Graph-Based Video Segmentation Using Optical Flow Fields

Bernard H. Lampe

Abstract. The ubiquity of video cameras has lead to an unprecedented amount of temporal information. Most of which is trivial or redundant and can be ignored. A key step in reducing data volumes is to identify portions of the video in which change is occurring while ignoring those areas which remain static. Purposed in this work is a novel algorithm for identifying and segmenting video frames into areas of motion through the analysis of optical flow fields. The algorithm is a three stage process. The first stage utilizes a decoded video stream to estimate the optical flow field with a hierarchical version of the Lucas-Kanade algorithm[3]. Then coherent, inter-frame integration of the optical flow field is performed to reduce the effects of camera and quantization noise. Finally, the magnitudes and positions of the optical flow estimates are used in a graph-based segmentation scheme that aims to create image segments that are uniform with respect to motion[1]. Finally, experimental results are presented on a traffic scene obtained from the web. Complete C++ code of the implementation accompanies this document.

1. Introduction

Given the ubiquity of modern video capturing technology it becomes apparent and increasingly important to develop methods to automatically understand dynamic scenes. Automatic methods of video understanding can reduce the need for costly human reviewers while simultaneously reducing the amount of redundant and trivial data inherent with high sampling rates. Humans are particularly adept to noticing and identifying moving objects in dynamic scenes. Figures 1, 2 and 3 illustrate a traffic scene whereby a motorcyclist is moving in a static traffic scene. The motion of the motorcyclist becomes immediately apparent to anyone observing the video stream. The ability to automatically identify areas of change and segment those areas into logic regions is a crucial step towards scene understanding and data reduction. The novel approach presented here aims to identify and segment moving objects in the scene via an analysis of dense, optical flow field estimates.

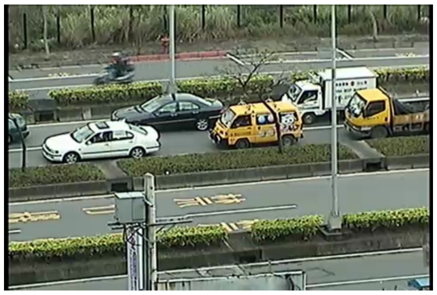


Figure 1. Traffic Scene at t = 0 s.

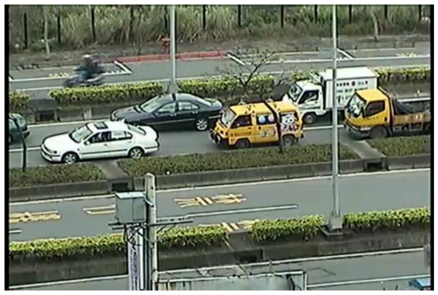


Figure 2. Traffic Scene at t = 1 s.

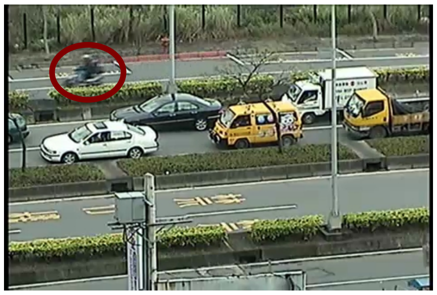


Figure 3. Traffic Scene at t = 1 s. Moving Object Detected.

2. Prior Work in Scene Segmentation

Scene segmentation is a well studied area of computer vision with a large body of documentation over the last 60 years. Segmentation techniques such as k-means, mean shift, graph cutting and graph merging are just a subset of well-known algorithms that have been used successfully in the past. Each technique maps attributes of the image into a feature space and then iteratively decides what constitutes a good segmentation of the feature space via the evaluation of a similarity predicate. Commonly, the motivation behind those predicates is to create regions which minimize the internal distance of the evaluated attribute while simultaneously maximizing the distance between different regions. Image attributes such as brightness, edge strength, color and texture have been the focus of single image segmentation paradigms for years[4][5]. With the advent of readily available video data, temporal features, such as motion, have been added to the set of image attributes suitable for region segmentation[6]. This has lead to a plethora of techniques collectively know as motion segmentation.

