

## ABSTRACT

Title of Dissertation:     **INERTIALLY CONSTRAINED  
RULED SURFACES FOR  
VISUAL ODOMETRY**

Chenqi Zhu  
Master of Science, 2024

Dissertation Directed by: **Professor Yiannis Aloimonos  
Department of Computer Science**

In computer vision, camera egomotion is typically solved with visual odometry techniques that relies on feature extraction from a sequence of images and computation of the optical flow. This, however, often requires a point-to-point correspondence between two consecutive frames which can often be costly to compute and its varying accuracy greatly affects the quality of estimated motion. Attempts have been made to bypass the difficulties originated from the correspondence problem by adopting line features and fusing other sensors (event camera, IMU), many of which still heavily rely on feature detectors. If the camera observes a straight line as it moves, the image of such line is sweeping a surface, this is a ruled surface and analyzing its shapes gives information about the egomotion. This research presents a novel algorithm to estimate 3D camera egomotion from scenes represented by ruled surfaces. Constraining the egomotion with inertia measurements from an onboard IMU sensor, the dimensionality of the solution space is greatly reduced.

INERTIALLY CONSTRAINED RULED SURFACES  
FOR VISUAL ODOMETRY

by

Chenqi Zhu

Thesis submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Master of Science  
2024

Advisory Committee:

Professor Yiannis Aloimonos, Chair/Advisor

Professor Christopher Metzler

Dr. Cornelia Fermüller

© Copyright by  
Chenqi Zhu  
2024

## Acknowledgments

First and foremost I would like to thank my advisor, Professor Yiannis Aloimonos, as well as Dr. Cornelia Fermüller, for giving me an invaluable opportunity to work on challenging and extremely interesting projects over the past two years. Thanks are also due to Professor Christopher Metzler for agreeing to serve on my thesis committee and for sparing his invaluable time reviewing the manuscript.

Furthermore, I would like to express my most sincere gratitude to Levi S. Burner, a PhD student at the University of Maryland, whose guidance and insights were instrumental to the completion of this thesis. His contributions in providing valuable ideas and helping to set up key experiments, along with his camera calibration tools, were essential to the progress of my research. Additionally, the foundational theories presented in this work are grounded in concepts originally proposed in his PhD proposal, without which this thesis would not have been possible.

I would also like to acknowledge the friends and colleagues at PRG for their support and assistance. Their help in setting up experiments and their valuable suggestions for improving my presentations greatly contributed to refining both my thesis and my oral exam.

Lastly, I owe my deepest thanks to my mother and father who have always stood by me through my career, and have supported me from the other side of the Pacific Ocean at both good and hard times.

## Table of Contents

Acknowledgements	ii
Table of Contents	iii
List of Tables	v
List of Figures	vi
Chapter 1: Introduction	1
1.1 Background . . . . .	1
1.2 Related Works . . . . .	3
Chapter 2: Problem Formulation	7
2.1 Ruled Surface . . . . .	7
2.2 Parameterization . . . . .	9
2.3 Ruling Reprojection Loss . . . . .	12
2.4 Dimensionality . . . . .	13
Chapter 3: Methodology	15
3.1 Initial Surface Estimation . . . . .	15
3.2 Extruding Surface into Future Frames . . . . .	16
3.3 Sliding Window . . . . .	17
3.4 Recover Camera Egomotion and Scene Geometry from Ruled Surface . . . . .	18
Chapter 4: Experiments	20
4.1 Setup . . . . .	20
4.2 Surface Estimation . . . . .	20
4.3 Derotation . . . . .	21
Chapter 5: Results	22
5.1 Overview . . . . .	22
5.2 Discussion . . . . .	25
5.2.1 Rotation-less Motions . . . . .	25
5.2.2 Motions with Rotations . . . . .	28
5.2.3 Non-smooth Motions . . . . .	29
5.2.4 Non-coplaner Scenes . . . . .	31
5.2.5 Free Motions . . . . .	31

5.3	Limitation and Future Work . . . . .	33
5.4	Conclusion . . . . .	34
	Bibliography	35

## List of Tables

5.1	Results for Rotation-less Motions . . . . .	25
5.2	Results for Motions with Rotations . . . . .	28
5.3	Results for Non-smooth Motions . . . . .	29
5.4	Results for Motions on Non-coplanar Scenes . . . . .	31
5.5	Results for Free Motions . . . . .	32

## List of Figures

2.1	Ruled Surface in Experiments . . . . .	7
2.2	Ruled Surface . . . . .	8
2.3	Ruled Surface Construction . . . . .	9
3.1	Edge Map . . . . .	15
5.1	Coplanar Scenes with Linear Motions . . . . .	22
5.2	Coplanar Scenes with Circular Motions . . . . .	23
5.3	Coplanar Scenes with Non-smooth Motions . . . . .	24
5.4	Experiment Setup for Non-coplanar Scenes . . . . .	25
5.5	Results for Linear Motions Parallel to Coplanar Scenes . . . . .	26
5.6	Results for Linear Motions Perpendicular to Coplanar Scenes . . . . .	27
5.7	Estimated Egomotions with Different Types of Motions . . . . .	27
5.8	Estimated Egomotions with Rotations . . . . .	29
5.9	Estimated Egomotions with Non-smooth Motions . . . . .	30
5.10	Results for Linear Non-smooth Motions . . . . .	30
5.11	Results for Rotating Circular Motion on Non-coplanar Scenes . . . . .	32
5.12	Limitation . . . . .	33

## Chapter 1: Introduction

### 1.1 Background

The task of egomotion estimation is of great significance. An autonomous agent's abilities to localize, such as determining the position of drones or vehicles relative to buildings or street signs, have consequential real world applications. This set of problems, where a 3D structure of the environment along with the motion signal of the agent is reconstructed from observations of 2D image sequence, is commonly referred to as Structure from Motion (SfM). It is traditionally solved with dense motion fields (optical flow) or triangulation via epipolar geometry. By identifying feature points between consecutive frames in a sequence of images and calculate the displacement between each pair of corresponding points, optical flow can be constructed as an approximation of the motion field from which the 3D motion can be recovered. While in triangulation via epipolar geometry, a 3D point in space is reconstructed by corresponding two image points taken from different views whose relationship is defined by an essential matrix.

All these strategies that rely heavily on the correctness of point correspondence have their shortcomings as their accuracies are often limited to the quality of feature extractors [12]. Given a agent moving in a unknown scene, some may anticipate it to be a better way to solve the 3D motion and the scene together instead of trying to guess them separately. In other words, they simultaneously refine the relative motion and scene geometry from a set of images of 3D points

from different views in a bundle adjustment fashion [13]. This approach is most prominent in its application in Simultaneous localization and mapping (SLAM), where both the mapping of an unknown environment and the tracking of the camera locations are recovered simultaneously [6]. But this method still depends on the abundance of features for appropriate point correspondence, the lack of which commonly challenges the quality of the algorithm in the wild. One famous incident being NASA's Ingenuity Mars helicopter crashed due to featureless terrain [7].

In recent years, neural networks are developed to leverages the power of deep learning to solve camera pose [20]. Rather than seeking to match the features explicitly, neural networks are typically capable of approximating optical flow by learning the hidden representation through their enormous size of parametrization. To offer more explainable results, works have also be done to incorporate additional constraints stemming from the physical properties of the motion field, like normal flow, into their network [23]. However, these results are often computationally costly, lack the theoretical guarantee, and are susceptible to over-fitting, making generalization difficult. Finally, due to computational cost, they can only consider visual information over small intervals such as 10's of milliseconds which greatly increases the difficult of the egomotion estimation problem. Here the presented research argues that the problem of egomotion can be solved with low dimensionality which only scales with the number of lines that are being tracked in the environment by incorporating the IMU sensor, and the intervals over which the visual data is considered can be 10's of seconds long with a sliding window solver.

This work is motivated by the fact that our world is filled with straight lines whether it be the edges of tables or outlines of buildings. Previous researches that worked on structure from motion with lines more or less viewed them as easy features for correspondence [1] [2] [3] [26] [28]. While the line geometry yields adequate assistance, it was noticed that the continuous motion of

lines in 3D shares special geometric properties, that they form a type of surfaces, named ruled surface, which describes the 3D scenes in terms of lines and can be used to derive the egomotion of the observing agent. The presented research offers a novel modeling based on ruled surfaces that enables the computation of camera egomotion and reconstruction of 3D scenes with low-dimensional solutions, meanwhile takes advantages of the geometric properties of ruled surface to avoid point-to-point correspondence and feature extraction in images. A reprojection loss with an inner optimization solved in closed form is developed for an optimization task which estimates the optimal parameters subject to those geometric constraints. At all stages, the possible ruled surfaces are constrained by the IMU, thus realizing an estimator with tight visual-inertial coupling despite the relaxation of point correspondence, and solved in a sliding window fashion.

## 1.2 Related Works

The classical way to solve structure from motion is to consider the problem as calculating the relative 3D motion from precise measurements on 2D images. With the spatio-temporal derivatives of two consecutive frames, either point correspondence [18] or optical flow [9], are first computed and the epipolar constraints are applied to gauge the 3D motion. Both of which are not trivial to solve and point correspondence is indispensable to these approaches, and the errors point correspondence as a result of failure in feature detectors can exacerbate the errors in 3D motion estimation. It is possible to improve the quality of point matching and optical flow themselves, whereas one might expect that it is better to introduce methods to provide additional guarantees on the 3D motion. Works have exploited the fact that the projection of optical flow in the direction of image gradient (normal flow) is resilient to outliers to enforce robust and

direct constraints on the estimated camera pose [8] [23]. Moreover, other works have proposed practices to discard the erroneous optical flow and recover motion from normal flow directly [4] [25]. Additional constraints are also available when considering the motion field. The epipolar constraint can be introduced by measuring the deviation in epipolar views but it is only available when optical flow is available; the positive depth constraint, which is derived from the fact that the scene has to be in front of the image to be visible, can be applied to normal flow [10] [12].

While the normal flow based techniques theoretically overcome the challenges of pure optical flow based ones, more can be done to better the egomotion estimation. When considering recovering motion from visual inputs, the geometric properties of the observed scene over long time intervals offer valuable information to solving the problem. One of such examples is the surfaces carved by straight lines in the world projected into image space. Line-correspondence has been proposed as an alternative to point-correspondence with a closed form solution for camera motion [29], albeit lines are not as simple as points due to their complexities with orientations. Nevertheless, with the use of Plücker's coordinates, spatial relationships of lines can be derived relatively easily [24] [27]. Algorithms that solve structure from motion with lines calculate the camera pose under the epipolar constraint but with the properties of lines, either through interpreting parallel lines as textures [1] [2] or line-to-line correspondence [3] [26]. Newer hardware like RGB-Depth camera also enabled camera extrinsic approximation from line features [28].

