

Motion Detection and Representing Moving Objects from Video

{hari, narnindi, kashimaro@spatial.maine.edu}

Introduction

Advances in image storage and processing techniques have made it possible to record videos at better spatial, spectral and temporal resolution. Spatial resolution deals with the clarity of the image, while spectral resolution with gray level intensity and temporal resolution on the smooth continuity of video. Video data sets capture the motion dynamics between frames that could be used to track moving objects. Motion detection has found applications in video surveillance, traffic monitoring and video compression. Object's trajectory information is particularly helpful in temporal compression of the video [1]. In this paper we address the challenging issues on motion detection and representing moving objects in a 3D environment. We interpret motion as change in the position of an object between frames. The paper also discusses an optical flow object-tracking algorithm to track objects moving with constant velocity.

The rest of the paper is organized as follows. Section 2 discusses the methodology of trajectory extraction with experimental setup, implementation aspects and representing the trajectories in a 3D environment. Section 3 presents the analysis and results of our experiments and Section 4 proposes future work.

2. Trajectory Extraction Methodology

2.1 Experimental Setup

The video was made to reflect a typical scenario in a parking lot. In our case, we recorded two scenarios one with cars moving in and out and the other with cars and people moving out of the parking lot. The scenes were recorded from top, side and front views using three cameras. The cameras were placed still during the whole exercise. The setup also assumes uniform illumination throughout the recording. All the scenes were recorded in 24-bit color at 30 frames per second with 720 X 480 spatial resolution. The experiments were done on a Pentium 4 1.6 GHz processor with 256MB RAM in MATLAB 6.5 environment. Quick views of what the videos were like are provided in figure 1. All through the paper we will refer these video scenarios as Video1 to be the 2 cars moving in the opposite direction; and Video2 to be the one consisting of 3 humans and a car moving along in the same direction.

2.2 Video pre-processing

Videos captured in our setup provide a great level of spatial and temporal redundancy. The cars in the scene were moving on an average of around 10 miles per hour and humans with an average of 5 miles per hour. To track objects that are moving at

a fairly slow, the motion detected between two consecutive frames on a video captured at 30 frames per second is negligible.



Fig 1: Snap shots of the video scenarios

To represent moving objects in our scenario with a smooth trajectory, a sampling rate of 10 frames per second served our purpose. The reduction in sampling rate significantly reduces the computation time. We also observed that the objects in our scene could be well detected with a spatial resolution of 200 X 300. The down sampling of spatial resolution also resulted in reduction in the processing time, because of the point operations that we used for accumulative frame differencing.

As a first step in separating moving objects from the background an accumulating frame differencing technique, ACD [2], was used. The positive and negative ACD only captures either the leading edge or the trailing edge of the moving objects depending on the illumination difference of the objects with respect to the background. An absolute frame differencing approach captures both the leading and trailing edges of a moving object irrespective of the illumination difference. We had accumulated 5 frames and applied a threshold of 20 to detect motion between frames. The resultant frames had some salt and pepper noise that was cleaned up using a 3X3 median filter. Objects of interest in this experiment have a rigid shape in reality, but objects resulted out of ACD had poor or no connectivity in some cases, which could be attributed to the non-uniform illumination difference across the object. To establish a rigid shape for the objects through proper connectivity a morphological close operation with a rectangular structuring element was used.

A statistical operation on each frame, which gives the area and centroid of each object, was applied. The area parameter was used to filter out humans and cars based on size. This provided a means of separating trajectories of humans and cars. This pre-processing procedure filtered out most of the unwanted motion in the video.

3 Analysis and Results

3.1 Minimum Spanning Tree (MST)

The Minimum Spanning Tree (MST) algorithm has originally been used in the electrical and computer engineering / science disciplines in graph theory. The MST of a graph defines the cheapest subset of edges that keeps the graph in one connected component. Telephone companies are particularly interested in minimum spanning trees, because the minimum spanning tree of a set of sites defines the wiring scheme that connects the sites using as little wire as possible. Minimum spanning trees [3] prove important for several reasons:

- They can be computed quickly and easily, and they create a sparse sub-graph that reflects a lot about the original graph.
- They provide a way to identify clusters in sets of points. Deleting the long edges from a minimum spanning tree leaves connected components that define natural clusters in the data set.
- They can be used to give approximate solutions to hard problems such as Steiner tree and traveling salesman whose objective is to minimize the traveling distance.
- As an educational and research tool, minimum spanning tree algorithms provide graphic evidence.

We used the MST algorithm to connect the points into trajectories, based on threshold parameter defining proximity. The Minimum Spanning Tree Algorithm calculates the distance between each point in the trajectory and connects all those points if the distance between those points falls below a pre-specified threshold. For example, if an object is moving continuously at a constant rate then the centroids of them would be equidistant from each of them. The MST would connect all those adjacent points whose distance is less than the threshold. Our scenarios involved acceleration as well as deceleration in the motion of the cars and humans. Therefore the centroids of the points are not equidistant but do fall under the specified threshold thus resulting in, something similar to a closely connected graph. Thus using this approach, the points in the trajectory have been plotted in 3D and shown in figures 2 and 3 for both the videos respectively. MST is significant in tracking all possible motion into one trajectory.

