Technical Report:
Clustering Optical Flow Tracks using Delaunay triangulation

Ramesh Marikhu
Computer Science and Information Management,
Asian Institute of Technology,
Pathumthani - 12120,
Thailand

Dr. Matthew N. Dailey
Computer Science and Information Management,
Asian Institute of Technology,
Pathumthani - 12120,
Thailand

Dr. Mongkol Ekpanyapong
Industrial Systems Engineering,
Asian Institute of Technology,
Pathumthani - 12120,
Thailand

Computer Science and Information Management & Industrial Systems Engineering
Asian Institute of Technology
Pathumthani -12120
Thailand

25 April 2016
Introduction

Detecting objects in the scene is the first step in the image processing tasks related to obtaining information about objects in the scene, be it shape, structure, motion and pose estimates, motion statistics etc. Clustering optical flows has been employed by Beymer et al., Saunier et al. and in many other research works. Here, Delaunay triangles are analyzed for common motion to enable identifying edges and vertices that do not comply to the constraints and are then removed to obtain triangle clusters first on image only, then on the ground plane and later including additional motion constraints to aid in effective clustering of objects in motion.

Table of contents

Introduction
Table of contents
Methodology
Analyzing optical flow tracks using Delaunay triangulation
  System run on Sutthisan video
  System Flowchart
  System initialization
  System run
    Extracted features (goodFeaturesToTrack())
    Optical flows
    Feature Tracks
    Delaunay triangulation
  Clustering by simple threshold on proximity and velocity analysis on triangle pair on image
    Analyzing triangle pair
      Example clustering results
  Clustering by simple threshold on proximity and velocity analysis on triangle pair on the ground plane.
    Analyzing triangle pairs
      Overview image to compare results between image based and ground plane based approach
  Clustering by threshold on 3D motion consistency runs on Sutthisan video
System specification
Methodology

- Estimate H using manually specified point correspondences
- Estimate K using parallel lines on the ground, perpendicular distance between the parallel lines and camera height.
- Given K, H, estimate P
- Extract optical flow between frames
- Track optical flows
- Validate optical flow tracks
- For each optical flow, projected onto normalized camera plane Z=1 and then considered in World coordinates, extract length of OF for a range of heights. The length will be max at h=0.0f; OF is considered on the ground.
- Under segment the OFs considering all OFs on the ground plane and then performing Delaunay triangulation using the tail of the latest OF of the track. Initially, there will be a single cluster.
- Obtain displacement of Delaunay triangles using optical flow. This will give a set of triangles and the optical flow will link the triangles from Delaunay triangulation with the new ones.
- Analyze triangle pairs considering features on the image
- Analyze triangle pairs considering features on the ground plane.
- Analyze triangle pairs with 3D motion constraints
  - Given a set of clusters at this point from Delaunay triangles (under clustering preferred), analyze the optical flows in the cluster. Select an OF at a particular height and get the no. of OFs that agree to the length of the chosen OF.
  - Optimization problem: For a particular frame with validated tracks considered, identify the height of OFs for which the score of each OF in the cluster is high. Optimize for every track.
  - For the current case, min height = 0.0m and max height = 3.0m will be considered for analysis and later, additional height levels will be considered for optimization.
Analyzing optical flow tracks using Delaunay triangulation

System run on Sutthisan video

Video filename (./Sutthisan/Thu 28 Jan 2016 09-03-06/2016-01-28 09:03:06.avi), AVI, Resolution (640x480), color, frame rate (30fps), video length (14m 54s), actual duration (1 hour), file size (809.5 MB)

Actual Resolution: 640x480
After specifying ROI: 616x440

Execution script:
./exec/LCD1 -location "Sutthisan" -stationid 1 \
-c1 emulator -n1 1 -cm1 1 -w1 640 -h1 480 -resizeimage 480 -p1 
"/media/marikhu/work/LaneChange_misc/Sutthisan/Thu 28 Jan 2016 09-03-06/2016-01-28 09:03:06.avi" -timestamp "/media/marikhu/work/LaneChange_misc/Sutthisan/Thu 28 Jan 2016 09-03-06/2016-01-28 09:03:06.yml" \
-fps 60 -process -enablelonchanchedetectionoftracks -debugmode 0 -dbconfig 
"./db/dbConfig.yml" -numframestoskip 0 -license ~/license_LaneChange.txt -paramsfile 
"./input/Sutthisan.params" -outputfolder "/../output/" -c 
"../input/Sutthisan640x480.config" -setregionsfromconfigfile -saveviolation -savesnapshot -savesnaphotothd -savevideoclip 
corrlptsfile "/../calib/Sutthisan_extrinsic_corrpts.yml" -extrinsiccalibfile 
"../calib/Sutthisan_extrinsic.yml" -dispimgtracks -dispimgresults -displocation -dispprocarea 
disptracks -disptimestamp -dispprocframerate -dispprocarea -dispimground 
-numframestograb 200
System Flowchart

System initialization

Processing ROI

Figure: User-defined processing area. Defining ROI increase processing frame rate as the regions that are of no interest as per the application can be ignored.
User-defined regions

Figure: User-defined regions specific to application.
Homography estimation

Figure: World ground plane overlaid. Green rectangle (3mx3m) is defined by user to estimate homography between the ground plane on image and ground plane in world frame.

