Motion Estimation and Velocity Tracking of a Flexible Link

Aamna Alshehhi 100062675

December 2024

Abstract

This project investigates the use of discrete video streams (DVS) to estimate the motion and velocity of a soft flexible link. The approach involves a series of preprocessing steps, contour extraction, alpha shape analysis, and velocity estimation using optical flow and Kalman filtering. The main goal is to track and visualize the movement of the robotic link while estimating the velocity at key points along the link's length.

1 Introduction

1.1 Background

In robotics, the ability to accurately track and analyze soft robotic components is crucial for various applications including automation, health robotics, and precision movements. Estimating the motion of soft robots presents significant challenges due to their deformable nature. DVS (Discrete Video Stream) data offers a way to capture these deformations, enabling real-time analysis of the link's movement.

1.1.1 Objectives

The primary objectives of this project are:

- To develop a pipeline for accurate velocity estimation of a soft robotic link using a DVS video.
- To visualize the movement of the link and estimate the velocity at multiple points along the link.
- To incorporate optical flow and Kalman filtering techniques for enhanced motion tracking.

1.1.2 Challenges

Challenges in the project include:

- Dealing with noise and artifacts from the DVS video.
- Extracting reliable contours to represent the soft link across frames.
- Predicting the link's movement even when parts of it are obscured or missing using Kalman filtering.
- Handling deformation of the link, which affects the accuracy of velocity estimation.

2 Literature Review

Soft robotics has emerged as a significant field of study, focusing on the manipulation and control of robots made from highly flexible materials. These robots exhibit complex, highly deformable structures that traditional rigid-body dynamics cannot accurately model. Essential to the development and deployment of soft robotics is the capability to precisely estimate motion and track velocity of flexible components in real-time.

2.1 Optical Flow in Soft Robotics

Optical flow methods have been adapted to track the detailed motion of flexible materials. One prominent approach is the polynomial expansion method by Farneback, which computes dense optical flow to capture the motion at every image pixel[1]. This method has been instrumental in studying the dynamics of soft robotic components by providing detailed motion vectors that reflect the actual deformation between image sequences.

2.2 Kalman Filtering for Motion Prediction

Kalman filtering is another cornerstone technique used to enhance motion tracking in environments with uncertain information. Originally developed by R.E. Kalman, this filter has been widely applied for predictive tracking in soft robotics, allowing for the estimation of the system state amid noise and partial occlusions [2]. The use of Kalman filtering in soft robotics helps in compensating for the unpredictable behaviors of flexible materials, by predicting the future states based on a series of noisy measurements.

2.3 Challenges and Recent Developments

Despite the advancements, several challenges remain, primarily due to the non-linear properties of soft materials and the complex, unpredictable environmental interactions affecting the sensors. Recent studies have suggested enhancements such as the Unscented Kalman Filter (UKF), which provides a better approximation for highly non-linear systems [3]. Additionally, integrating deep learning methods with traditional optical flow has shown promise in handling complex deformations and rapid movements more robustly [4].

3 Methodology

3.1 Pipeline Overview

The pipeline can be divided into the following steps:

1. Preprocessing
2. Contour Extraction
3. Smoothing and Outlier Removal
4. Motion Estimation and Tracking with Optical Flow and Kalman Filtering
5. Velocity Analysis

3.2 Detailed Description

3.2.1 Data Acquisition

The video was captured from a DVS camera. The resolution of the video is 640x480 pixels, with known dimensions of $40\,\text{cm} \times 50\,\text{cm}$. This information is used for converting pixel values into metric measurements during velocity estimation.

3.2.2 Preprocessing

- **Adaptive Thresholding:** Adaptive thresholding is applied to handle varying lighting conditions in the video. This technique calculates different threshold values for different regions in the frame, making it ideal for non-uniform illumination.

$$T(x, y) = \text{mean}_{\text{neighborhood}}(x, y) - C \tag{1}$$

- **Gaussian Blur:** A Gaussian blur is used to smooth the frames while preserving edges. The Gaussian function is defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

where σ determines the level of smoothing.

- **Morphological Operations:** Morphological closing is used to fill small gaps and remove noise:

$$\text{Morph}_{\text{close}}(I) = (I \oplus K) \ominus K \tag{3}$$

where I is the binary image, K is the structuring element (kernel), $\oplus$ represents dilation, and $\ominus$ represents erosion.

- **Edge Detection:** The Canny edge detection algorithm is used to highlight the structure of the link:

$$E(x, y) = \begin{cases} 255 & \text{if } G(x, y) \geq T_{\text{high}} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $G(x, y)$ represents the gradient magnitude.

