Autonomous Path Following via Optical Flow of Environmental Features

Pranav Tripathi[1,2], Xinhuan Sang[2], Roberto Tron[2]

Eastlake High School, 400 228th Ave NE, WA 98074[1]

Boston University Robotics Lab, 750 Commonwealth Ave B29, MA 02446[2]

Introduction

Background

- Traditional methods of autonomous navigation typically rely on positional data or other sensor inputs (ultrasonic, LiDAR, etc.)
- Such data is unavailable in many environments, limiting the scope of these methods

Objective

- In theory, a camera-enabled device can center itself on a path using the optical flow (apparent motion) of environmental features (points in the environment) [1]
- This project aims to design an algorithm to guide a Tello drone on a path [2], by tracking AprilTags in the environment **(Fig. 1.)**

Fig. 1. Process diagram of drone tracking optical flow

Methods

Optical Flow Calculations

- Applied rigid body transformations **(1)** and pinhole camera model **(2)** to calculate velocity of features in simulation
- Used OpenCV point tracking to find velocities (in pixels/second) in real life

$$(1)\quad {}^{b}\dot{\mathbf{p}} = {}^{b}\mathbf{R}_{w}\omega \cdot {}^{w}\mathbf{p} + {}^{w}\dot{\mathbf{T}}_{b} \qquad (2)\quad \dot{\mathbf{x}} = \frac{\mathbf{V}_x \cdot \mathbf{z}_o - \mathbf{V}_z \cdot \mathbf{x}_o}{{\mathbf{z}_o}^2}$$

Velocity Filtering

- Gaussian Filtered Derivative – Multiplies 9 frames of positional data with derivative of a normal distribution, filtering noise **(Fig. 2.)**
- Normalization – Compresses velocities from 0 to 1, minimizing discrepancies **(Fig. 3.)**
- MAD Filter – Filters extreme outliers by finding deviation from median **(Fig. 4.)**

Point Tracking and Masking

- AprilTags placed in the environment provide reliable tracking using the centers of tags [3]
- Mask **(Fig. 5.)** helps only select points in specific regions, minimizing unwanted data

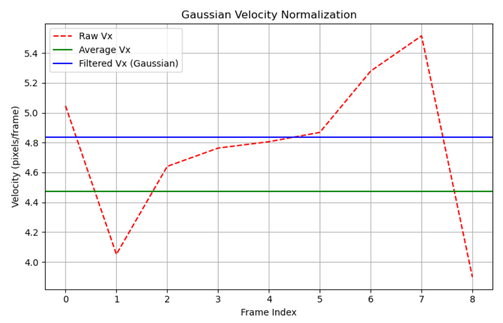

Fig. 2. Velocity from Gaussian Filter vs. Average

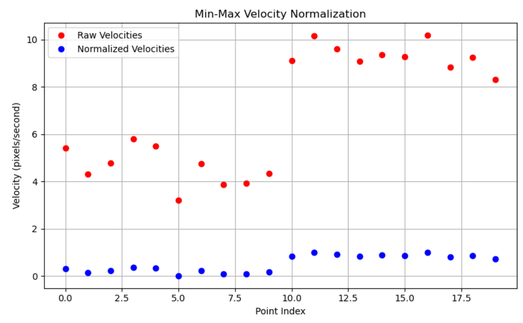

Fig. 3. Normalized velocities via min-max method

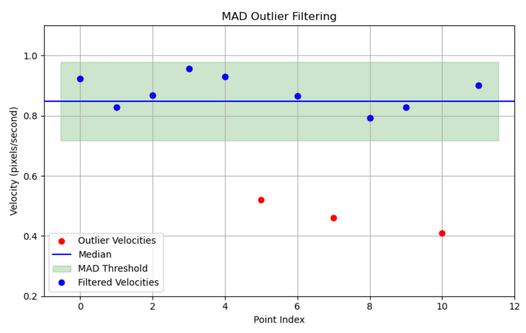

Fig. 4. Outlier filtering using median-absolute deviation

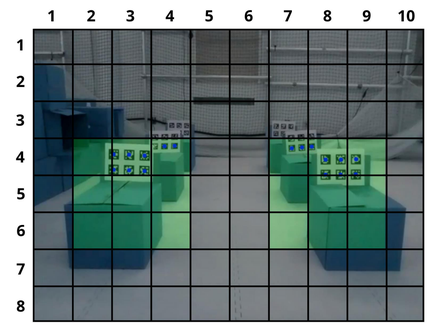

Fig. 5. 8x10 grid used to create mask

Results

Control Fields

- Simulated features are mapped to vector fields showing optical flow from varying distances **(Fig. 6, 7.)**
- Fields of each feature are summed to create an overall control field **(Fig. 8, 9.)**, directed towards the center

Point Identification & Tracking

- Drone was able to identify and track points within masks even at high angles/offsets **(Fig. 10.)**
- Mask helped with separation of left and right-side points, especially on curved paths **(Fig. 11.)**

