Project 4: Neuromorphic Vision! Event-Based Visual Flow

Wenyan Li
Email: wenyanli@umd.edu

***Abstract*—This project mainly focuses on computing the visual flow with the provided data by following the algorithm given in Ref. [1] In order to fit the plane for the event points, both methods of least square minimization and Principal Component Analysis(PCA) have been tried. By comparing the results, PCA gives a better result as expected.**

I. Introduction

The implementation of the algorithm can be generalized into three steps:

1) Create spatiotemporal neighborhood for each event point. In this step, elementwise function bsxfun() and find() function are used for finding the neighbor points and code simplicity.
2) Fitting the plane for each neighborhood. Both least square minimization and PCA are taken to estimate the parameters of the plane. The results are analysed and compared.
3) Attribute velocity to each event using parameters of the best fitted plane.

II. Create spatiotemporal neighborhood

In this part, spatiotemporal neighborhoods are created for each event.The events are visualized as shown in Fig.1. And the events of different polarity are shown in Fig.2.

First, we group the events within certain time intervals, i.e. create a temoral neighborhood. Then in this temoral neighborhood, neighbor events are selected by defining a spatio range for each event position. $bsxfun()$ function is used for elementwise calculation, and $find()$ function is used to obtain the index of the neighbors within the defined range.

III. Fitting the plane

In order to fit the in each neighborhood, both methods of least square minimization and PCA are used.

A. Least square minimization

The Eq.(6) in Ref. [1]. can be implemented as finding the vector Π that minimize the equation $\Pi^T S \Pi$, where S is $X^T X$, and X is the $4 \times N$ matrix which each column represents an event$(x_i, y_i, t_i, 1)$. By taking derivative, we need to solve the equation $\Pi^T S = 0$, where we SVD can help. So our solution is the eigenvector corresponds to the smallest eigenvalue of S, i.e.$V(:, 4)$. Then by following step 4, 5, and 6 in algorithm 1 (same parameters except the neighborhood range are used as suggested in the paper) in Ref. [1] we can refine the fitted plane and obtain the estimated parameters $(a, b, c, d)^T$.

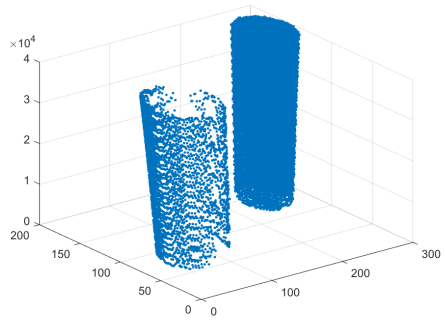

Fig. 1: 3D plot of (x, y, ts)

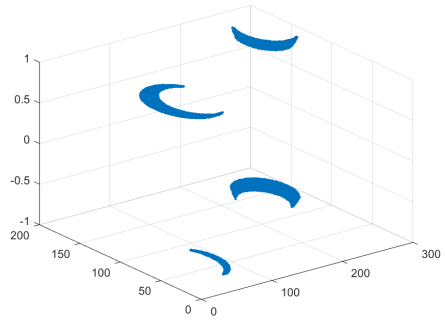

Fig. 2: Events of different polarity

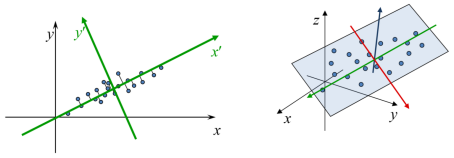

Fig. 3: PCA help us minimize the orthogonal distances

B. PCA

Compared to least square minimization which minimize the least square distances of the events from the plane, PCA finds an orthogonal basis that best represents a given data set, as shown in Fig.3. In PCA, the fitted plane is assumed to pass the centroid of the points and thus the centroid and the covariance is calculated in function applyPCA().

Again in PCA, SVD is used and we pick the last column as the normal of our fitted plane.