The challenges presented by temporal data segmentation are different than those encountered when segmenting a single image. In order to take advantage of temporal scene changes it is necessary to discount or threshold irrelevant changes which arise due to camera noise, illumination changes due to participating atmospheric media and quantization noise. Quantization noise is prominent in video due to lossy encoding which is often necessary when compressing video streams to a reasonable data size. To address the stochastic nature of video streams, analysis paradigms often fit high-order models to the each scene region and then compute a probability of change. Other techniques involve adapting robust segmentation algorithms which will account for such noise. These techniques are typically very computationally expensive and require special hardware in order to achieve real-time performance.

This work purposes a simple technique to reduce the effect of noise through coherent, inter-frame optical flow field integration. This approach aims to improve the performance of the segmentation algorithm without impacting run-time performance. The simplicity of this technique permits real-time system implementation on modern, commodity hardware. In addition, complete C++ code accompanies this work which demonstrates the implementation.

3.1 Technical Overview

The purposed system is illustrated in figure 4 and is summarized in four discrete processing stages.

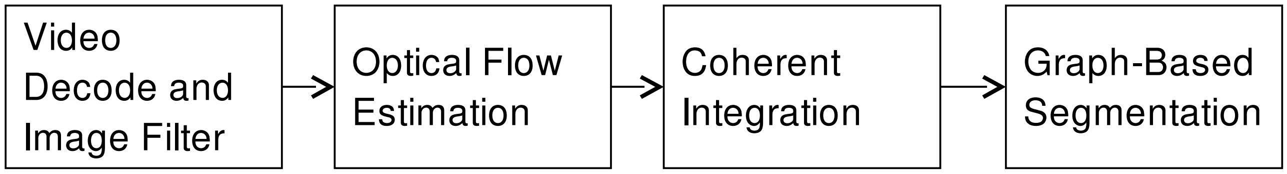


Figure 4. Algorithm Overview.

Step one decodes the video stream using a third party software package and a library of video codecs provided by FFMPEG. The raw images are individually filtered with a user-defined Gaussian kernel immediately after decoding to remove any camera noise and improve the quality of the gradient calculations which assume the signal is differential. The next step of the algorithm computes a 2-D optical flow vector field using motion between pairs of frames. A hierarchical version of the Lucas-Kanade algorithm is used here because it provides robustness to the aperture problem. The third step in the algorithm is to aggregate optical flow measurements across frames using coherent integration. This suppresses errors in the optical flow field due to quantization noise. Finally, the magnitudes of the integrated optical flow vector fields are used in a graph-based image segmentation algorithm[1]. The segmentation algorithm is a bottom up algorithm that merges pixels together with similar magnitudes of integrated optical flow.

3.2. Technical Details

3.2.1. Video Decoding and Filtering

The first processing step of the algorithm is to decode and filter the images. Decoding the images was accomplished using FFMPEG. FFMPEG is a complete, cross-platform software package that encodes and decodes audio and video streams. It is based on a software library named libavcodec which contains most of the common video codecs. This enables the program to work with many different types of video encodings. Immediately after the frames are decoded, they are converted to gray scale. The gray scale conversion replaces each pixel with the Euclidean magnitude of each RGB color vector. This magnitude is proportional to pixel brightness. The final step in the first processing stage is to perform a 2-D convolution with a 1-D Gaussian function. The 2-D Gaussian function is isotropic about the center of the signal. Therefore, a 2-D convolution can be accomplished with a pair of 1-D convolutions in both dimensions. Discrete estimation of the image derivatives is an important first step in estimating optical flow and the image filtering improves the estimation process by ensuring the signal is differentiable at all pixel locations.