Additionally, the types of information that can be retrieved on hardware level are crucial to addressing any problems. In the case of motion, one must remember IMU sensors which record the angular velocity and acceleration up to a gravitational bias. For instance, PL-VIO combines the properties of lines and inertial odometry, namely inertial measurements enabled by IMU

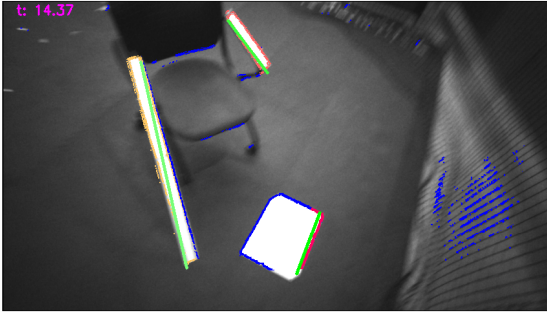
sensor, to solve the camera egomotion using point and line correspondences with a tight coupling to the IMU [17]. Their approaches unify the points features and line features while making use of the inertial data; a joint optimization is set up to minimize the IMU measurement residuals and re-projection residuals.

Recent advancements in visual sensors also helped expand the types of visual inputs available. Event cameras are a special type of vision sensors that capture the change of brightness in images asynchronously. This unique property has lead to more robust computation of optical flow by analyzing the time surface spanned by events [22], which is then integrated into solving egomotion estimation, either through a combination of event camera and traditional camera [30] or through deep learning [31]. Furthermore, due to the nature of the sensor it is easier to identify and cluster events belonging to the same edge. Utilizing this characteristic of event camera and line geometry, [5] proposed event-based line-SLAM that works in the fashion of parallel tracking and mapping. Combining that with IMU measurements, linear solver was used to estimate motion with event camera. For a small enough time interval and assuming constant linear velocity, events generated by a line circumscribe a manifold from which the motion can be recovered [15] [16]. [21] extracts line positions from temporal event clouds through minimizing event-to-line distance in the image projections.

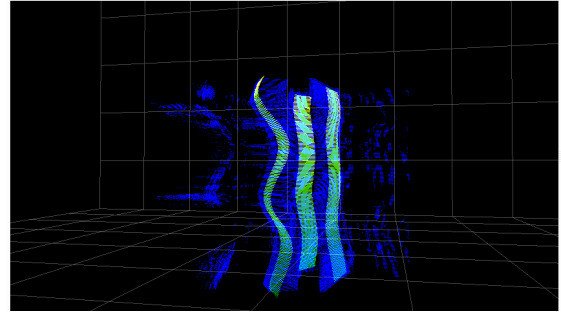
The presented approach goes beyond treating lines in each images as individual features for correspondence. Rather it observes that a straight line sweeps a special type of surface, ruled surface, as it moves. By inspecting the shapes of ruled surfaces, the camera egomotion can be directly reconstructed without any assumption on its velocity. Furthermore, with the incorporation of an onboard IMU sensor, the camera egomotion can be constrained with the linear and angular velocity measurements, which reduces the dimensionality of the solution space

to levels that only scales with the number of ruled surfaces being analyzed.

## Chapter 2: Problem Formulation



(a) Rulings projected to image frame. The blue points are thresholded image gradient pixels, and the green lines represent rulings projected to image frame. Pixels within a certain distance of the rulings are colored in red and yellow; they are used as data points for ruled surface estimation.



(b) Surfaces  $TS_{X_0, V_0}(t, \alpha)$  ruled by projections of lines in image-time space, approximated in a 1.5-second-long window. The  $z$ -axis (vertical up) is time  $t$ .

Figure 2.1: Rulings projected to image frame and ruled surfaces in image-time space.

### 2.1 Ruled Surface

In geometry, a surface  $S$  is ruled if for every point  $p$  on the surface there's a straight line  $l$  on  $S$  that also passes through  $p$  (figure 2.2). Formally, a ruled surface is a mapping  $F_{(\gamma, \delta)} :$

$I \times \mathbb{R} \rightarrow \mathbb{R}^3$  defined as

$$F_{(\gamma, \delta)}(t, \alpha) = \gamma(t) + \alpha\delta(t) \quad (2.1)$$

Where directrix  $\gamma(t) : I \rightarrow \mathbb{R}^3$ , ruling direction  $\delta(t) : I \rightarrow \mathbb{R}^3 \setminus \{\mathbf{0}\}$  are smooth mappings on open interval  $I$ . At each fixed time  $t$ , surface  $F_{(\gamma,\delta)}$  is ruled by a straight line that only depends on  $\alpha$ :  $\mathbf{u}(\alpha) = \gamma(t) + \alpha\delta(t)$  [19].

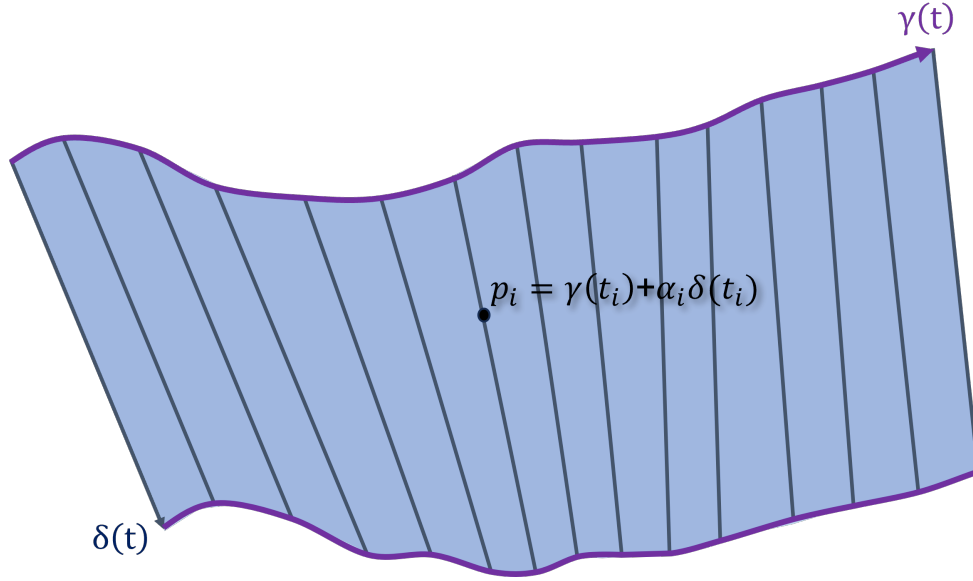


Figure 2.2: On a ruled surface, each point  $p_i$  also lies on a straight line.

For some fixed  $t_0$  at which the observation starts, let  $X_0 = \gamma(t_0)$  and  $V_0 = \delta(t_0)$ , a line in  $\mathbb{R}^3$  parameterized by only  $\alpha$  is

$$X(\alpha) = X_0 + \alpha V_0 \quad (2.2)$$

For a rotation-less camera (stabilized by an IMU sensor), if the camera translates relative to that 3D line according to signal  $-X(t)$ , for a rigid environment, the motion of the line relative to the camera is  $X(t)$ . This subsequently leads to a line that translates in 4-dimensional space-time space. The surface ruled by such line in camera frame is

$$S_{X_0, V_0}(t, \alpha) = X_0 + \alpha V_0 + X(t) \quad (2.3)$$

If that line is in the camera field of view, its projection in 2D image plane is

$$p(t, \alpha) = \frac{X_0 + \alpha V_0 + X(t)}{[X_0 + \alpha V_0 + X(t)]^z} \quad (2.4)$$

Since the projection of a 3D line is still a line in image plane (except when the line is perpendicular to the image plane), it naturally entails that the surface swept by the projected line in image-time space is also ruled. To differentiate from (2.3), below it is defined as ruled image-surface which is a surface in 3-dimensional image-time space

$$TS_{X_0, V_0}(t, \alpha) = \begin{bmatrix} p(t, a) \\ t \end{bmatrix} \quad (2.5)$$

It is evident that  $TS_{X_0, V_0}$  can be constructed by  $S_{X_0, V_0}$  (figure 2.3), but the converse remains to be solved.

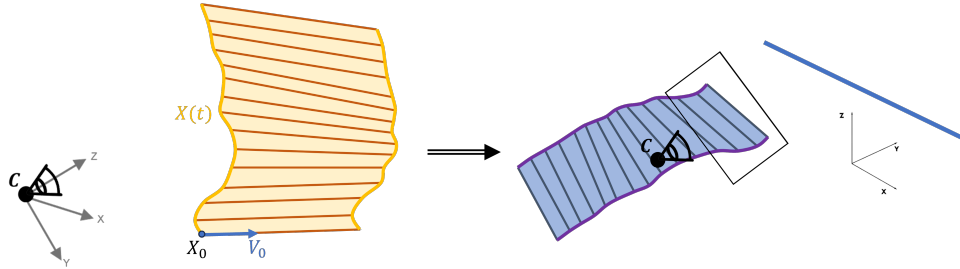


Figure 2.3: The surface  $S_{X_0, V_0}$  on the left is swept by the motion of a line in camera frame; the surface  $TS_{X_0, V_0}$  on the right is swept by the projection of a line in the image plane across time.  $S_{X_0, V_0}$  leads to  $TS_{X_0, V_0}$  by construction.

## 2.2 Parameterization

Given a set of 2D data points from a sequence of images  $\{p_i\}_{i=1}^N$  (whose selection procedure is explained in Chapter 3) that presumably belong to a surface  $TS_{X_0, V_0}(t, \alpha)$  which is ruled by

the projection of a line  $l$  in  $\mathbb{R}^3$ , the objective is to approximate  $TS$ , from which a surface  $S$  ruled by the motion of  $l$  in camera frame can be deduced. That will ultimately yield the camera translation signal  $-X(t)$ , which is the camera egomotion relative to the ruling  $l$  minus rotation which is taken out by the IMU.

Thus, the focus is on setting up an optimization problem with a parameterization of  $TS$ . Start by rearranging equation 2.4

$$[X_0 + \alpha V_0 + X(t)]^z p(t, \alpha) = X_0 + \alpha V_0 + X(t) \quad (2.6)$$

If the linear acceleration measurements from the IMU sensor is available,  $X(t)$  can be decomposed into

$$X(t) = X(t_0) + \dot{X}(t_0)(t - t_0) + \int_{t_0}^t \int_{t_0}^{\sigma} \ddot{X}(\sigma_2) d\sigma_2 d\sigma_1 \quad (2.7)$$

An additional term of  $\frac{1}{2}t^2g$  may be needed if only an IMU with gravitational bias is available. In practice, it is observed that an additional linear bias  $\xi$  is introduced by the integration error, which can be lumped together with  $\dot{X}$  during optimization.