3.2.3 Contour Extraction and Outlier Filtering

Contours are extracted using the detected edges. The points representing these contours are filtered using DBSCAN to remove outliers:

$$\text{DBSCAN}(P, \epsilon, m) \rightarrow \{C_1, C_2, \ldots, C_k\} \tag{5}$$

where P represents the extracted points, ϵ is the neighborhood radius, and m is the minimum number of points required to form a cluster.

3.2.4 Alpha Shape Application

To compute a concave hull that closely represents the structure of the link, an alpha shape is used:

$$\text{Alpha Shape}(P, \alpha) = \text{Concave Hull}(P, \alpha) \tag{6}$$

where α controls the tightness of the hull. This helps create a smooth boundary around the link.

3.2.5 Tracking with Optical Flow and Kalman Filtering

Optical Flow: Farneback optical flow is applied to track changes in the pixel intensity pattern between frames:

$$f(x, y, t) = f(x + u, y + v, t + 1) \tag{7}$$

where (u, v) are the flow vectors that describe the displacement of a pixel.

Kalman Filtering: The Kalman filter is used to predict the centroid's position when parts of the link are missing or obscured. The prediction and update equations are:

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_{k-1} + w_k \tag{8}$$

$$z_k = H\hat{x}_k + v_k \tag{9}$$

where A is the state transition matrix, B is the control input, H is the observation model, and w_k, v_k are process and measurement noise, respectively.

3.2.6 Velocity Estimation

Velocity estimation is performed on three specific points along the link. The conversion from pixel to meter units is carried out as follows:

$$(x_{\text{meters}}, y_{\text{meters}}) = \left(\frac{x_{\text{pixels}}}{W_{\text{frame}}} \times W_{\text{video}}, \frac{y_{\text{pixels}}}{H_{\text{frame}}} \times H_{\text{video}} \right) \tag{10}$$

Velocity is then calculated as:

$$v = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{\frac{1}{\text{fps}}} \tag{11}$$

3.2.7 Visualization

The final velocity values for specific points along the link and the shape are visualized using OpenCV.

3.3 Implementation

3.3.1 Programming Language and Libraries

The implementation was done using Python. Key libraries used include

- OpenCV for image processing
- DBSCAN for clustering
- KalmanFilter from the `pykalman` library for motion prediction.

3.3.2 Code Flow

The flow follows a systematic approach:

- Load frames from the video.
- Preprocess each frame to reduce noise and extract the edges.
- Extract contours and use Alpha shape to compute a smooth representation of the link.
- Track the link using optical flow and Kalman filter.
- Estimate velocity at key points along the link and visualize it on the frames.

The results of the implemented pipeline is shown here for a specific frame `https://aamnaal.github.io/RP-Project/#motion-estimation`

3.3.3 Challenges in Implementation

Challenges included the variability of the link's shape, noise in the video, and difficulties in parameter tuning for the filters and clustering algorithms.

4 Results

4.0.1 Velocity Estimation Visualization

The estimated velocity for each of the three key points along the link was displayed on the video. The visualization included tracking of the link shape and video can be found here `https://aamnaal.github.io/RP-Project/#motion-estimation`.

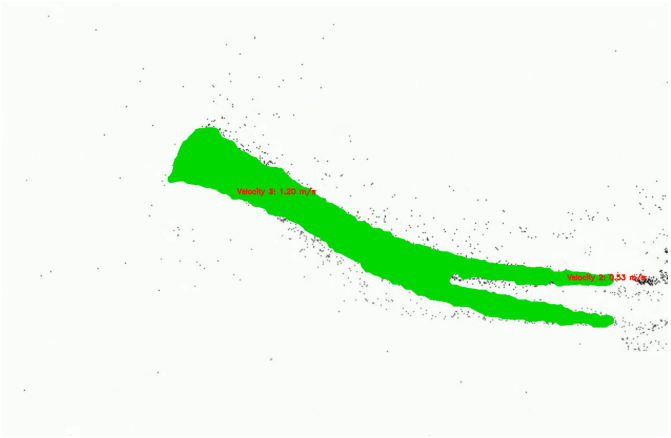

Figure 1: Velocity Estimation Visualization

4.0.2 Velocity Analysis

The velocity of the link varied significantly due to its deformable nature. Optical flow provided real-time estimation, while the Kalman filter helped reduce noise in the movement prediction.