3.2.2. Optical Flow Estimation

The second processing step of the algorithm is to estimate the 2-D optical flow vector field using pairs of video frames. The Hierarchical Lucas-Kanade (HLK) algorithm was selected to perform the estimation. Other optical flow estimation algorithms were evaluated but deemed to be too slow or sensitive to noise[2][3]. In order to estimate optical flow at each location in the image, a number of assumptions need to be made about the ability to associate image locales between video frames. The assumptions necessary for optical flow estimation are:

Assumption #1. The motion between the frame is less than the user-defined neighborhood size.
Assumption #2. The brightness of the image pixels is preserved at each pixel between frames.

Under these crucial assumptions it is permissible to perform the following Taylor expansion of the images as in Equations 1, 2, 3 and 4. This defines the first linear constraint denoted in Equation 5. Where *I* is the 2-D image function, *(x,y)* is a 2-D image coordinate in the natural image basis and *t* is the frame index relative to the first frame. *H.O.T.* denotes Higher Order Terms which are ignored in subsequent steps of the derivation.

$$I(x,y,t)=I(x+\delta x, y+\delta y, t+\delta t) \tag{1}$$

$$I(x+\partial x, y+\partial y, t+\partial t)=I(x,y,t)+\frac{\partial I}{\partial x}\delta x+\frac{\partial I}{\partial y}\delta y+\frac{\partial I}{\partial t}\delta t+H.O.T. \tag{2}$$

$$\frac{\partial I}{\partial x}\delta x+\frac{\partial I}{\partial y}\delta y+\frac{\partial I}{\partial t}\delta t=0 \qquad (3)$$

$$\frac{\partial I}{\partial x}V_x+\frac{\partial I}{\partial y}V_y+\frac{\partial I}{\partial t}=0 \qquad (4)$$

Equations 1 – 4. Taylor Expansion of 2-D Image Function.

$$I_x V_x+I_y V_y=-I_t \qquad (5)$$

Equations 5. Linear Constraint Equation Derived From Taylor Expansion.

The single linear constraint of equation 5 is not sufficient to uniquely solve for optical flow. Therefore, the Lucas-Kanade algorithm makes another assumption which gives rise to a system of linear constraints that can be solved in a least squares sense.

Assumption #3. The optical flow in any local image region is uniform.

With this assumption a linear system can be constructed by applying the Taylor expansion from Equation 5 to individual pixels in any local region of the image. Linear systems 1 and 2 illustrate the construction.

$$\begin{array}{c} I_{x_1}V_x+I_{y_1}V_y=-I_{t_1}\\ I_{x_2}V_x+I_{y_2}V_y=-I_{t_2}\\ I_{x_3}V_x+I_{y_3}V_y=-I_{t_3}\\ \ldots\ldots\\ I_{x_n}V_n+I_{y_n}V_y=-I_{t_n} \end{array} \qquad (1)$$

$$A\vec{v}=-\vec{b} \qquad (2)$$

System 1 and 2. Linear System Constructed from 3rd Assumption (numerical subscripts denote different pixels in any local neighborhood).

The solution to this system can be derived as in Systems 3, 4 and 5.

$$A^T A\vec{v}=A^T(-b) \qquad (3)$$

$$\vec{v}=(A^T A)^{-1}A^T(-\vec{b}) \qquad (4)$$

$$\begin{bmatrix} V_x\\ V_y \end{bmatrix}=\begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i}I_{y_i}\\ \sum I_{x_i}I_{y_i} & \sum I_{y_i}^2 \end{bmatrix}^{-1}\begin{bmatrix} -\sum I_{x_i}I_{t_i}\\ -\sum I_{y_i}I_{t_i} \end{bmatrix} \qquad (5)$$

Systems 3 – 5. Derivation of Least Squares Solution to Linear System 2.

Armed with the solution in System 5. The optical flow can be estimated at each image location using the Lucas-Kanade algorithm by constructing the second moment matrix and computing the inverse. This algorithm does suffer from the aperture problem which can cause the system above to be degenerate. If the inverse of the matrix A does not exist, the optical flow cannot be estimated. To overcome this limitation caused by the aperture problem the algorithm is executed in a coarse to fine manner using a constant window size.