Define operator  $\Gamma\{\ddot{X}\} = \int_{t_0}^t \int_{t_0}^{\sigma} \ddot{X}(\sigma_2) d\sigma_2 d\sigma_1$  and assume the sequence starts at  $t_0 = 0$

$$X(t) = X(0) + (\dot{X}(0) + \xi)t + \Gamma\{\ddot{X}\} + \frac{1}{2}t^2g \quad (2.8)$$

Combining equation 2.5 and equation 2.8, note the displacement  $X(0) = 0$

$$\begin{aligned}
& \left[ X_0^z + \alpha V_0^z + t(\dot{X}^z(0) + \xi^z) + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2g^z \right] p(t, \alpha) \\
& = \left[ X_0^{(x,y)} + \alpha V_0^{(x,y)} + t(\dot{X}^{(x,y)}(0) + \xi^{(x,y)}) + \Gamma\{\ddot{X}^{(x,y)}\} + \frac{1}{2}t^2g^{(x,y)} \right]
\end{aligned} \tag{2.9}$$

Lumping the terms by their common multipliers and substituting the symbols give the parameterization by  $a, b, c, d, e, f, g$ , where  $d, e, f$  are two dimensional while  $a, b, c$  are scalars.

$$p(t, \alpha) = \frac{\underbrace{\alpha V_0^{(x,y)}}_d + \underbrace{X_0^{(x,y)}}_e + t \underbrace{(\dot{X}^{(x,y)}(0) + \xi^{(x,y)})}_{f} + \Gamma\{\ddot{X}^{(x,y)}\} + \frac{1}{2}t^2g^{(x,y)}}{\underbrace{X_0^z}_a + t \underbrace{(\dot{X}^z(0) + \xi^z)}_b + \alpha \underbrace{V_0^z}_c + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2g^z} \tag{2.10}$$

Where the directrix at  $t_0$  is  $X_0$ , the ruling direction vector is  $V_0$ , and the camera translation signal is  $-X(t)$ . They are the components of (2.3) which can now be constructed from  $TS_{X_0, V_0}(t, \alpha)$ .

$$X_0 = \begin{bmatrix} e^x \\ e^y \\ a \end{bmatrix}, V_0 = \begin{bmatrix} d^x \\ d^y \\ c \end{bmatrix}, X(t) = \begin{bmatrix} tf^x + \Gamma\{\ddot{X}^x\} + \frac{1}{2}t^2g^x \\ tf^y + \Gamma\{\ddot{X}^y\} + \frac{1}{2}t^2g^y \\ tb + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2g^z \end{bmatrix} \tag{2.11}$$

This parameterization gives six unknowns for the ruling, three unknowns for the camera initial velocity plus bias, and three unknowns for gravitational bias. An unbounded line in 3D, however, can be reduced to only 4 degrees of freedom, as it can translate and rotate along the axis of itself.

Therefore, additional constraints to bound the dimensionality of the solution space are needed. Forcing  $V_0$  to be a unit vector and  $X_0$  to be the point such that the vector from origin to  $X_0$  is perpendicular to the line essentially removes two degrees of freedom and reflects the line's four degrees of freedom in the 3D space. In addition, the denominator in the right-hand side of equation 2.10 should be strictly greater than 0 as the observed scene is in front of the image plane.

### 2.3 Ruling Reprojection Loss

Let  $\{p_i\}_{i=1}^N$  be the set of points that are projected by some ruling defined by a point  $X_0$  and direction  $V_0$ , it is desired to find optimal parameters that minimizes this error of ruling reprojection, which is the difference between left-hand side and right-hand side of equation 2.4, or rather its parameterized equivalent equation 2.10, bounded by other constraints.

Formally, define the ruling reprojection loss at time  $t$  to be the sum of the squared difference between projected points and estimated ruling with parameters  $\theta = (a, b, c, d, e, f, g)$

$$\mathcal{L}_t(\theta) = \sum_{i=1}^N \left\| p_i - \frac{\alpha_i d + e + t f + \Gamma\{\ddot{X}^{(x,y)}\} + \frac{1}{2}t^2 g^{(x,y)}}{a + t b + \alpha_i c + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2 g^z} \right\|^2 \quad (2.12)$$

Another issue remained to be tackled is the implicit  $\alpha_i$  along the ruling for each point  $p_i$ . They are dependent on each other. In the real world, some unknown  $\alpha_i$  evaluates a point  $p_i$  through projection while in the optimization problem  $p_i$  is known and  $\alpha_i$  needs to be estimated. This is equivalent to solving

$$\min_{\alpha_i} \left\| p_i \left( a + t b + \alpha_i c + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2 g^z \right) - \left( \alpha_i d + e + t f + \Gamma\{\ddot{X}^{(x,y)}\} + \frac{1}{2}t^2 g^{(x,y)} \right) \right\|^2 \quad (2.13)$$

By setting the inner part to 0 and switching variables between its left-hand-side and right-hand-side, it can be solved with a system of 2 linear equations.

$$\begin{aligned} (p_i^x c - d^x) \alpha_i &= \Phi_i^x \\ (p_i^y c - d^y) \alpha_i &= \Phi_i^y \end{aligned} \quad (2.14)$$

Where

$$\Phi_i^w = \left[ e^w + t f^w + \Gamma\{\ddot{X}^w\} + \frac{1}{2}t^2 g^w - p_i^w \left( a + t b + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2 g^z \right) \right] \quad (2.15)$$

Now, the whole constrained minimization problem became

$$\min_{\theta} \mathcal{L}_t(\theta) \quad (2.16)$$

$$s.t. \quad \|V_0\| = 1$$

$$X_0 \cdot V_0 = 0$$

$$a + tb + \alpha_i c + \Gamma\{\ddot{X}^z\} + \frac{1}{2}t^2 g^z > 0, \forall i$$

$$\text{where } \alpha_i = (P_i^\top P_i)^{-1} P_i^\top \Phi_i$$

$$P_i = \begin{bmatrix} p_i^x c - d_x \\ p_i^y c - d_y \end{bmatrix}, \Phi_i = \begin{bmatrix} \Phi_i^x \\ \Phi_i^y \end{bmatrix}$$

## 2.4 Dimensionality

While the problem is solvable for one surface, the estimated pose would be unbounded along the direction of its ruling. Thus to restore the unique camera pose, a minimum number of 2 non-parallel lines are necessary.

For  $M$  lines ( $M > 1$ ), assuming static environment, the parameters  $b$ ,  $f$ , and gravitational bias  $g$  can be shared across different lines. Each surface would have 6 independent parameters and 6 shared parameters that give  $6M + 6$  unknowns. As discussed, an unbounded line in 3D have only 4 degrees of freedom. Thus, under additional constraints, the dimensionality of the solution space is only  $4M + 6$ .

Next, a set of surfaces ruled by the motion of lines in camera frame on a closed time interval can be defined as the following. Let  $X_{t_0}^l, V_{t_0}^l$  be the ruling position and direction for the  $l$ -th ruling starting from time  $t_0$ , and  ${}^{-t_0}X(t)$  be the camera translation signal originating at  $t_0$  independent of individual surface. A set of ruled surfaces on interval  $[t_0, t_n]$  is

$$RS(t_0, t_n) = \{R(l, t_0, t) = X_{t_0}^l + \alpha V_{t_0}^l + {}^{t_0}X(t) | t \in [t_0, t_n]\}_{l=1}^N \quad (2.17)$$

Where  $R(l, t_0, t)$  denotes the ruling at time  $t$  of the  $l$ -th surface originating from time  $t_0$ , in which each line is optimized by section 2.3. For example, figure 2.1a illustrates the projected rulings in an image frame at some time. The surfaces ruled by these projections are shown in figure 2.1b.

## Chapter 3: Methodology

### 3.1 Initial Surface Estimation

Only straight edges are crucial to the ruled surface formulation. Inspired by event camera's exceptional capabilities at detecting edges in vision tasks [14], the presented research establishes a procedure with image gradient to simulate "events".

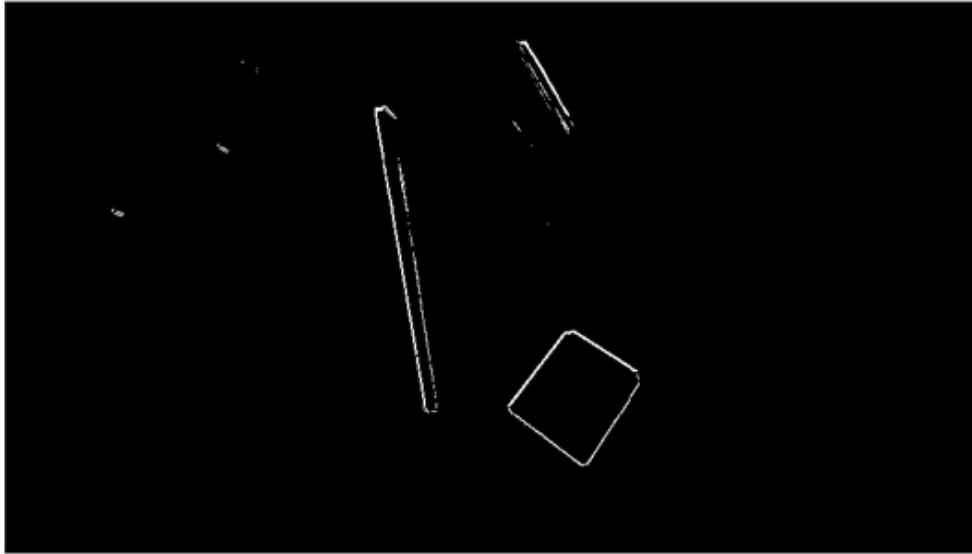


Figure 3.1: Edge map made by differentiating frames in time and thresholding them.

By differentiating the consecutive image frames by time and threshold them, an edge map like figure 3.1 is generated. Let  $[t_0, t_0 + \delta t]$  represent a small enough time interval during which the change in image gradient along each line remains below a certain threshold  $\epsilon$ .  $N$  initial ruled surfaces can be found based on lines identified by Hough transform. From there, the directions of

the rulings in the projected image can also be calculated and used as part of the initial parameters.

### 3.2 Extruding Surface into Future Frames

For surfaces estimated on a time interval  $[t_0, t_i]$ , the algorithm has to extrude the same surface to a longer interval  $[t_0, t_{i+1}]$ .