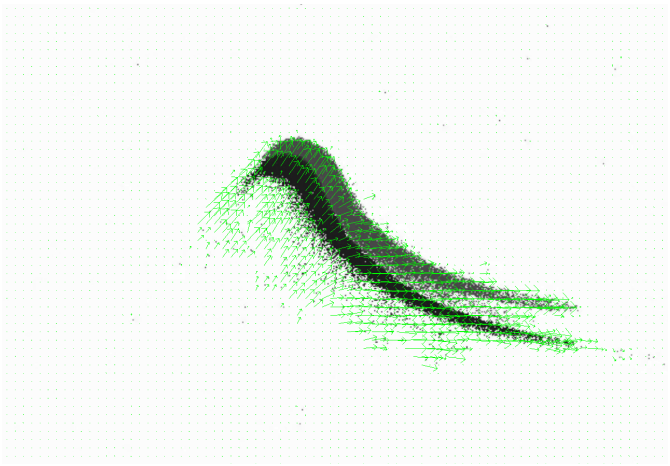

Figure 2: Optical Flow Representation

4.1 Discussion

4.1.1 Accuracy and Improvements

The velocity estimation in this project was found to be fairly accurate under controlled conditions where the entire link was fully visible throughout the video. This is largely because the combination of alpha shape contour fitting and optical flow worked well to accurately represent the motion of the link, while the Kalman filter was able to smooth out minor irregularities in movement.

However, in scenarios where parts of the link were obscured or had a more complex movement (such as bending or twisting out of the camera's plane), there were some challenges that affected accuracy:

- **Partial Visibility:** When parts of the link were not captured by the DVS, the performance of the velocity estimation declined. The Kalman filter, although effective in predicting positions of missing data points, is based on a linear prediction model. For more complex non-linear dynamics, this leads to inaccuracies as it assumes constant velocity or simple changes, which does not hold for the highly deformable nature of the link.

- **Deformable Nature:** The deformable nature of the link made it difficult to consistently track the entire shape. The optical flow approach used (Farneback) is sensitive to large displacements and changes in appearance that occur due to bending. In particular, Farneback optical flow works well for relatively small displacements but struggles with complex deformations or large motions.
- **Kalman Filter Assumptions:** The Kalman filter assumes that the noise is Gaussian and that the motion follows linear dynamics, which is often not the case for soft robotic components. For instance, when the link exhibits sudden movements, bending, or twisting motions, the Kalman filter's linear model fails to capture these rapid non-linear changes, resulting in inaccuracies in position prediction. To improve accuracy under these scenarios, the Kalman filter could be replaced or augmented with a non-linear variant such as the Unscented Kalman Filter (UKF).
- **Temporal Consistency and Smoothing:** The calculated velocity, particularly during rapid or jerky movements, exhibited significant fluctuations due to the deformable nature of the link and errors in tracking individual points. To improve the temporal consistency of velocity estimation, additional smoothing techniques could be applied. For instance, Savitzky-Golay filtering could be applied to the velocity data to provide a smoother trajectory and velocity profile, which would help in reducing spikes caused by noise or inaccuracies in frame-to-frame tracking.

Overall, while the initial pipeline worked reasonably well under controlled conditions, the aforementioned improvements could significantly enhance the robustness and accuracy of the tracking and velocity estimation. These improvements would allow the system to handle more challenging scenarios involving partial occlusions, rapid movements, and complex deformations more effectively.

4.2 Conclusion

The developed pipeline effectively tracks and estimates the velocity of a soft robotic link using a combination of optical flow and Kalman filtering. This integrated approach provides robust results under controlled conditions, ensuring accurate tracking when the link is fully visible and its movement is relatively smooth. The use of adaptive thresholding, DBSCAN filtering, and alpha shape fitting has been pivotal in identifying the link structure effectively.

However, certain challenges still exist in consistently detecting the link in all frames, especially under conditions of rapid motion, or significant noise in the video feed. The complex deformations of the link, such as bending, twisting, and stretching, present additional challenges that are difficult for the linear Kalman filter to predict accurately. Incorporating advanced non-linear models and more sophisticated optical flow techniques, as well as introducing physics-based dynamics, could help overcome these challenges. These improvements would make the pipeline more resilient, allowing for better performance in scenarios with highly dynamic and deformable structures, ultimately increasing the accuracy and reliability of velocity estimation for soft robotic components.

4.3 Future Work

Future work includes incorporating more advanced tracking methods such as deep learning-based approaches for more robust detection and prediction, using physics-based simulations to model the link's behavior, and applying RAFT optical flow for enhanced accuracy.

References

[1] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Scandinavian Conference on Image Analysis*, pp. 363–370, 2003.

[2] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[3] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *ASME 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158, IEEE, 2000.

[4] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of Imaging Understanding Workshop*, 1981.

[6] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. 1, pp. 35-45, 1960.

[7] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Proceedings of the Scandinavian Conference on Image Analysis*, 2003.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226-231.

[9] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the Shape of a Set of Points in the Plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551-559, 1983.

[10] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.

[11] PyKalman Library, "Kalman Filtering and Smoothing in Python." [Online]. Available: `https://pykalman.github.io/`. [Accessed: Nov. 28, 2024].

[12] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.