The HLK algorithm computes the Gaussian pyramid of images and then estimates the optical flow using the lowest resolution image. These estimates are utilized as an initial estimate when computing the optical flow in the next, higher resolution level. Also, the images are progressively warped as the resolution increases. This overcomes the limited aperture because the algorithm is running with a progressively smaller aperture when traversing the pyramid. Figures 5 and 6 illustrate the hierarchical approach to Lucas-Kanade.

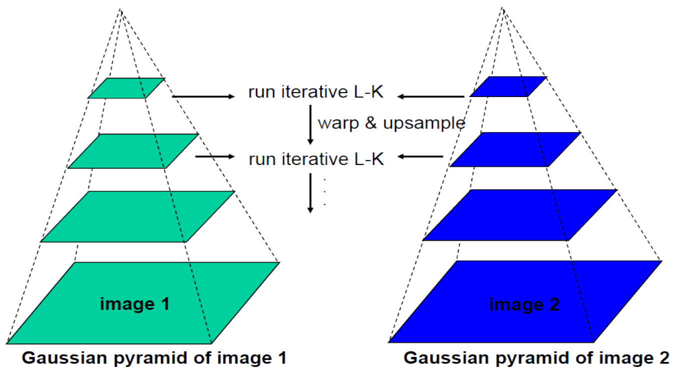
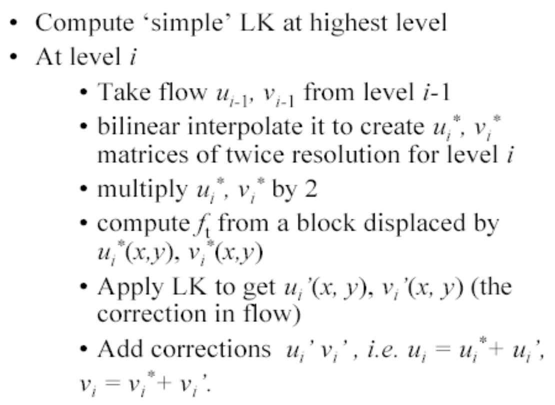


Figure 5. Illustration of HLK Algorithm.



- Compute 'simple' LK at highest level
- At level i
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution for level i
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.

Figure 6. Outline of HLK Algorithm.

Testing the optical flow estimation using the classic spinning ball example is shown in Figure 6. Executing the HLK algorithm on a pair of image frames from the traffic scene above results in the optical flow field depicted in Figure 7. Notice the errors in the estimates in the background due to quantization noise. In addition, the optical flow estimates of the motorcyclist seem noisy. This is due to motion blur which varies between frames.

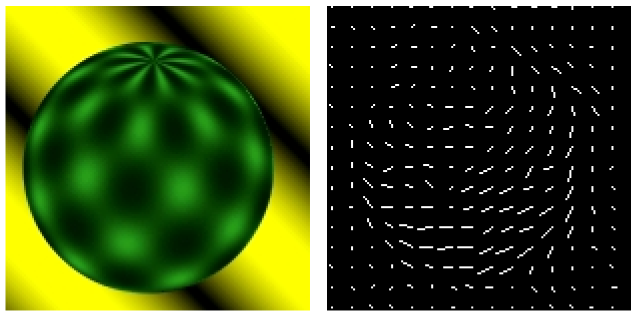


Figure 6. Test of HLK using Classic Spinning Ball Example.

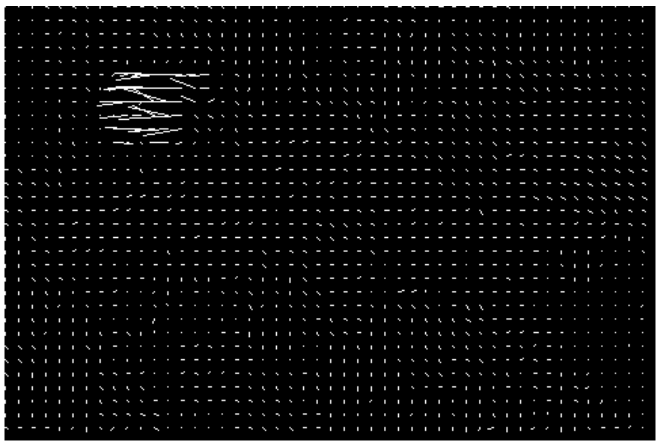


Figure 7. Optical Flow Estimate of Traffic Scene using HLK.