---

**Algorithm 1** Extrude Surface

---

**Given:**

Estimated  $RS(t_0, t_i)$

Set  $E$  of points projected by  $RS(t_0, t_i)$  on  $[t_0, t_i]$

**Do:**

At  $t_{i+1}$

**for** each line  $l$  **do**

    Estimate ruling  $R(l, t_0, t_{i+1})$

**for** each  $p_i$  measured at  $t_{i+1}$  **do**

**if**  $dist(p_i, R(l, t_0, t_{i+1})) < \tau$  **then**

            Add  $p_i$  to  $E$

**end if**

**end for**

**end for**

**while** Loss of  $RS(t_0, t_{i+1})$  on  $E > 0.01$  **do**

    Add small perturbation to  $RS(t_0, t_{i+1})$  and run the optimizer again

**end while**

**Return:**

Estimated  $RS(t_0, t_{i+1})$  from appended  $E$

---

As described in algorithm 1, the surface at  $t_{i+1}$  can be predicted with current estimation and acceleration measurements from the IMU. This prediction approximates a ruling  $R(l, t_0, t_{i+1})$  based on the existing surface estimation  $RS(t_0, t_i)$ . Image points close to the estimated ruling within a certain distance,  $dist(p_i, R(l, t_0, t_{i+1})) < \tau$ , are sampled and appended to all points within that distance from the surface on  $[t_0, t_i]$ . This distance is the total least squares distance from sample point  $p_i$  to the projection of the ruling  $R(l, t_0, t_{i+1})$ .

$$\text{dist}(p_i, R(l, t_0, t_{i+1})) = \frac{(r_2^y - r_1^y)p_i^x - (r_2^x - r_1^x)p_i^y + r_2^x r_1^y - r_2^y r_1^x}{\sqrt{(r_2^y - r_1^y)^2 + (r_2^x - r_1^x)^2}} \quad (3.1)$$

where

$$r_1^x = \frac{[X_{t_0}^l + t_0 X(t)]^x}{[X_{t_0}^l + t_0 X(t)]^z}, r_1^y = \frac{[X_{t_0}^l + t_0 X(t)]^y}{[X_{t_0}^l + t_0 X(t)]^z}$$

$$r_2^x = \frac{[X_{t_0}^l + V_{t_0}^l + t_0 X(t)]^x}{[X_{t_0}^l + V_{t_0}^l + t_0 X(t)]^z}, r_2^y = \frac{[X_{t_0}^l + V_{t_0}^l + t_0 X(t)]^y}{[X_{t_0}^l + V_{t_0}^l + t_0 X(t)]^z}$$

Here, a two-point formulation of distance calculation is used. Since  $\alpha$  is free, the points  $r_1, r_2$  can be easily calculated by setting  $\alpha = 0, 1$ .

The aggregated data is then used to estimate new surfaces over the extended time interval, resulting in  $RS(t_0, t_{i+1})$ . To improve this process, if the average cost in the new estimation is greater than some small value (0.01), some random perturbation is added to the parameters and the optimization process is run again until optimal solution with low cost is achieved. Repeating this process allows surface estimation to be extended into future frames.

### 3.3 Sliding Window

Due to the errors inherent to acceleration and gyro sampling on the hardware level, the estimation  $RS(t_0, t_n)$  would be less accurate as the length of the interval increases. This problem, however, can be mitigated by adopting a sliding window approach and gradually move the start time  $t_0$  of the estimated surfaces.

For a recording sampled for  $\mathcal{T}$  seconds long at some frequency  $\mathcal{F}$ , there are  $\mathcal{N} = \mathcal{T} \cdot \mathcal{F}$  images captured at unique timestamps. Let the start time of the  $\mathcal{N}$ -long sequence be  $t_0$  and the end be  $t_{\mathcal{N}}$ , for any time interval  $[t_j, t_{j+k}]$  where  $j \geq 0$  and  $(j+k) \leq \mathcal{N}$ , define it to be a window

$wind(k, t_j)$  of size  $k$  starting at  $t_j$  for some  $k$  such that  $(t_{j+k} - t_j) < \iota$  where  $\iota$  is of reasonable length such that the hardware sampling error is negligible.

---

**Algorithm 2** Sliding Window

---

**Given:**

Estimated  $RS(t_j, t_{j+k})$  on  $wind(k, t_j)$

Set  $E$  of points projected by  $RS(t_j, t_{j+k})$  on  $wind(k, t_j)$

**Do:**

At  $t_{j+k+1}$

**for** each line  $l$  **do**

    Estimate ruling  $R(l, t_j, t_{j+k+1})$

**for** each  $p_i$  measured at  $t_{j+k+1}$  **do**

**if**  $dist(p_i, R(l, t_j, t_{j+k+1})) < \tau$  **then**

            Add  $p_i$  to  $E$

**end if**

**end for**

**end for**

Trim  $E$  to interval  $[t_{j+1}, t_{j+k+1}]$

**while** Loss of  $RS(t_{j+1}, t_{j+k+1})$  on  $E > 0.01$  **do**

    Add small perturbation to  $RS(t_{j+1}, t_{j+k+1})$  and run the optimizer again

**end while**

**Return:**

Estimated  $RS(t_{j+1}, t_{j+k+1})$  from appended  $E$

---

As illustrated in algorithm 2, starting with  $j = 0$  in window  $wind(k, t_j)$  and estimated surfaces on that window  $RS(t_j, t_{j+k})$ , sliding the window and increasing  $j$  repetitively, it can estimate the surfaces on all windows  $wind(k, t_j)$  for  $j \geq 1$ .

### 3.4 Recover Camera Egomotion and Scene Geometry from Ruled Surface

After collecting the surface estimations on each window from a sequence, the whole camera egomotion, as well as surrounding scenes in terms of lines, can be reconstructed. While the latter is derived directly from the ruled surface as discussed in section 2.2, the former is less explicit.

Let the camera translation signal  $-{}^{t_i}X(t_j)$  denotes the displacement of camera at time  $t_j$

relative to lines on its trajectory starting from  $t_i$ , the full egomotion  ${}^{t_0}\hat{X}^{t_n}(t)$  from  $t_0$  to  $t_n$  is the summation of all consecutive displacements along the trajectory

$${}^{t_0}\hat{X}^{t_n}(t) = - \sum_{i=0}^n {}^{t_i}X(t_{i+1}) \quad (3.2)$$

## Chapter 4: Experiments

### 4.1 Setup

Using a Realsense 435i camera with 90 fps and internal IMU, the algorithm is tested on recovering the correct camera egomotion on a series of scenes with different settings. The ground truth values of the egomotion trajectories are recovered in two ways: for the experiments with UR10 robot arm, the physical dimensions of the camera is collected from its design file such that the robot is set up in a way the trajectory of the tip of the arm is the same as the camera center up to millimeter accuracy; for the experiments with Vicon tracking, a joint calibration is done so that the rotation and translation from the Vicon world frame to the camera pose is always known, from which the ground truth camera egomotion as well as the ground truth line poses in camera frame are both known through Vicon. Each sequence is roughly 12 seconds long.

### 4.2 Surface Estimation

To derive ruled surfaces, first a 0.1 seconds long interval is chosen for initial line edges detection. The parameters for hough transform have  $\rho = 5$  pixels and  $\theta = 1$  degree with threshold for the accumulator being 100, minimum line length being 100 pixels and maximum line gap being 5 pixels. Pick the line with the highest votes from the hough transform accumulator, remove

all the data points that are within some distance (5 centimeters in undistorted image plane) to such line and run hough transform on the rest of the data points again. Repeat this process  $M$  times result in  $M$  initial surfaces ( $N = 3$  in the example 2.1b).

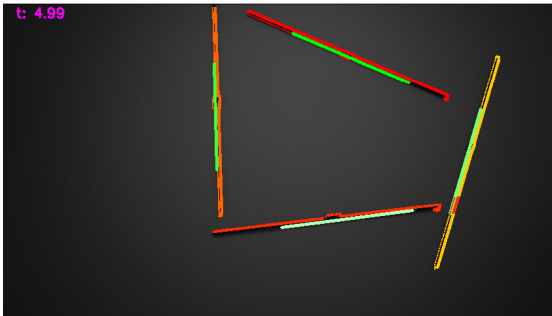
After the initial estimation, the surfaces can be extruded into the future with the sliding window technique to eventually recover the camera egomotion. Based on observations, setting the window size to 140 frames or roughly 1.55 seconds works well. The parameters are calculated through solving a nonlinear least-squares problem that minimizes the ruling reprojection loss of equation 2.12.

### 4.3 Derotation

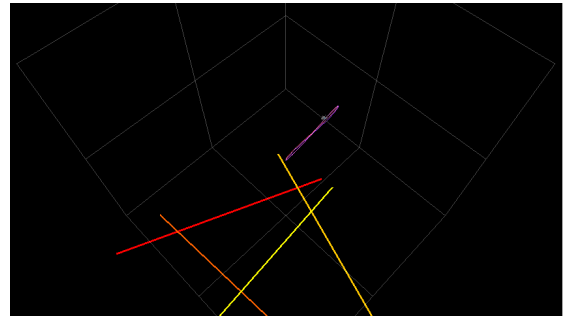
To achieve the requirement of rotation-less camera, it is critical to derotate the camera frames with gyro measurements from the IMU sensor, which can be integrated into quaternion units. Let rotation matrix  ${}^{t_2}R^{t_1}$  denotes the rotation of the camera pose from time  $t_1$  to  $t_2$ . For sliding window, all the frames within a single window need to be de-rotated to the same frame at  $w_0$ . Denote this  $w_0$  as the rotation center of the window.

For a window of size  $k$ , for each frame at  $w_i, 0 < i \leq k$ , rotate it by  ${}^{w_0}R^{w_i}$  to negate the rotation of camera within the window. Since this process makes the camera and line poses in each window unique to their rotation, another procedure to rotate each window to some common rotation center  $t_0$  is naturally needed. For a window  $wind(k, w_j)$  starting at some time  $w_j$ , apply rotation  ${}^{t_0}R^{w_j}$  to frames and estimated poses within a window before combining windows. To restore the original frames, a final rotation  ${}^{t_i}R^{t_0}$  can be applied to each frame where  $t_i$  is their respective time stamp.

## Chapter 5: Results



(a) Scene used for experiments in Section 5.2.1. Four coplanar lines are present in the scene. Estimated rulings are in green, and the sampled image points belonging to the ruled surfaces of each line are shown in warm colors.



(b) Estimated trajectory of camera egomotion and estimated poses of lines in the scene in world coordinates. Lines are in warm colors, and the trajectory in purple, indicating linear motion parallel to the scene plane.

Figure 5.1: Scene setup with coplanar lines and estimated rulings, and the camera egomotion trajectory alongside line poses in world coordinates.

### 5.1 Overview

Starting from simple motions, first consider coplanar lines (lines formed in a square shape as in figure 5.1a) and single directional rotation-less motions, parallel or perpendicular to the plane of the lines. On top of linear motions, circular ones with sinusoidal signals in two or more directions are also tested, illustrated in figure 5.2.

