method, a 3D point representations of x and x' are needed, can be defined as:

$$\mathbf{X} = \begin{bmatrix} XYZ1 \end{bmatrix}^T \tag{11}$$

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:

$$x(p^{3T}\mathbf{X}) - (p^{1T}\mathbf{X}) = 0, \tag{12}$$

$$y(p^{3T}\mathbf{X}) - (p^{2T}\mathbf{X}) = 0, \tag{13}$$

$$x(p^{2T}\mathbf{X}) - y(p^{1T}\mathbf{X}) = 0 \tag{14}$$

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:

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \tag{15}$$

They demonstrate that the solution is the unit singular vector corresponding to the smallest singular value of A [4].

3.3 Auxiliary: 2D Trajectory Classification

This implementation method is not in the scope of this paper and needs further testing and research but thought to be worth mentioning. This method is implemented for a case where 2D trajectories may not be available and the only data provided are x, y coordinates with camera id and frame number. 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 are known. The further implementation should be able to deal with objects leaving the FOV by assigning them a new 2D trajectory ids. This implementation 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 at section 3.1 can be a possible solution for these kinds of problems. When there are overlapping candidates or candidates which have similar distances smaller than 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 at section 3.1. 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 3.1, 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 a 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 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. As mentioned before, this method is not in the scope of this paper and left as a potential future work.

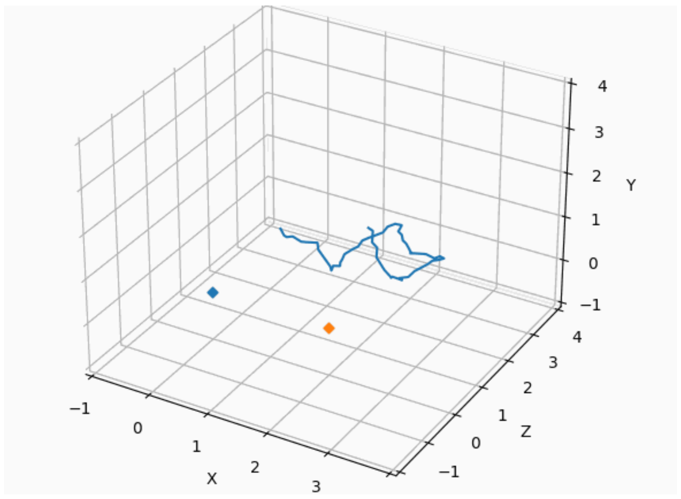

Figure 5: An example image of the implementation setup.

4 RESULTS

For the results, different ground truth data are used with 3, 5 and 7 number of objects resulting with a total number of 3 different ground truth data. The details of data creation process and setup can be found at Section 4.1. In order to measure the performances, Gaussian noise is added to projected 2D points. The performance with and without Gaussian noise will be discussed at following sections.

4.1 Data Creation

In order to test the implementation, random synthetic data is created. Random data for objects are defined with minimum 2 meters in front of the cameras without leaving the FOV of the both cameras while keeping the 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 [6]. 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 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. Number of created objects are mentioned at results. An example of created test setup with 2 cameras and an object's 3D trajectory can be seen at Figure 5.

4.2 Corresponding 2D Trajectory Matching

As mentioned at 3.1, 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 at Figure 6. Sigma indicates the standard deviation value used for Gaussian noise which has zero mean distribution.

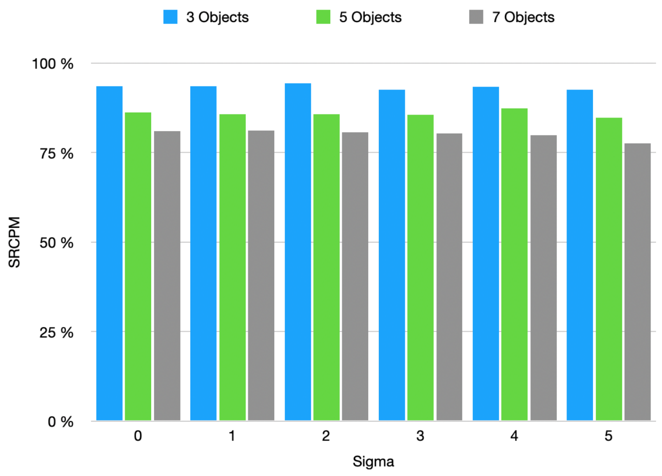

Figure 6: Successful results of corresponding points matching (SRCPM) percentage and used sigma values during the corresponding 2D trajectory matching.