3.2.3. Coherent Integration

The third processing step in the algorithm is to coherently integrate the optical flow estimates between frames. The introduction of this processing in the algorithm sequence improves the quality of the segmentation in the final processing step by reducing the error in the optical flow field estimates. The assumption made by this algorithm is stated as follows.

Assumption #4. Quantization noise between image frames is not correlated with time.

Under this assumption it is safe to perform the integration. The user defines the number of frames on which to integrate. As the number of integrated frames becomes large the ability to distinguish small motion in the scene diminishes. Conversely, if the number of frames is too small, this algorithm will be less effective. Figures 8 and 9 demonstrate the notional effect of the algorithm on the optical flow vector field. Note that persistent motion across the frames leads to cooperative vectors that tend in the direction of motion, while vectors without contributing motion tend towards zero. This destructive reinforcement is a result of the lack of noise correlation in areas without motion.

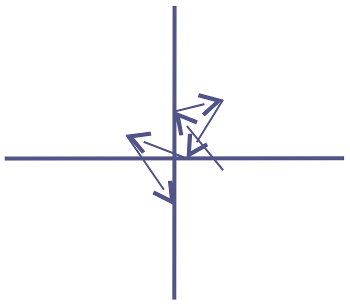


Figure 8. Destructive Reinforcement of Optical Flow Estimates.

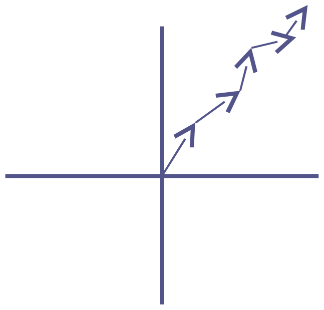


Figure 9. Constructive Reinforcement of Optical Flow Estimates Due to Coherent Motion.

Figures 10 and 11 demonstrate the reduction of noise in the optical flow vector field as a result of coherent, inter-frame integration. Notice the lack of false positive estimates in the background areas of the scene when compared to Figure 7.

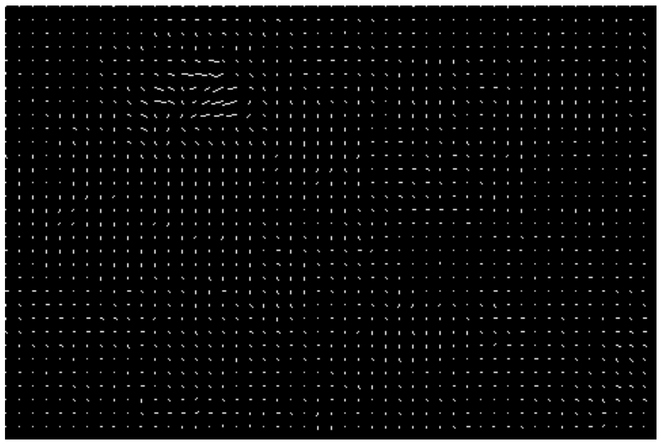


Figure 10. 3 Frame Integration of Optical Flow.

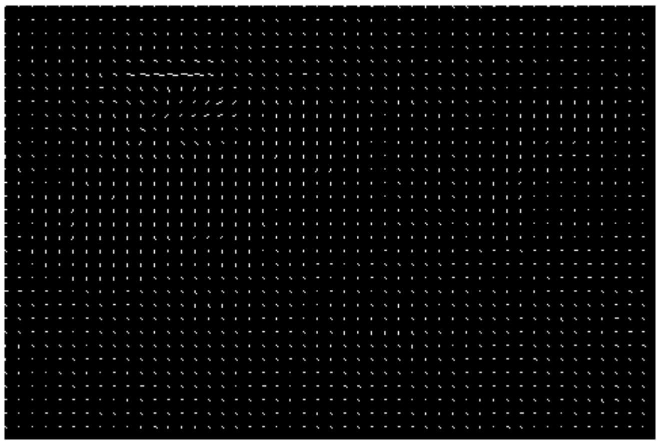


Figure 11. 10 Frame Integration of Optical Flow.