Naturally, any real world camera motion would involve rotations. The motions with rotations are grouped into two sets, linear and circular, similar to the previous setup. To test the robustness

of the algorithm, rotations in all three directions, X, Y, Z, relative to camera itself, are added to each motion. If the translation is in one direction, this setup is intended to test if the rotation in any direction would disproportionately affect the evaluation quality.

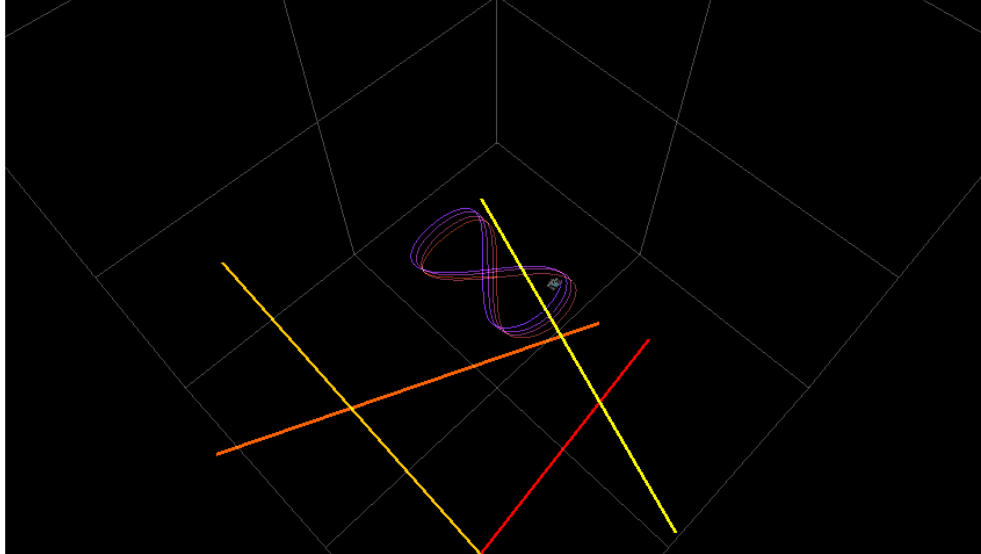


Figure 5.2: Estimated trajectory of camera egomotion and the estimated poses of the lines in the scene in world coordinates. Lines are in warm colors and the trajectory in purple. Circular motion parallel to the scene plane.

Other than only smooth, sinusoidal motions, it is sensible to study the cases where the motion is not smooth as well. In the same form of back-and-forth motions like figure 5.1b, instead of sinusoidal change of positions, a zigzag signal is used. And on top of those movements, motion in the shape of a square, like figure 5.3, with abrupt stopping at each corner is also used.

Advancing from simple coplanar lines, tests are performed on more complicated non-coplanar scenes to test if the algorithm generalizes. Figure 5.4 sums up the setup for this experiment. The four lines being tracked are carefully placed to be non-coplanar which generates a more convoluted scene when estimating 2.5. Since the lines are not coplanar, all motions are tilted with respect to the axis of each line.

Finally, to test beyond idealized motions, tests were done with hand held camera where

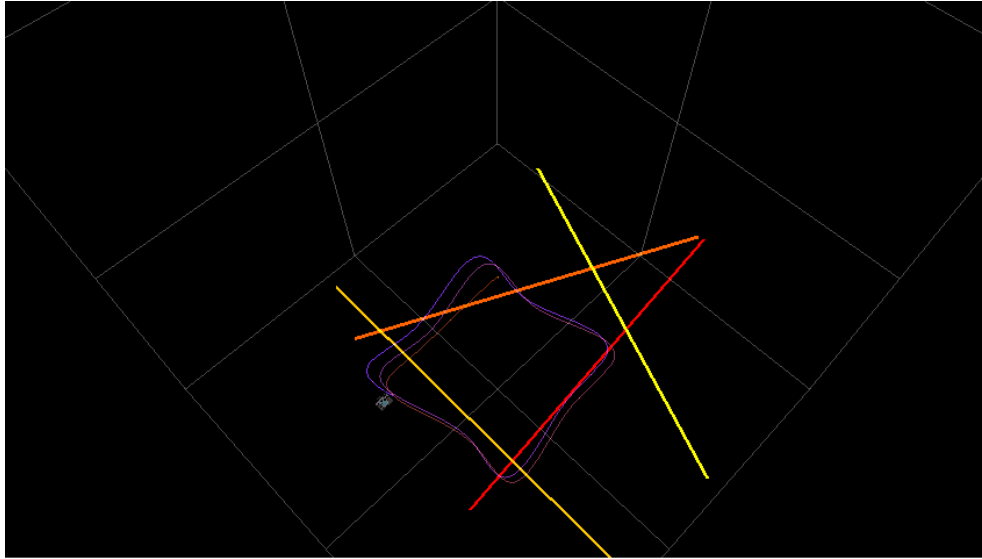
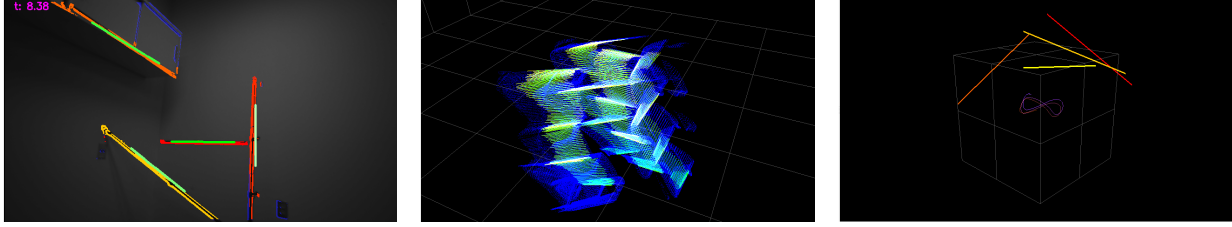


Figure 5.3: Estimated trajectory of camera egomotion and the estimated poses of the lines in the scene in world coordinates. Lines are in warm colors and the trajectory in purple. Square non-smooth motion parallel to the scene plane.

ground truth egomotion is recorded with Vicon motion tracking system. The scene is setup as illustrated in figure 2.1a and both coplanar and non-coplanar scenes are tested. The camera is held by a tester who moves it freely in front of a scene. In that scene, markers for tracking the poses of objects are attached to the lines so that the Vicon system can record the ground truth poses for lines in world coordinates as well. Running a joint calibration between the Vicon system and the camera, a transformation from Vicon world frame to camera frame can be obtained, from which the distance between markers and estimated lines in camera coordinates is calculated in ensuing experiments.



(a) Scene used for experiments in Section 5.2.4. Four non-coplanar lines are tracked in this scene.

(b) The non-coplanar scene generates a much more complicated point cloud in image-time space, making fitting surfaces difficult.

(c) Estimated trajectory of camera egomotion and the estimated poses of the lines in scene in world coordinates. The latter are clearly not coplanar.

Figure 5.4: (a) Scene setup with non-coplanar lines, (b) point cloud and estimated ruled surface in image-time space, and (c) the camera egomotion trajectory alongside line poses.

## 5.2 Discussion

### 5.2.1 Rotation-less Motions

Figure 5.1b, for example, is a trajectory of an slice of egomotion of roughly 12 seconds in world frame, its equivalent trajectory in world coordinates relative to the camera initial pose on X,Y,Z directions is plotted in 5.5a and figure 5.6a shows the perpendicular case. While the trajectory is accurately recovered, a drift caused by cumulating IMU measurement errors is visible particularly in the latter graph. This is also evident in Table 5.1 where the perpendicular motion has higher mean and standard deviation on differences from the ground truth.

<b>Motion</b>	<b>X (m)</b>	<b>Y (m)</b>	<b>Z (m)</b>
Linear Parallel	$0.0019 \pm 0.0080$	$0.0138 \pm 0.0215$	$0.0251 \pm 0.0173$
Linear Perpendicular	$0.0251 \pm 0.0159$	$0.0341 \pm 0.0270$	$0.0554 \pm 0.1004$
Circular Parallel	$0.0097 \pm 0.0197$	$0.0320 \pm 0.0230$	$0.0159 \pm 0.0211$
Circular Perpendicular	$0.0228 \pm 0.0258$	$0.0317 \pm 0.0384$	$0.1201 \pm 0.0800$
Circular Tilted-Axis	$0.1038 \pm 0.0808$	$0.0163 \pm 0.0236$	$0.1049 \pm 0.0665$

Table 5.1: Difference between True and Estimated Trajectories, Rotation-less Motion

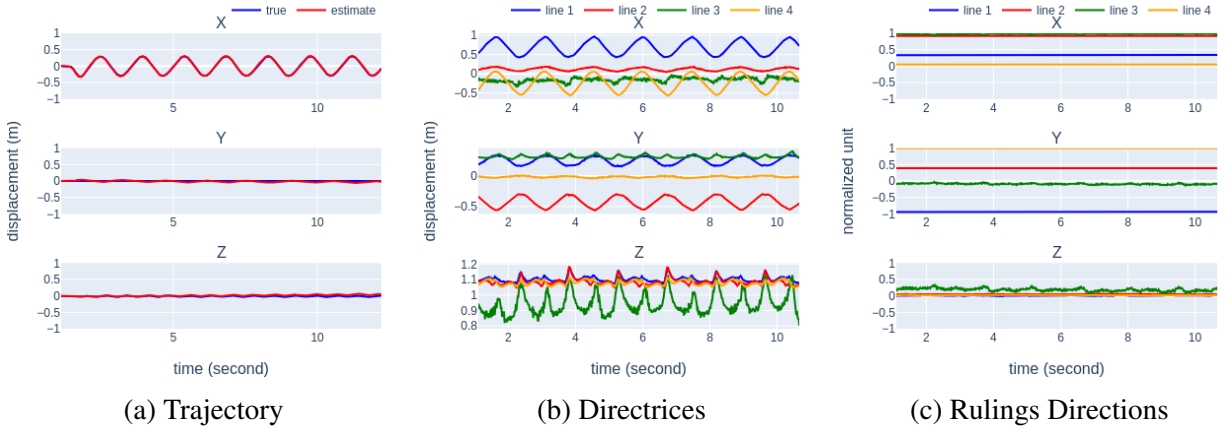


Figure 5.5: (a) Estimated trajectory of camera egomotion, (b) trajectory of the directrix for each ruled surface, and (c) ruling direction for each ruled surface, (a) in world coordinates with respect to the initial camera pose, (b)(c) in derotated camera coordinates; linear motion parallel to the scene plane.