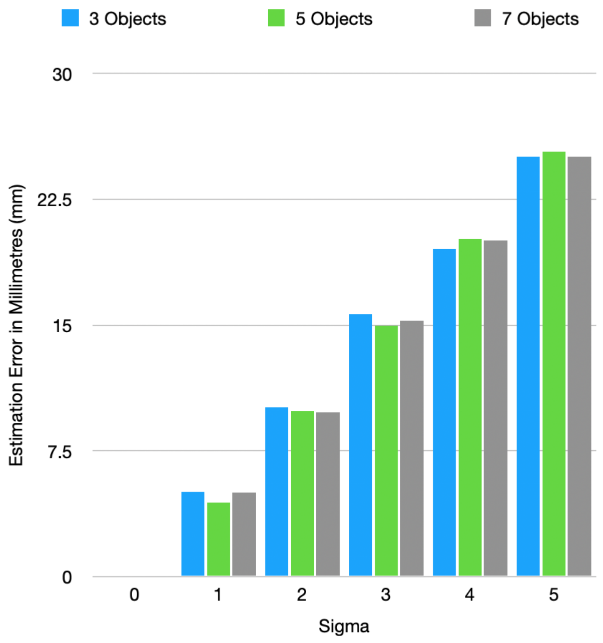

Figure 7: Average estimation errors in millimeters and used sigma values during the reconstruction of 3D trajectories.

4.3 3D Trajectory Construction

During the 3D trajectory construction, errors are calculated in millimeters when comparing them to the ground truth values. At Figure 7 an illustration of estimation errors are shown with respect to the used standard deviation value for Gaussian noise which has zero mean distribution.

5 DISCUSSION

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

5.1 Disclaimers

During the implementation and gathering the results, versions of Python 3.9, and OpenCV-python 4.5.2.54 are used. The implemented methods are not tested with other versions. Therefore, with other versions, the results may differ.

Furthermore, due to using randomly generated ground truth data, implemented methods may or may not perform as expected in real world. Using real world data, such as collected data from birds or bats, is planned to be future work of the implementations.

5.2 Issues

As it can be seen at Figure 6, the implemented method for 2D trajectory matching, the performance is dropped each time amount of objects in FOVs is increased. This issue is due to the points' perpendicular distances to the epipolar lines when there are many candidates are present, which leads to mismatches.

As it can be seen at Figure 7, estimation error of 3D of the points, which belong to the 3D trajectories, are increasing linearly. At zero sigma, the minimum estimation error average was 0,0354mm and maximum was 0,0367mm.

5.3 Possible Improvements

Although the performance of corresponding 2D trajectory matching method was enough for identify the corresponding trajectories, by using velocity of the objects, these results can be improved because the velocity information would provide an expected region for objects' position at next frame. As it was mentioned at Section 3.1, there is a better way of calculating epipolar line by using fundamental matrix and planned as future work.

6 CONCLUSION

In this paper, I implemented 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 amount of objects in FOVs may decrease performance of 2D trajectory matching. The implemented methods performance were satisfactory, however, they need real world testing because the implemented methods aim to solve real-world problems.

Finally, with synthetic data, the implemented methods achieved the purpose as they successfully matched 2D trajectories and reconstructed them in 3D world coordinate system when there is not too much noise. The possible solutions for improving reconstruction results will be investigated. For the 2D trajectory matching method, the results indicate that if number of objects are increasing, the matching performance will be decreasing. In order to be able to solve this problem, a velocity method or a more reactive matching method can be implemented.

REFERENCES

[1] 2021. Camera Calibration and 3D Reconstruction. *OpenCV* (2021). https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d
[2] 2021. OpenCV. *OpenCV* (2021). https://opencv.org/
[3] D Chotrov, Z Uzunova, Y Yordanov, and S Maleshkov. 2018. Mixed-reality spatial configuration with a zed mini stereoscopic camera. In *Conf. Techno. Edu. Smart World*, Vol. 11.
[4] Richard Hartley and Andrew Zisserman. 2004. *Structure Computation* (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511811685.017

[5] H Louhichi, T Fournel, JM Lavest, and H Ben Aissia. 2007. Self-calibration of Scheimpflug cameras: an easy protocol. *Measurement Science and Technology* 18, 8 (2007), 2616.

[6] Diane Theriault, Zheng Wu, Nickolay I Hristov, Sharon M Swartz, Kenneth S Breuer, Thomas H Kunz, and Margrit Betke. 2010. Reconstruction and analysis of 3D trajectories of Brazilian free-tailed bats in flight. In *20th Int. Conf. on Pattern Recognition*. 1–4.

[7] Mi Wang, Fen Hu, and Jonathan Li. 2010. Epipolar arrangement of satellite imagery by projection trajectory simplification. *The Photogrammetric Record* 25, 132 (2010), 422–436.

[8] Zhengyou Zhang. 1998. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision* 27, 2 (1998), 161–195.