3.2.4. Graph-Based Segmentation

The final step in processing is to segment the graph into coherent regions of motion using the magnitude of the optical flow estimates. The algorithm developed by Pedro F. Felzenszwalb and Daniel P. Huttenlocher detailed in their paper titled *"Efficient Graph-Based Image Segmentation"* was utilized[1]. This algorithm was selected due to its ability to run in real-time and segment arbitrary image attributes.

The algorithm proceeds by constructing the complete graph of the image. Each pixel of the image becomes a node in the graph and is 8-connected to each neighbor. Edge weights are computed using the Euclidean distance between the pixel weights. For our purposes, the pixel weights are the magnitudes of the integrated optical flow estimates. After construction, the graph is traversed along the minimum spanning tree by iterating over the edges from lowest to highest weight. Merging of pixels into coherent groups is performed by comparing the internal distance to the external distance between sets of pixels. The internal distance of a set of pixels is defined to the be maximum distance between any two pixels in the set. The external distance between two sets of pixels is defined to be the minimum distance between any two pixels between the sets. If the internal distance greater than the external distance, then the two sets of pixels are merged. The intuition motivating this algorithm is to minimize the internal variance of the sets of pixels while maximizing the external variance between adjacent sets of pixels. Variance in this case is estimated as the range of the distribution of pixels internal and external to the sets.

4. Experimental Results

Results of the algorithm on the traffic scene data is shown in Figures 12, 13, 14 and 15. The false colors have no meaning themselves, but differentiate between various regions. During this time frame, only the motorcyclist was moving in the scene. Another demonstration of the algorithm is given in Figures 14 and 15 where other, larger vehicles in the scene were moving.

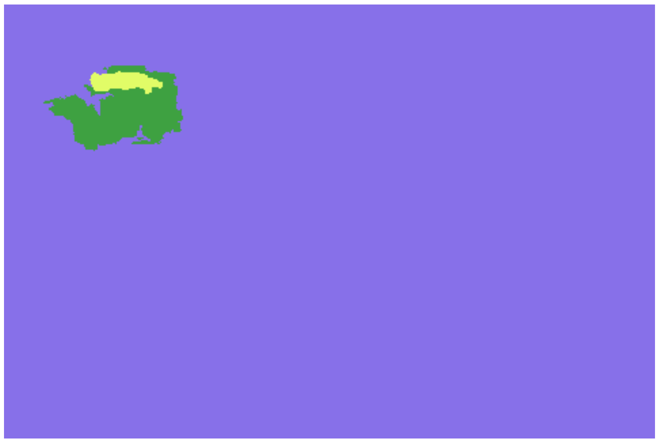


Figure 12. Segmentation Result of Motorcyclist.

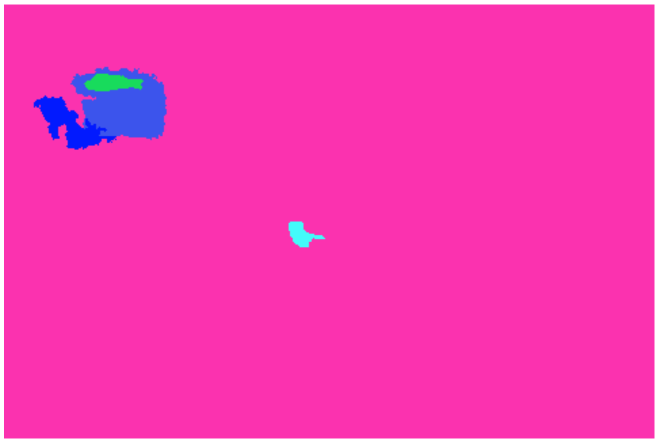


Figure 13. Segmentation Result of Motorcyclist + 2 Frames.