The directrix (figure 5.5b 5.6b) and ruling (figure 5.5c 5.6c) in derotated camera coordinates can also be recovered. Since the lines are organized in a roughly square formation, it is reflected in parallel motion (figure 5.5b) where two of the lines move similarly to the camera motion while the other two don't as the two sides of the square that are parallel to the motion are not moving in the sense of line orientation. On the other hand, for the perpendicular motion (5.6b), the sinusoidal motion in all lines can be observed with the depth of the lines changing in uniform. Note that the trajectory of the directrix which portrays partly the motion of the line in camera coordinates is not just influenced by the camera motion, it is also forced such that the vector from camera origin to the directrix is perpendicular to the ruling for dimensionality reduction purposes as discussed in section 2.4. The ground truth of the line poses, which are compared in a later section, are not available in this set of experiments.

Since the motion is supposedly rotation-less, the ruling should remain relative constant across time; as revealed in figure 5.5c 5.6c the estimated results have different levels of noises, particularly troublesome in the Z direction. Albeit the camera trajectory is well constructed,

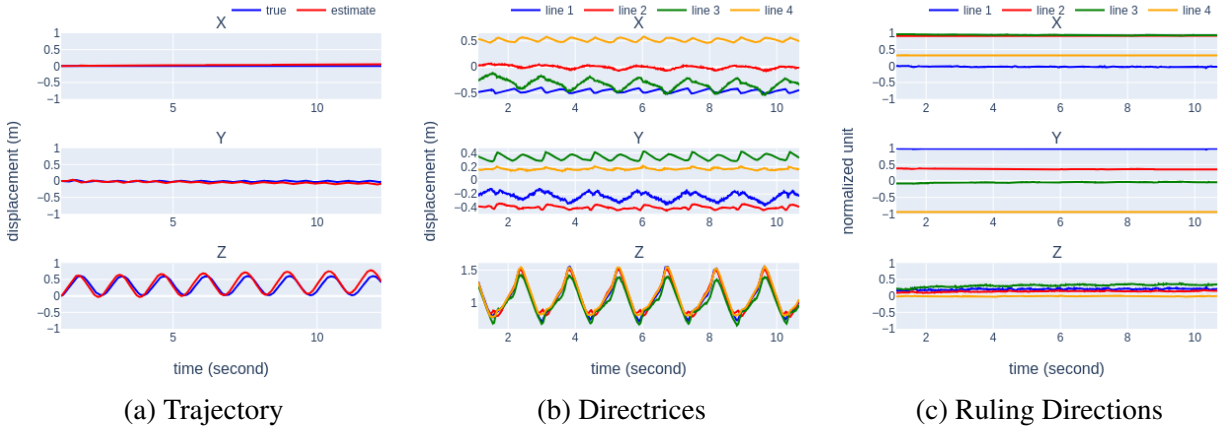


Figure 5.6: (a) Estimated trajectory of camera egomotion, (b) trajectory of the directrix for each ruled surface, and (c) ruling direction for each ruled surface, (a) in world coordinates with respect to the initial camera pose, (b)(c) in derotated camera coordinates; linear motion perpendicular to the scene plane.

the line scene geometry is less stable. The X,Y directions of the rulings are initialized via the projected direction of the lines in the image while the Z direction is unknown. This is the reason why the algorithm produces more noise on estimating ruling’s Z direction. When comparing the estimated trajectories to the ground truth, as shown in figure 5.7, it is clear that the perpendicular motions are worse than the parallel one. This is most likely caused by the drift of acceleration measurements on the IMU sensor, possibly in one direction more than the others.

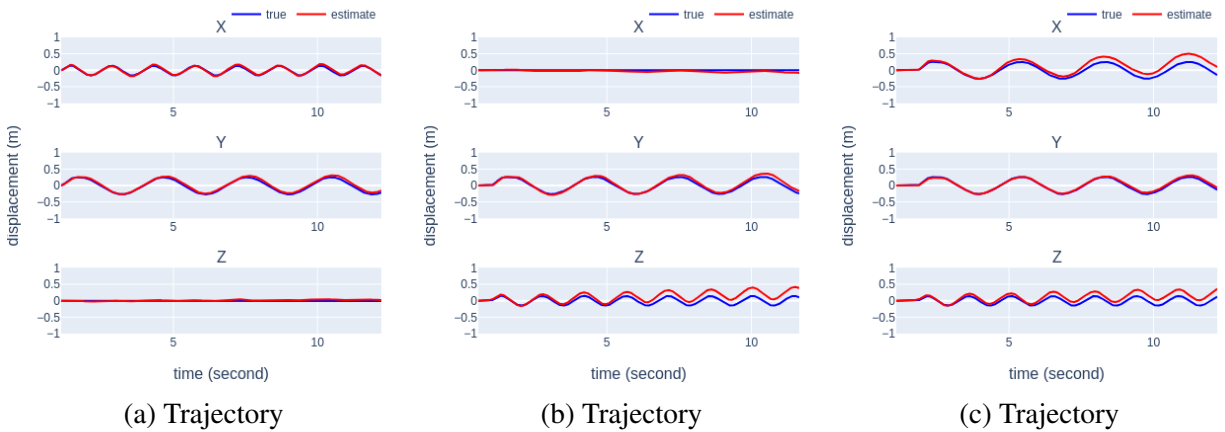


Figure 5.7: Estimated camera egomotion trajectories under different circular motion configurations: (a) parallel to the scene plane, (b) perpendicular to the scene plane, and (c) tilted to the scene plane.

## 5.2.2 Motions with Rotations

Again, motion in linear and circular forms are tested with motions parallel, perpendicular, and tilted with respect to the lines plane. The quantitative results are listed in table 5.2.

Motion	Rotation Direction	X (m)	Y (m)	Z (m)
Linear Parallel	X	$0.0278 \pm 0.0289$	$0.0171 \pm 0.0369$	$0.0041 \pm 0.0086$
	Y	$0.0510 \pm 0.0446$	$0.0223 \pm 0.0170$	$0.0874 \pm 0.0607$
	Z	$0.0035 \pm 0.0115$	$0.0452 \pm 0.0240$	$0.0222 \pm 0.0104$
Linear Perpendicular	X	$0.0130 \pm 0.0260$	$0.0310 \pm 0.0144$	$0.0461 \pm 0.0421$
	Y	$0.0211 \pm 0.0305$	$0.0395 \pm 0.0230$	$0.0427 \pm 0.0268$
	Z	$0.0064 \pm 0.0117$	$0.0164 \pm 0.0130$	$0.0313 \pm 0.0306$
Circular Parallel	X	$0.0163 \pm 0.0156$	$0.0438 \pm 0.0252$	$0.0229 \pm 0.0189$
	Y	$0.0052 \pm 0.0204$	$0.0285 \pm 0.0247$	$0.0381 \pm 0.0255$
	Z	$0.0005 \pm 0.0254$	$0.0195 \pm 0.0182$	$0.0270 \pm 0.0164$
Circular Perpendicular	X	$0.0012 \pm 0.0174$	$0.0222 \pm 0.0249$	$0.0002 \pm 0.0141$
	Y	$0.0207 \pm 0.0250$	$0.0510 \pm 0.0286$	$0.0098 \pm 0.0211$
	Z	$0.0528 \pm 0.0286$	$0.0283 \pm 0.0187$	$0.0616 \pm 0.0431$
Circular Tilted	X	$0.0068 \pm 0.0202$	$0.0933 \pm 0.0552$	$0.1125 \pm 0.0846$
	Y	$0.0836 \pm 0.0672$	$0.0039 \pm 0.0102$	$0.0272 \pm 0.0182$
	Z	$0.0856 \pm 0.0510$	$0.0103 \pm 0.0136$	$0.0957 \pm 0.0439$

Table 5.2: Difference between true and estimated trajectories, motion with rotation.

Not only the rotation in any direction does not seem to impose any negative effects on the estimated egomotion, it seems to perform better than the rotation-less cases. For example, in figure 5.6a it was tested that there's a drift in the world Z direction that has been negatively impacting the estimation. If the rotation is added, however, the algorithm was able to predict more accurate results as shown in figure 5.8. It is theorized that the drift is along a certain camera axis while the egomotion is estimated in world frame. Without rotations, the camera axes align with the world one up to an axis swap, making the drift accumulating along one world axis. But with rotations, this drift is dispersed and averaged out. Thus, the errors accumulated in the Z direction of the actual trajectory is reduced. Similar improvements can also be seen in other

types of motion.

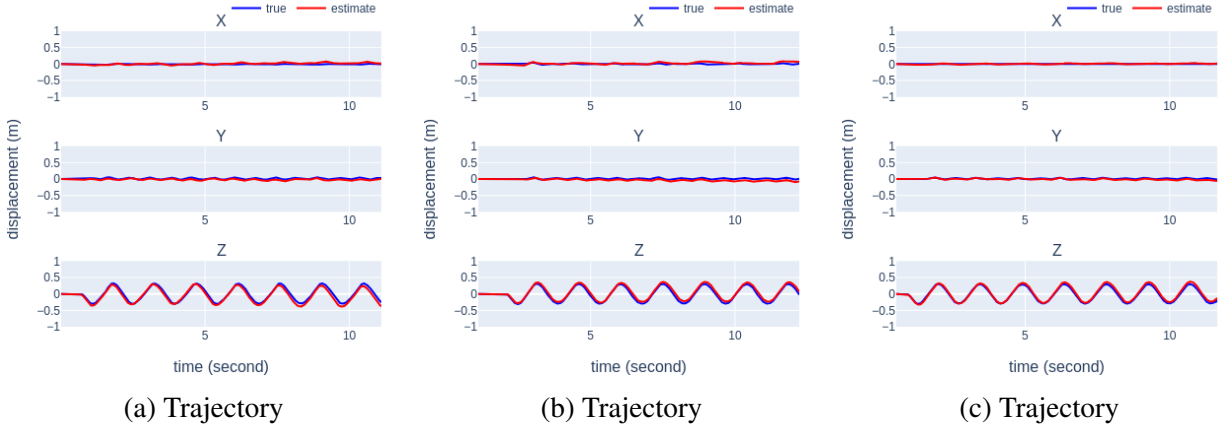


Figure 5.8: Estimated camera egomotion trajectories with linear motion perpendicular to the scene plane, combined with rotation in (a) the X direction, (b) the Y direction, and (c) the Z direction of the camera.