Course Correction

- Simulation – drone corrected and stayed within 5 units of center, on a 120-unit wide path (**Fig. 12, 13.**)
- Real-life – drone successfully centered when drifting left **(Fig. 14.)** and followed a curved path **(Fig. 15.)**

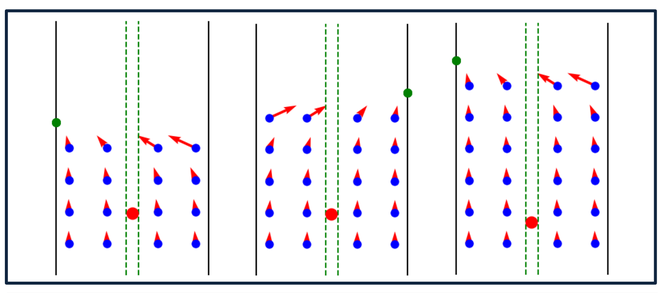

Fig. 6. Vector fields of test points on straight path

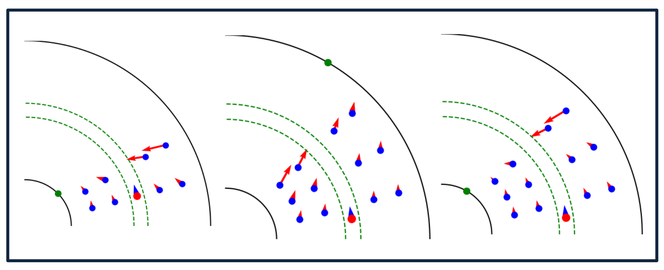

Fig. 7. Vector fields of test points on curved path

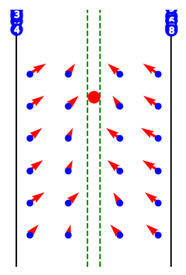

Fig. 8. Overall control field on straight path

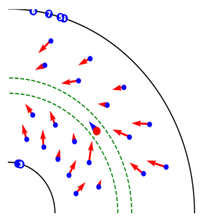

Fig. 9. Overall control field on curved path

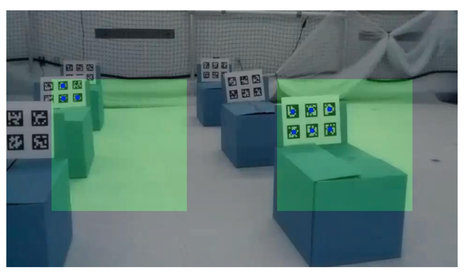

Fig. 10. Points tracked by drone at 30-degree angle

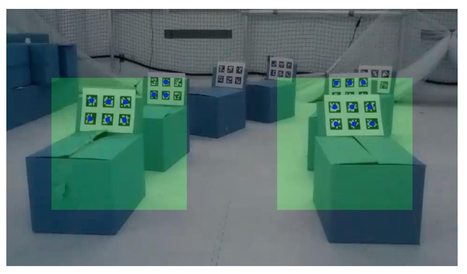

Fig. 11. Points tracked on each side in curved path

Fig. 12. Simulation of algorithm on curved path

Fig. 13. Simulation of algorithm on straight path

Fig. 14. Drone centering and counteracting drift

Fig. 15. Drone following curved path

Conclusions

Optical Flow

- Tracking the velocities of features was accurate in both simulation and real-life tests. Using AprilTags provided a reliable method of tracking a variety of points across frames
- The Gaussian filter used to calculate initial velocities was robust to noise from a few skewed frames. Min/max normalization made velocity values more reliable for calculations, by compressing discrepancies. MAD filtering was also very helpful, removing large outliers

Path Following

- Algorithm successfully generated signals to follow paths and orient drone towards the center
- Creating masks in real-life images was very beneficial—it prevented significant outliers from very close/far points, and allowed for accurate detection of which side a point was on

Future Work

- Could find better methods of scaling the signals to be stronger at edges and weak in the center
- Algorithm can be modified to handle more difficult paths (e.g., a 90-degree corner) and extreme starting conditions (e.g., a starting offset of 45 degrees). One way to achieve this would be adaptive mask generation, requiring a deeper understanding of the environment

References

[1] Boretti, C.; Bich, P.; Zhang, Y.; Baillieul, J. Visual Navigation Using Sparse Optical Flow and Time-To-Transit. arXiv.org 2021. https://doi.org/10.1038/293293a0).

[2] Tron, R. pythonDJITello. Bitbucket.org. https://bitbucket.org/tronroberto/pythondjitello/src/master.

[3] Tron, R. apriltag. Bitbucket.org. https://bitbucket.org/tronroberto/ros2me416_apriltag/src/main.

Additional information available at https://pranavtrip.blogspot.com.

Acknowledgements

I am sincerely grateful to everyone who made this project possible. I would especially like to thank Professor Tron and Leo for their mentorship throughout the project and for enabling my success. I would also like to thank the RISE program for providing me with this opportunity, and for the incredible facilities and resources available at Boston University. Finally, I would like to thank my parents for supporting me throughout this process as well.