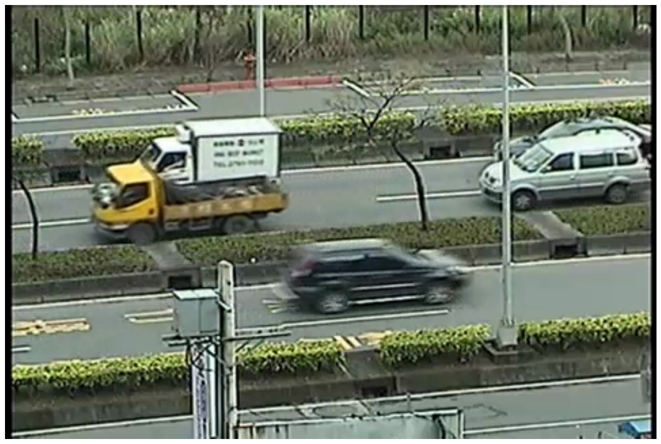


Figure 14. Later Traffic Scene with Larger Moving Vehicles.

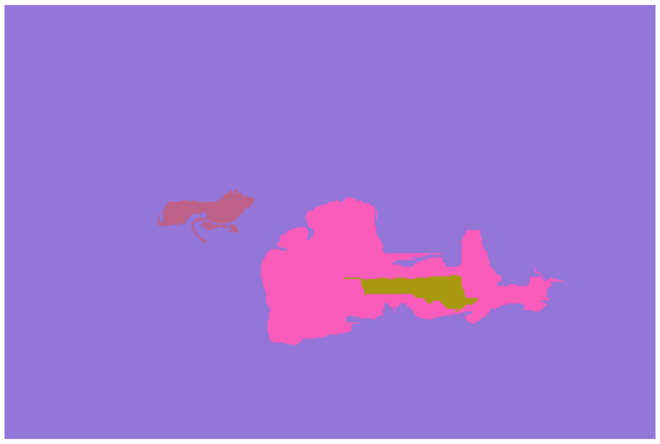


Figure 15. Segmentation Result of Moving Vehicles in Figure 14.

5. Conclusions

This work has demonstrated the utility of using motion attributes in the form of optical flow estimates when finding regions of change in dynamic video sequences. Motion attributes can form the basis of video segmentation algorithms. Such algorithms are an ever increasing important step towards automated, dynamic scene understanding systems. Algorithms such as these can be utilized in real-time and can be very useful in reducing the amount of trivial and redundant data inherent in high sampling rate video systems.

This work has also highlighted some of the important challenges when working with video data. Quantization noise is prolific due to lossy video compression. Methods to counter the effects of this noise need to be part of the initial design of automated scene understanding. The algorithm which resulted from this work has explored a novel, real-time, noise suppression method. In addition, it was found that single level optical flow estimation algorithms are particularly sensitive to quantization noise in video sequences. The iterative algorithm purposed by Horn and Schunck and the single level Lucas-Kanade algorithm are unsuitable for levels of noise found in videos readily available via website such as youtube.com.

6. References

1. Felzenszwalb & Huttenlocher, *Efficient Graph-Based Image Segmentation*, International Journal of Computer Vision, vol 59, 2004.
2. Horn & Schunck, *Determining Optical Flow,* Artificial Intelligence, vol 17, pp 185-203, 1981.
3. Lucas & Kanade, *An iterative Image Registration Technique with an Application to Stereo Vision*. Proceedings of Imaging Understanding Workshop, pp 121 – 130, 1981.
4. Malik, Belongie, Leung & Shi, *Contour and Texture Analysis for Image Segmentation,* International Journal of Computer Vision, vol 43(1), pp 7-27, 2001.
5. Shi & Malik, *Normalized Cuts and Image Segmentation***,** IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 22 (8), 2000.
6. Wong & Spetsakis, *Motion Segmentation and Tracking*, Dept. of Computer Science, York University, 2000.