### 5.2.3 Non-smooth Motions

Non-smooth Motion	X (m)	Y (m)	Z (m)
Linear Parallel	$0.0081 \pm 0.0071$	$0.0504 \pm 0.0267$	$0.0317 \pm 0.0319$
Linear Perpendicular	$0.0241 \pm 0.0206$	$0.0204 \pm 0.0202$	$0.0957 \pm 0.0558$
Linear Tilted	$0.0255 \pm 0.0221$	$0.0259 \pm 0.0266$	$0.0631 \pm 0.0362$
Square Parallel	$0.0850 \pm 0.0478$	$0.0138 \pm 0.0182$	$0.0626 \pm 0.0427$
Square Perpendicular	$0.0096 \pm 0.0217$	$0.0126 \pm 0.0077$	$0.0449 \pm 0.0549$

Table 5.3: Difference between true and estimated trajectories, non-smooth motion.

The estimated and true trajectories are shown in figure 5.9. Notice that the same drift still persist on world Z axis. Moreover, taking the tilted motion, 5.10 , for example, the estimated directrices have more drastic movements as in 5.10b and the rulings have more noises as in 5.10c. This, however, does not equates to bad camera egomotion estimation in 5.10a, and does not necessarily equates to bad scene estimation either as the directrix can move along the axis of

the line itself. Analyzing the errors quantitatively, in table 5.3, there is not a negative impact of the non-smoothness on the sequences. It can be seen that the algorithm fares well on even the sporadic motions.

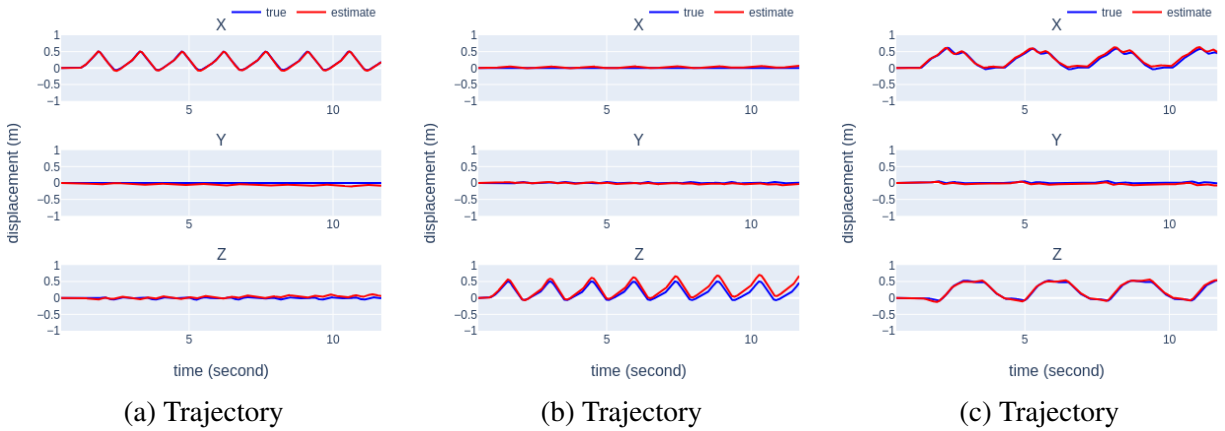


Figure 5.9: Estimated camera egomotion trajectories with linear non-smooth motion in different orientations relative to the scene plane: (a) parallel, (b) perpendicular, and (c) square.

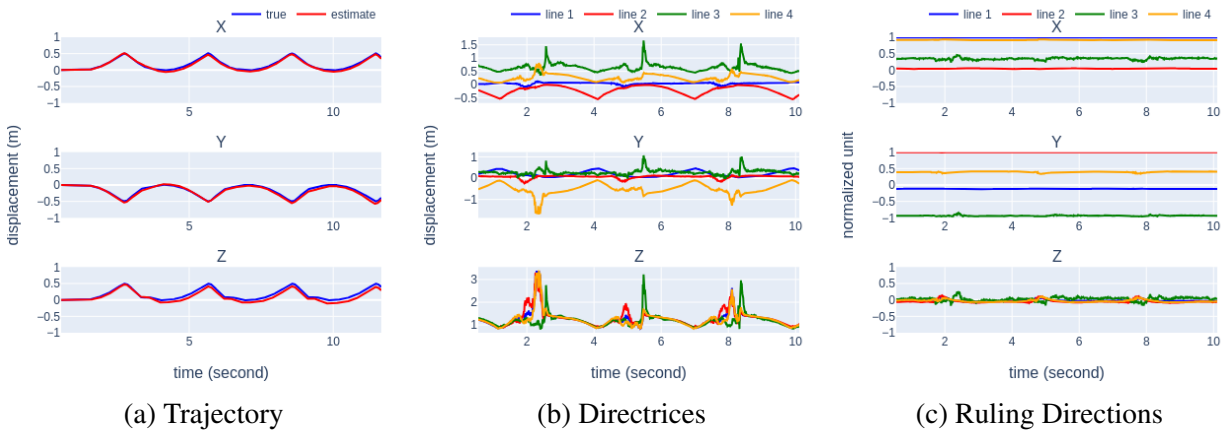


Figure 5.10: (a) Estimated trajectory of camera egomotion, (b) trajectory of the directrix for each ruled surface, and (c) ruling direction for each ruled surface, (a) in world coordinates with respect to the initial camera pose, (b)(c) in derotated camera coordinates; non-smooth linear motion tilted to the scene plane.

<b>Motion</b>	<b>X (m)</b>	<b>Y (m)</b>	<b>Z (m)</b>
Circular	$0.0347 \pm 0.0404$	$0.0189 \pm 0.0387$	$0.2237 \pm 0.1428$
Circular w/ Rotation	$0.0410 \pm 0.0531$	$0.0423 \pm 0.0578$	$0.0373 \pm 0.0543$

Table 5.4: Difference between true and estimated trajectories, non-coplanar scene.

## 5.2.4 Non-coplanar Scenes

Investigating the errors in table 5.4, it can be said that the algorithm achieves comparably accuracy on the non-coplanar scenes. The drift on the world Z axis is once again observed and adding rotations to the motion improves the estimated egomotion. The each directrix has different motion, manifested in figure 5.11b, as the rulings are non-coplanar. Examining figure 5.11c, line 1 is drifting disproportionately in the camera Z direction comparing the other lines. This does signify a potential discrepancy between the estimated and true line pose, though the latter of which is not known in this part of experiments. Nevertheless, the impact of one erroneous ruled surface can be shown to have limited impact on the camera egomotion estimation. Taking a look at figure 5.11a, one can get a sense of how well the camera egomotion was reconstructed.

## 5.2.5 Free Motions

Moving on from the previous setup, in order to see how well the presented research works in less idealized environment, the investigations in this section are done to test sequences with free movements using ground truth from the Vicon motion capture system. The estimation errors are listed in table 5.5. While the algorithm achieved similar performance as previous experiments in the coplanar scene, the estimation on the non-coplanar ones are the least accurate so far. It can be said that it is more difficult to fit surfaces ruled by non-coplanar lines with irregular motions

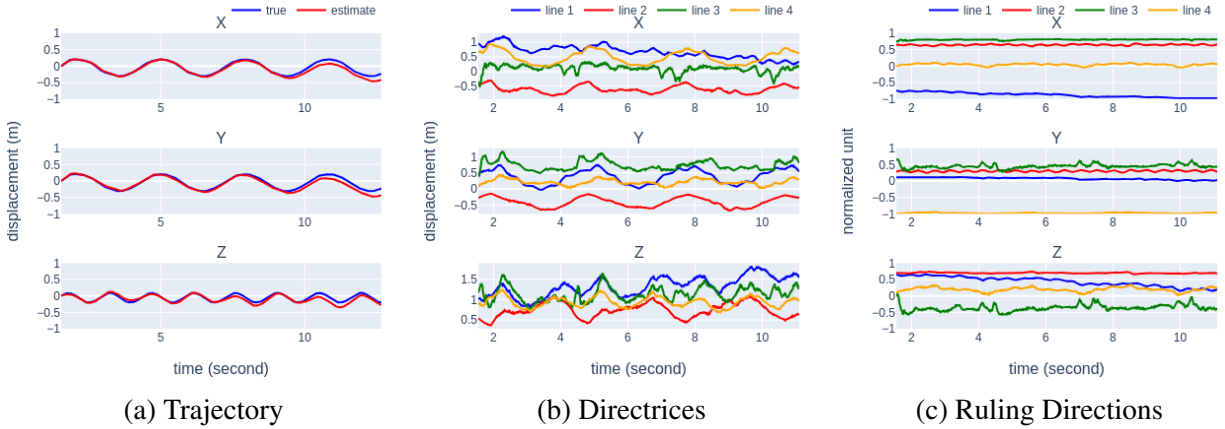


Figure 5.11: (a) Estimated trajectory of camera egomotion, (b) trajectory of the directrix for each ruled surface, and (c) ruling direction for each ruled surface, (a) in world coordinates with respect to the initial camera pose, (b)(c) in derotated camera coordinates with rotating circular motion on non-coplanar scenes.

but considering neither non-smooth motions in section 5.2.3 nor non-coplanar scenes in 5.2.4 had any significant impact on the estimation accuracy, it is not known what exactly is hindering the performance.

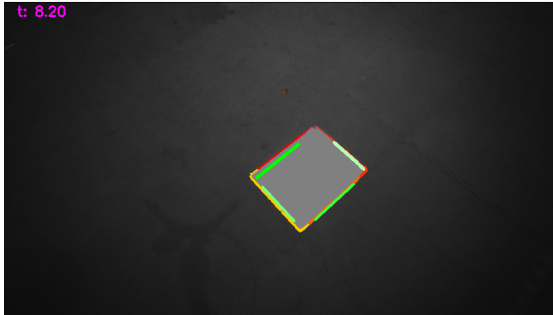
Scene	X (m)	Y (m)	Z (m)	Line 1 (m)	Line 2 (m)
Coplanar 1	$0.0236 \pm 0.0147$	$0.0257 \pm 0.0084$	$0.0207 \pm 0.0140$	$0.0597 \pm 0.0290$	$0.0591 \pm 0.0339$
Coplanar 2	$0.0139 \pm 0.0492$	$0.0519 \pm 0.0785$	$0.0600 \pm 0.0749$	$0.2111 \pm 0.1940$	$0.2061 \pm 0.1412$
Non-Coplanar 1	$0.0821 \pm 0.0642$	$0.0181 \pm 0.0919$	$0.1968 \pm 0.1301$	$0.3713 \pm 0.2017$	$0.3885 \pm 0.2390$
Non-Coplanar 2	$0.2441 \pm 0.1387$	$0.0169 \pm 0.0182$	$0.1739 \pm 0.0614$	$0.3935 \pm 0.1586$	$0.3419 \pm 0.1990$

Table 5.5: Difference between true and estimated trajectories and line poses, under free motion.