System run

Extracted features (goodFeaturesToTrack())

Figure: (a) Previous image (Frame no. 4). (b) Current image (Frame no. 5).
Figure: Features extracted (displayed as blue dots) from the displayed frame (Frame no. 4) using goodFeaturesToTrack() function in OpenCV. The processing image is cropped out of the actual image and the regions outside processing image is set to black. This causes features to be detected along the slanted boundaries of the processing ROI which can be easily filtered out before actual processing of detected features.

Parameters used for goodFeaturesToTrack() for feature detection:
- Min distance = 2 px
- Block size = 5 x 5
- Use Harris = false (if true, Kappa = 0.04 (default) [0.04 - 0.15])
- Quality level: 0.05
- Max no. of features = 1000

**Including features from validated tracks**
For better tracking of features, the features from the head of the validated tracks are included into the set of extracted tracks and duplicate features are removed. This ensures that the good tracks still have chance to continue the tracks along the trajectory.
Figure: Features extracted on Frame 79.
Figure: The features (blue and green) obtained on Frame No. 79 are to be used for extracting optical flow using the next frame (Frame No. 80). The blue features are the ones that are obtained from the existing tracks, those that were not extracted in Frame No. 79.
Optical flows

Figure: Optical flow (green lines) draw on previous image (Frame no. 79) overlaid on current image (Frame no. 80). The motion is more visible in the car on the left lane with the speed of the car higher than those on the right lane due to traffic on the right lane. Green lines are drawn for validated optical flows whereas red lines are drawn for invalidated optical flows. There are several checks done to validate optical flows all the while not being too strict to avoid deleting good optical flows.
Feature Tracks

Figure: Tracks drawn on previous frame (Frame no. 79). The yellow lines are drawn for tracks with only 1 optical flow and were currently extracted. The green tracks are valid tracks that are still being tracked. The blue tracks are valid tracks but were not tracked at this instant.

Delaunay triangulation

The features extracted on the previous image is tracked on the current image using KLT tracker. The tracked features will be used to create Delaunay triangles.
Figure: Delaunay triangulation DT(P) of P given points on a plane. The triangulation ensures that no point lies inside the circumcircle of any triangle in DT(P). Delaunay triangulation maximizes the minimum angle of all the angles of the triangles in the triangulation.

Figure: Delaunay triangulation of feature points obtained from tracked features in frame 86 overlayed on frame 86.
Figure: Optical flows in the frame pair (Frame No. 86 and 87) overlaid on frame 86
Figure: Some noisy optical flow still remains on the bottom left of the image.
Clustering by simple threshold on proximity and velocity analysis on triangle pair on image

Obtaining triangles on the current image

- The optical flow vectors are sorted to enable binary search to find the index of the tracked feature.
- The corresponding triangle on the current frame is constructed for the triangle in the previous frame using the vertices on the previous triangle as tail of the optical flows.
- Triangles consisting of inconsistent optical flows from previous triangle to current triangle will be removed in the set of triangles in the current frame.
Figure: (a, b): Considering the red triangle as previous triangle, blue triangle as the current triangle and the yellow lines as optical flows, the triangle pair has 1 stationary point. Invalidate the stationary point and the edge linking to it. Two vertices and the edge connecting them are validated in this case.

Figure: (a, b, c) Considering the red triangle as previous triangle, blue triangle as the current triangle and the yellow lines as optical flows, the triangle pair has 2 stationary points. Invalidate the stationary points and the edges linking to it. Only one vertex is validated in this case.
Figure: (a,b) Given the red triangle as previous triangle, blue triangle as the current triangle and the yellow lines as optical flows, the magnitude and direction of optical flows connected to the 3 vertices of the triangle-pair.
Figure: Triangle pair with notations used to specify criteria for invalidating vertices and edges. The blue triangle $P_1'P_2'P_3'$ represents the Delaunay triangle for previous frame. The optical flows $P_1P_1'$, $P_2P_2'$ and $P_3P_3'$ are used to obtain the green triangle $P_1P_2P_3$. The blue and green triangle constitute the triangle pair for the given 3 optical flows.
Analyzing triangle pair