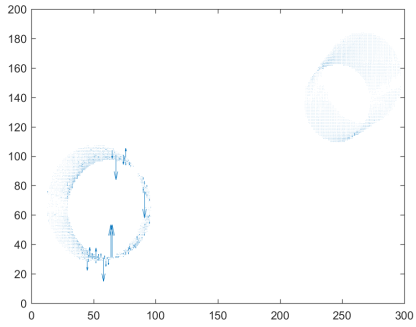

Fig. 4: Result of using least square minimization

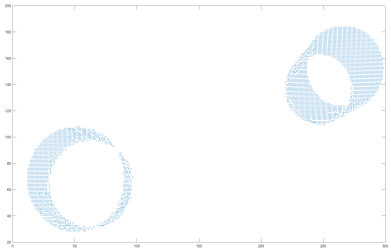

Fig. 5: Result of using PCA

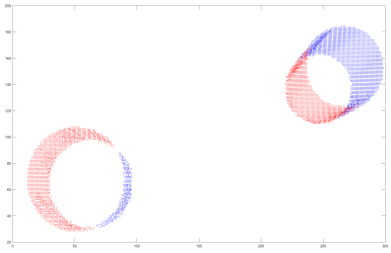

Fig. 6: Event polarity display

IV. Attribute velocity

The plane is described by $ax + by + ct + d = 0$, then (v_x, v_y) of each point at position(x, y) is the partial derivitive of x and y of function $t = -\frac{a}{c}x - \frac{b}{c}y - \frac{b}{d}$. So (v_x, v_y) is simply $(-\frac{a}{c}, -\frac{b}{c})$. By attributing the velocity to each event, we can obtain the visual flow. The result is shown in Fig.4 and Fig.5. In Fig.6, the polarity is also colored.

V. Observation and Discussion

By comparing the result using different methods for plane fitting, PCA outperforms least square minimization as expected. Though, the result shown in Fig.5 seems to be what we've expected as the circles become enlarged and with visual flows pointing to the corners. If give a close look to the visual flows, as shwon in Fig,7, not all of them are consistent with their neighbors. And if we reduce the time interval used in our neighborhood, the result can be effected, as shown in Fig.8. Meanwhile, considering the value of the velocity of

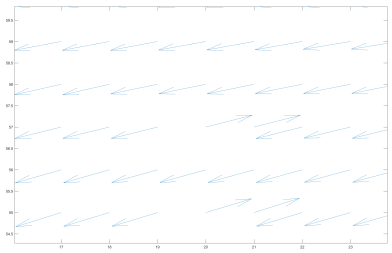

Fig. 7: Left corner taken from Fig.5

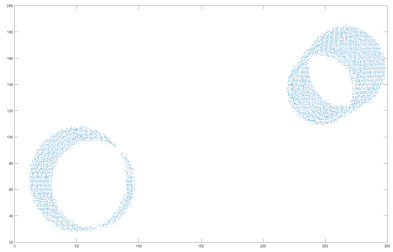

Fig. 8: Results affected by adjusting the neighborhood range

the events won't vary too much, so here we mainly focus on the orientation of the flow, and only takes the sign of c into consideration during calculation.

The reason that I think which may accout for the inconsistence of the visual flow between neighbors is that PCA may return arbitrarity oriented eigenvectors, i.e. the normal we obtained for the fitted plane is the opposite (or not consistent) with its neighbors, which may further influence the event flow. The solution for this is to ensure that the neighoboring points should have similar normals. Another possible reason for the inconsistence is the wrong allocation of points when creating neighborhood(which may explains why the time interval affects the result).

VI. Conclusion

In this project, event-based visual flows are successfully computed and visualized for the provided data collected with dynamic vision sensor. However, further work can be done to reduce inconsistency and refining the flow visualization.

References

[1] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi., *Event-based visual flow*.IEEE Transactions on Neural Networks and Learning Systems,25(2):407-417, 2014.