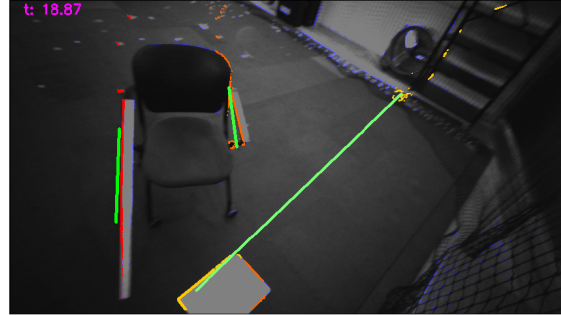
As for the accuracy of the estimated line poses, the distance from Vicon markers that are supposedly on lines to the estimated lines in 3D is computed. Each line is typically marked by 3 to 4 markers and the average between them are taken as the reported result. For each scene two lines are tracked by the Vicon system. In general the quality of line poses is lower comparing to the relative accurate camera egomotion, shown in table 5.5. In the scene of Coplanar 2, it seems even bad (20cm difference) line poses have only limited effects on egomotion (3cm increase in error). On the other hand, bad egomotions in non-coplanar scenes are accompanied by erroneous

line poses estimation. This is because the inertial measurements from the IMU sensor helps constrain the egomotion.

### 5.3 Limitation and Future Work



(a) Coplanar Scene, Clean Background



(b) Non-Coplanar Scene, Noisy Background

Figure 5.12: (a) coplanar scene and (b) non-coplanar scene used in section 5.2.5. The algorithm performs well on the former but struggles on the latter. Since the rulings are considered unbounded noisy points in the background negatively impacts the estimation.

Given all the experiments conducted, the limitation of the presented research is discussed. As evident in section 5.2.5, the algorithm is far from perfection when tested on free motions with non-coplanar scenes, which are more akin to scenarios in the real world. Here it is argued that the decrease in accuracy is not a result of non-coplanarity or non-smoothness of motions, but rather happens during surface growing phase which is done rather naively. Demonstrated in figure 5.12, as the current approach views the rulings as unbounded lines, noisy scene and outlier points belonging to other lines can easily interfere the sliding windows when new frames are added, which in turn further decreases the estimation accuracy in the next fitting step at the end of algorithm 2. Resolving this issue, however, is not as straight forward as one might think since allowing the line to be unbounded reduces its degrees of freedom as discussed in section 2.4 and attempts to associate points with lines may introduce correspondences.

For future work, it is necessary to address this limitation. Introducing additional constraints originated from the normals and curvatures of ruled surfaces could help further bound the estimated egomotion but its effectiveness remains to be tested. It is also interesting to extend the algorithm to curves not just straight lines. Using event camera for better edge detection is also a possible direction towards improving overall performance.

## 5.4 Conclusion

The presented research designed a novel algorithm to solve camera egomotion from visual input through the construction of ruled surfaces. Instead of relying on point-to-point correspondence like traditional approaches, the presented research avoid it by exploiting the geometric properties of ruled surfaces. To reduce the dimensionality of the solution space, measurements from an IMU sensor are used, constraining the estimated motion. Various sequences of videos containing different types of motions and scenes are tested on to prove the accuracy of the algorithm empirically.

## Bibliography

- [1] P. Baker and Y. Aloimonos, “Structure from motion of parallel lines,” in *Computer Vision - ECCV 2004*, T. Pajdla and J. Matas, Eds., Berlin, Heidelberg: Springer, 2004, pp. 229–240. ISBN: 978-3-540-24673-2.
- [2] P. Baker and Y. J. Aloimonos, *Structure from motion on textures: theory and application to calibration*, PhD thesis, University of Maryland at College Park, USA, 2005. ISBN: 0542183110.
- [3] A. Bartoli and P. Sturm, “Structure-from-motion using lines: Representation, triangulation, and bundle adjustment,” *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005. DOI: 10.1016/j.cviu.2005.06.001.
- [4] F. Barranco, C. Fermüller, Y. Aloimonos, and E. Ros, “Joint direct estimation of 3D geometry and 3D motion using spatio temporal gradients,” *Pattern Recognition*, vol. 113, p. 107759, 2021. DOI: 10.1016/j.patcog.2020.107759.
- [5] W. Chamorro, J. Solà, and J. Andrade-Cetto, “Event-Based Line SLAM in Real-Time,” *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8146–8153, Jul. 2022. DOI: 10.1109/LRA.2022.3187266.
- [6] A. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 2, pp. 1403–1410, 2003. DOI: 10.1109/ICCV.2003.1238654.
- [7] A. Evan, “Blade Strike on Landing Ends Mars Helicopter’s Epic Journey.” *IEEE Spectrum*, 27 Jan. 2024. [spectrum.ieee.org/mars-helicopter-ingenuity-end-mission](https://spectrum.ieee.org/mars-helicopter-ingenuity-end-mission).
- [8] C. Fermüller, “Passive navigation as a pattern recognition problem,” *International Journal of Computer Vision*, vol. 14, no. 2, pp. 147–158, Mar. 1995. DOI: 10.1007/BF01418980.

- [9] C. Fermüller and Y. Aloimonos, “On the geometry of visual correspondence,” *International Journal of Computer Vision*, vol. 21, no. 3, pp. 223–247, 1997. DOI: 10.1023/A:1007951901001.
- [10] C. Fermüller and Y. Aloimonos, “Ambiguity in Structure from Motion: Sphere versus Plane,” *International Journal of Computer Vision*, vol. 28, no. 2, pp. 137–154, Jun. 1998. DOI: 10.1023/A:1008063000586.
- [11] C. Fermüller, R. Pless, and Y. Aloimonos, “The Ouchi illusion as an artifact of biased flow estimation,” *Vision Research*, Vol. 40, No. 1, pp. 77–95, 2000. DOI: 10.1016/S0042-6989(99)00162-5.
- [12] C. Fermüller and Y. Aloimonos, “Observability of 3D Motion,” *International Journal of Computer Vision*, vol. 37, no. 1, pp. 43–63, Jun. 2000. DOI: 10.1023/A:1008177429387.
- [13] A. W. Fitzgibbon and A. Zisserman, “Automatic camera recovery for closed or open image sequences,” in *Computer Vision — ECCV’98*, H. Burkhardt and B. Neumann, Eds., Berlin, Heidelberg: Springer, 1998, pp. 311–326. ISBN: 978-3-540-69354-3.
- [14] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conrath, K. Daniilidis, and D. Scaramuzza, “Event-based vision: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 44, No. 1, pp. 154–180, Jan. 2022. DOI: 10.1109/TPAMI.2020.3008413.
- [15] L. Gao, H. Su, D. Gehrig, M. Cannici, D. Scaramuzza, and L. Kneip, “A 5-Point Minimal Solver for Event Camera Relative Motion Estimation,” in *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8015–8025, Oct. 2023. DOI: 10.1109/ICCV51070.2023.00739.
- [16] L. Gao, D. Gehrig, H. Su, D. Scaramuzza, and L. Kneip, “An N-Point Linear Solver for Line and Motion Estimation with Event Cameras,” in *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14596–14605, Jun. 2024. DOI: 10.1109/CVPR52733.2024.01383.
- [17] Y. He, J. Zhao, Y. Guo, W. He, and K. Yuan, “PL-VIO: Tightly-coupled monocular visual–inertial odometry using point and line features,” *Sensors*, Vol. 18, 2018. Available: <https://api.semanticscholar.org/CorpusID:4793973>.
- [18] A. Heyden, “Three-dimensional geometric computer vision,” in *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neural Computing, and Robotics*, pp. 305–347, Springer, Berlin, Heidelberg, 2005. DOI: 10.1007/3-540-28247-5\_10.

- [19] S. Izumiya and N. Takeuchi, “Special curves and ruled surfaces,” *Beiträge zur Algebra und Geometrie*, Vol. 44, No. 1, pp. 203–212, 2003.
- [20] A. Kendall, M. Grimes, and R. Cipolla, “PoseNet: A convolutional network for real-time 6-DOF camera relocalization,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, Los Alamitos, CA: IEEE Computer Society, pp. 2938–2946, Dec. 2015. DOI: 10.1109/ICCV.2015.336.
- [21] C. Le Gentil, F. Tschopp, I. Alzugaray, T. Vidal-Calleja, R. Siegwart, and J. Nieto, *IDOL: A Framework for IMU-DVS Odometry using Lines*, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020. DOI: 10.1109/iros45743.2020.9341208.
- [22] J. Nagata, Y. Sekikawa, and Y. Aoki, “Optical flow estimation by matching time surface with event-based cameras,” *Sensors*, Vol. 21, No. 4, Article 1150, 2021. DOI: 10.3390/s21041150.
- [23] C. M. Parameshwara, G. Hari, C. Fermuller, N. J. Sanket, and Y. Aloimonos, “DiffPoseNet: Direct differentiable camera pose estimation,” in *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA: IEEE Computer Society, pp. 6835–6844, Jun. 2022. DOI: 10.1109/CVPR52688.2022.00672.
- [24] J. Plücker, “On a new geometry of space,” *Philosophical Transactions of the Royal Society of London*, Vol. 155, pp. 725–791, 1865.
- [25] Z. Ren, B. Liao, D. Kong, J. Li, P. Liu, L. Kneip, G. Gallego, and Y. Zhou, “Motion and structure from event-based normal flow,” arXiv preprint, arXiv:2407.12239, 2024. DOI: 10.48550/arXiv.2407.12239.
- [26] G. Schindler, P. Krishnamurthy, and F. Dellaert, “Line-based structure from motion for urban environments,” in *Proc. Third Int. Symp. 3D Data Processing, Visualization, and Transmission (3DPVT)*, pp. 846–853, 2006. DOI: 10.1109/3DPVT.2006.90.
- [27] F. Shevlin, “Analysis of orientation problems using Plucker lines,” in *Proceedings of the Fourteenth International Conference on Pattern Recognition*, vol. 1, pp. 685–689, 1998. DOI: 10.1109/ICPR.1998.711236.
- [28] J. Shin, S. Yun, and A. Kim, “PeLiCal: Targetless Extrinsic Calibration via Penetrating Lines for RGB-D Cameras with Limited Co-visibility,” in *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14506–14512, 2024. DOI: 10.1109/ICRA57147.2024.10611415.

- [29] M. E. Spetsakis and J. Aloimonos, "Structure from motion using line correspondences," *International Journal of Computer Vision*, vol. 4, no. 3, pp. 171–183, Jun. 1990. DOI: 10.1007/BF00054994.
- [30] L. Yang, "Ego-motion estimation based on fusion of images and events," arXiv preprint, arXiv:2207.05588, 2022. Available: <https://arxiv.org/abs/2207.05588>.
- [31] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," arXiv preprint, arXiv:1812.08156, 2018. Available: <https://arxiv.org/abs/1812.08156>.