Currently, the following criteria is considered invalidating vertices and edges of a triangle pair:

- **Edge length** threshold: $T_{\text{edge\_length}} = 50\text{px}$
  - if $(P_1P_2 > T_{\text{edge\_length}})$ then invalidateEdge($P_1P_2$)
  - if $(P_2P_3 > T_{\text{edge\_length}})$ then invalidateEdge($P_2P_3$)
  - if $(P_3P_1 > T_{\text{edge\_length}})$ then invalidateEdge($P_3P_1$)

- **OF magnitude** threshold: $T_{\text{OF\_mag}} = 50\text{px}$ (NOTE: Using velocity should be better)
  - if $(L_1 > T_{\text{OF\_mag}})$ then invalidateVertex($P_1$); invalidateVertex($P_1'$)
  - if $(L_2 > T_{\text{OF\_mag}})$ then invalidateVertex($P_2$); invalidateVertex($P_2'$)
  - if $(L_3 > T_{\text{OF\_mag}})$ then invalidateVertex($P_3$); invalidateVertex($P_3'$)

- **OF magnitude difference ratio**: $T_{\text{OF\_rel\_mag}} = 0.2$
  - if ( $|L_1 - L_2| / L_1 > T_{\text{OF\_rel\_mag}}$ ) $||$ $|L_1 - L_3| / L_1 > T_{\text{OF\_rel\_mag}}$ )
    - then invalidateVertex($P_1$); invalidateVertex($P_1'$)
  - if ( $|L_2 - L_3| / L_2 > T_{\text{OF\_rel\_mag}}$ ) $||$ $|L_2 - L_1| / L_2 > T_{\text{OF\_rel\_mag}}$ )
    - then invalidateVertex($P_2$); invalidateVertex($P_2'$)
  - if ( $|L_3 - L_1| / L_3 > T_{\text{OF\_rel\_mag}}$ ) $||$ $|L_3 - L_2| / L_3 > T_{\text{OF\_rel\_mag}}$ )
    - then invalidateVertex($P_3$); invalidateVertex($P_3'$)

- **OF slope difference**: $T_{\text{deg}} = 15\text{ degrees}$
  - if $(|s_1 - s_2| > \text{thr} \&\& |s_1 - s_3| > T_{\text{deg}})$ then invalidateVertex($P_1$); invalidateVertex($P_1'$)
  - if $(|s_2 - s_1| > \text{thr} \&\& |s_2 - s_3| > T_{\text{deg}})$ then invalidateVertex($P_2$); invalidateVertex($P_2'$)
  - if $(|s_3 - s_1| > \text{thr} \&\& |s_3 - s_2| > T_{\text{deg}})$ then invalidateVertex($P_3$); invalidateVertex($P_3'$)

For a given triangle, Invalidating one vertex invalidates the two edges connected to it. Likewise, invalidating an edge invalidates the two vertices that the edge links.
Figure: Edges/Vertices of the obtained triangles in current frame invalidated using various criteria.
Figure: Validated triangles for Frame 87. Here, since the edges are invalidated due to large distance specified in pixels, the edges near the bottom of the image could be wrongly invalidated whereas those at the top could result in under clustering. This necessitates the use of homography to consider the feature points and the Delaunay triangulation on the ground plane.
Figure: Validated triangles overlaid on the current image (Frame no. 87)
Figure: Blobs extracted from grouping of triangles.
Example clustering results

Frame No. 150

Frame No. 152

Frame No. 154
Frame No. 200
Clustering by simple threshold on proximity and velocity analysis on triangle pair on the ground plane.

In order to improve the results of clustering, it is essential to consider the features on the ground plane while using a similar approach of simple threshold on proximity and velocity analysis on the triangle pair but on the ground plane. The assumption in both of these cases is that all optical flows are on the ground.

Figure: Red features are extracted from the previous frame (Frame No. 86, displayed). Blue features are the ones included from the existing tracks.
Figure: Optical flow from frame 86 to frame 87.

Figure: Delaunay triangulation using features defined in image coordinates.
Figure: (a, b) Optical flows and tracks projected onto the ground respectively. (c) Delaunay triangulation obtained from the head of optical flow tracks and displayed on the ground.
plane. (d) The Delaunay triangles are used to obtain the displayed set of triangles using optical flow in the current frame. The triangles along the border which are not valid with respect to the input data points are filtered out. (e) The triangle pairs (previous triangles in yellow and current triangles in green) displayed on the ground plane. The results are displayed for Frame no. 87. The displayed ground plane image is 200px x 1481px and the region in the ground plane (fMinX: -0.180152m fMinY: -5.28582m fMaxX: 6.45401m fMaxY: 44.065m) is 6.634m x 49.35m.

Analyzing triangle pairs

The triangle pair consisting of one triangle on previous image obtained from input features and the other triangle on current image obtained from optical flows for the input features. The edges of the current triangles and the optical flows connecting the vertices of the two triangles in the triangle pair are analyzed for motion constraints.

Currently, the following criteria is considered invalidating vertices and edges of a triangle pair.

- **Edge length** threshold: \( T_{\text{edge length}} = 3\text{m} \)
  - if \(|P_1P_2| > T_{\text{edge length}}\) then invalidateEdge(\(P_1P_2\))
  - if \(|P_2P_3| > T_{\text{edge length}}\) then invalidateEdge(\(P_2P_3\))
  - if \(|P_3P_1| > T_{\text{edge length}}\) then invalidateEdge(\(P_3P_1\))

- **OF speed** threshold: \( T_{\text{OF speed}} = 35\text{mps} \equiv 126\text{kph} \)
  - if \(|L_1| > T_{\text{OF speed}}\) then invalidateVertex(\(P_1\)); invalidateVertex(\(P_1'\))
  - if \(|L_2| > T_{\text{OF speed}}\) then invalidateVertex(\(P_2\)); invalidateVertex(\(P_2'\))
  - if \(|L_3| > T_{\text{OF speed}}\) then invalidateVertex(\(P_3\)); invalidateVertex(\(P_3'\))

- **OF magnitude difference ratio**: \( T_{\text{OF rel mag}} = 0.2 \)
  - if \(|L_1 - L_2| / L_1 > T_{\text{OF rel mag}}|| |L_1 - L_3| / L_1 > T_{\text{OF rel mag}}\)
    - then invalidateVertex(\(P_1\)); invalidateVertex(\(P_1'\))
  - if \(|L_2 - L_1| / L_2 > T_{\text{OF rel mag}}|| |L_2 - L_3| / L_2 > T_{\text{OF rel mag}}\)
    - then invalidateVertex(\(P_2\)); invalidateVertex(\(P_2'\))
  - if \(|L_3 - L_1| / L_3 > T_{\text{OF rel mag}}|| |L_3 - L_2| / L_3 > T_{\text{OF rel mag}}\)
    - then invalidateVertex(\(P_3\)); invalidateVertex(\(P_3'\))

- **OF slope difference**: \( T_{\text{deg}} = 15\text{degrees} \)
  - If \(|s_1-s_2| > \text{thr} && |s_1-s_3| > T_{\text{deg}}\) then invalidateVertex(\(P_1\)); invalidateVertex(\(P_1'\))
  - If \(|s_2-s_1| > \text{thr} && |s_2-s_3| > T_{\text{deg}}\) then invalidateVertex(\(P_2\)); invalidateVertex(\(P_2'\))
  - If \(|s_3-s_1| > \text{thr} && |s_3-s_2| > T_{\text{deg}}\) then invalidateVertex(\(P_3\)); invalidateVertex(\(P_3'\))
Figure: (a) Delaunay triangles obtained using features from frame 86. (b) Triangles obtained using optical flows from Frame 87. (c) Invalidating vertices and edges of the triangle based on simple thresholds of proximity and velocity along with some more criteria detailed below. (d) Surviving triangles and the clusters after enforcing constraints on vertices and edges. (e) Thick white border over the 3 clusters. The results are better than the case when considering features in image space only as shown in the figure below where 4 clusters are extracted.

Figure: Over-clustering (2 clusters for a single object at the bottom-right of the image) and under-clustering (1 cluster for 3 objects at the top-right of the image) for Frame no. 87. The use of features on the ground has improved the clustering of features near the bottom of the image.
Overview image to compare results between image based and ground plane based approach
Frame No. 191
Clustering by threshold on 3D motion consistency runs on Sutthisan video

The assumption that the optical flows are on the ground is obviously not true for all optical flows and the height of the optical flow will depend on the location of the feature on the object that is being tracked. This further requires estimating the height of the optical flow using common motion constraint and known camera calibration.

Considering minimum height of the optical flows to be 0.0m (on the ground) and the maximum height to be 3.0m, the information from the optical flow tracks can be used to improve the clustering. It is essential to have good estimates of homography and focal length to estimate camera extrinsics if it is not acquired directly by the calibration process.
System specification

Programming language: C/C++
Image processing library: OpenCV 2.4.9, libVGL 2.0 (April 2016)
Operating system: Ubuntu 14.04 LTS
Development environment: Eclipse IDE 3.8.1