In the ideal case, the top view of the videos gives the best results, but since we do not have the camera placed normal to the ground, we would get a better plot of the trajectory in the same plane of the video taken i.e. Y (col) vs. X (row). One important thing in following the plots is that the plots have been shown as Y vs. X plots where each Cartesian coordinate (x, y) point plotted in all the figures is ideally (-x, y) i.e. (row_id - number_of_rows, col_id) in the image coordinate system denoted as (row, col). The Y vs. X plots for video1 is shown in figure 4 and the same for video2 is shown in figures 5, 6 and 7 which includes plots of only humans and only car which are extracted.

The MST 3D plot shown in figure 2 clearly connects all the centroidal points of each car in a fairly good manner showing the two distinct trajectories. The break in one of the

trajectory and the additional points scattered around that frame attributes to one of the cars that has stopped and thus disappeared in the accumulated differenced movie through 6-7 frames. Even then the trajectory has been well connected through the rest of the frames. The MST 3D plot of the second video (figure 3), shows the trajectories of the humans and the car.

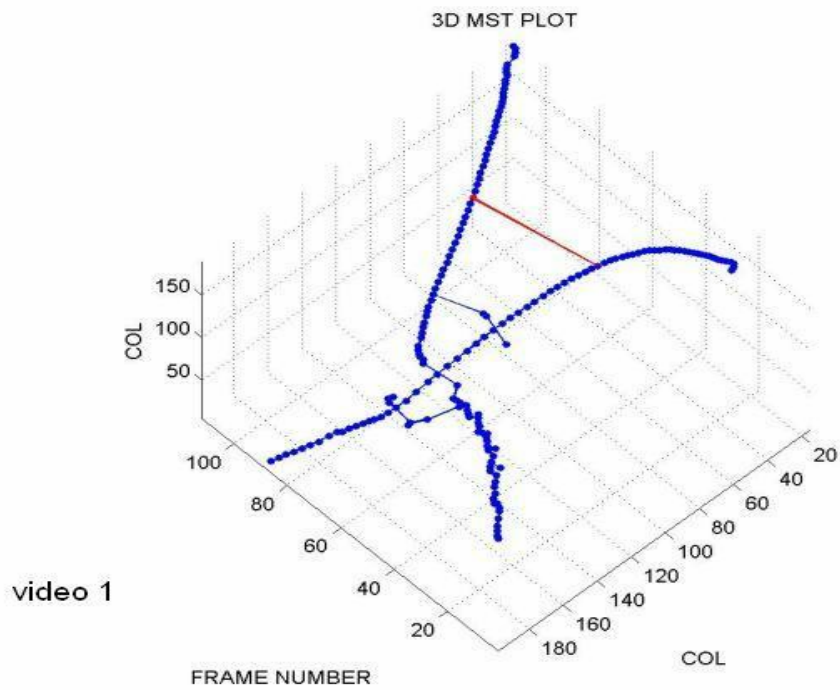


Figure2: Video-1 3D MST PLOT

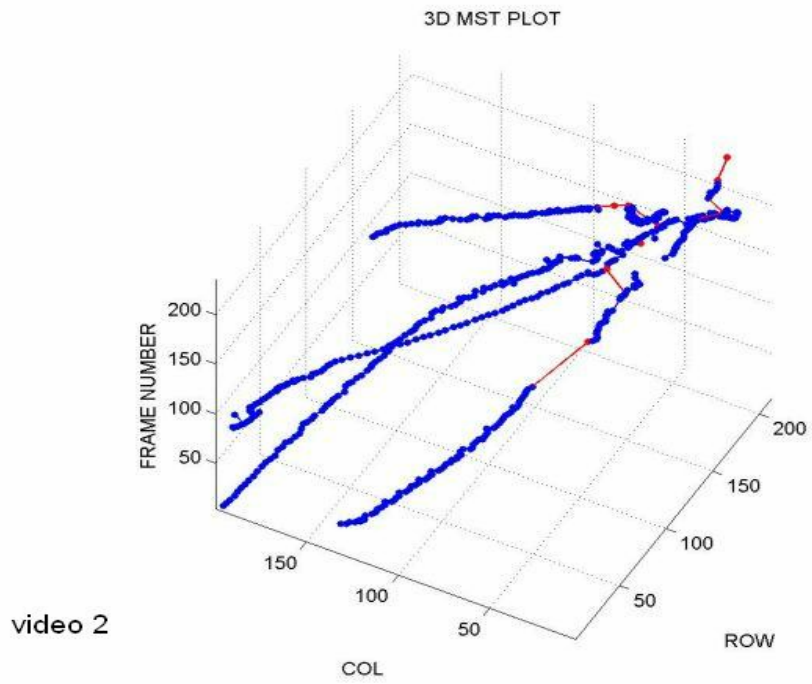


Figure 3: Video -2 3D MST PLOT

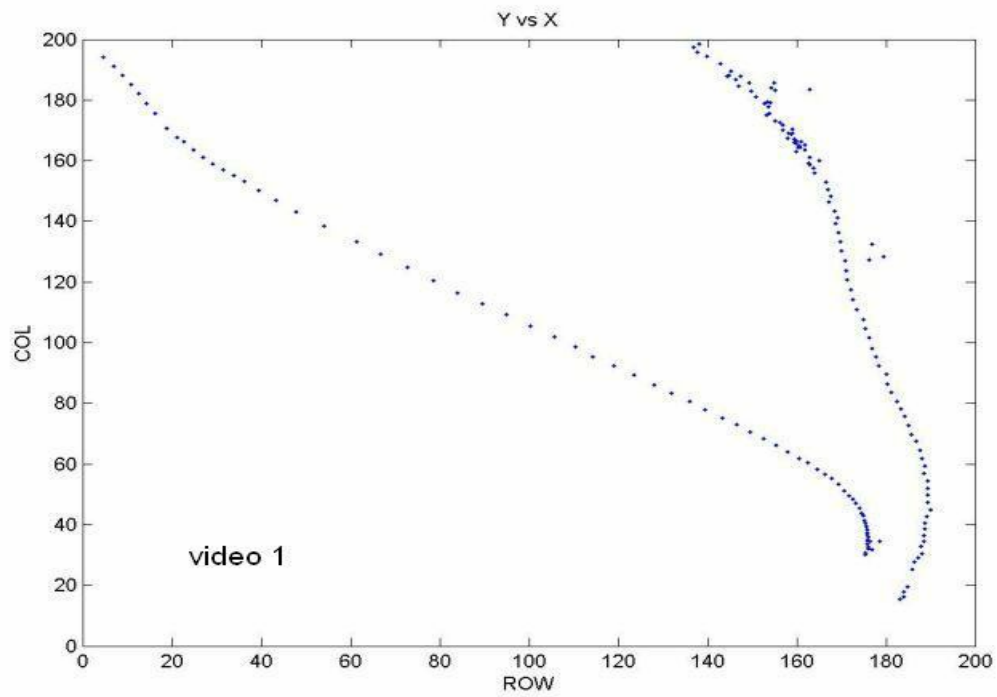


Figure 4: Video1 Y vs. X plot

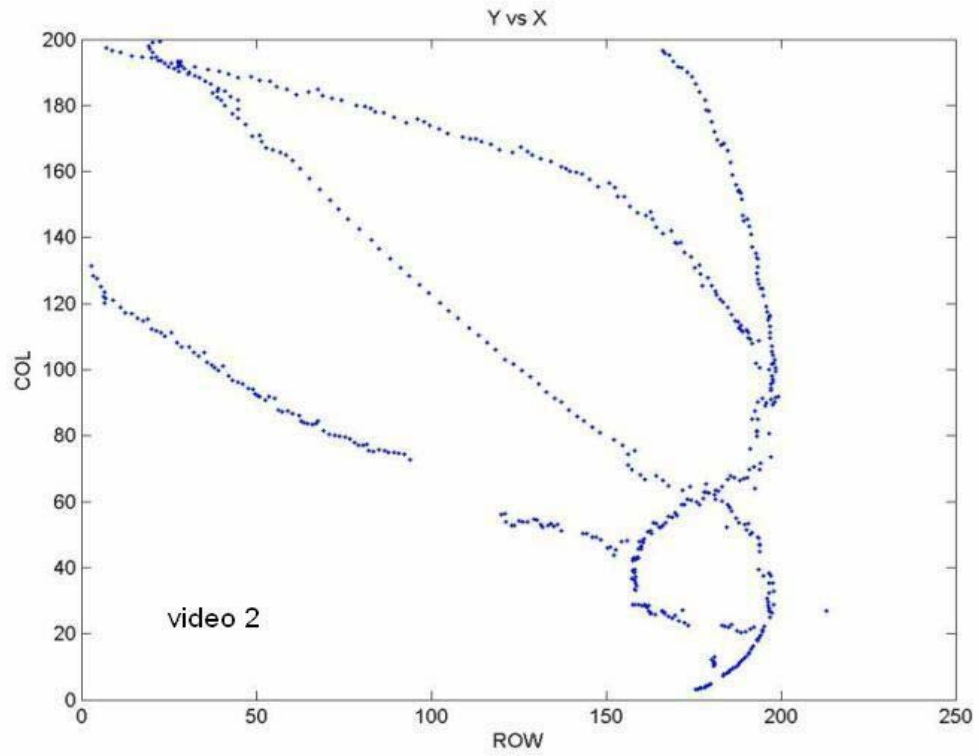


Figure 5: Video2 Y vs. X plot

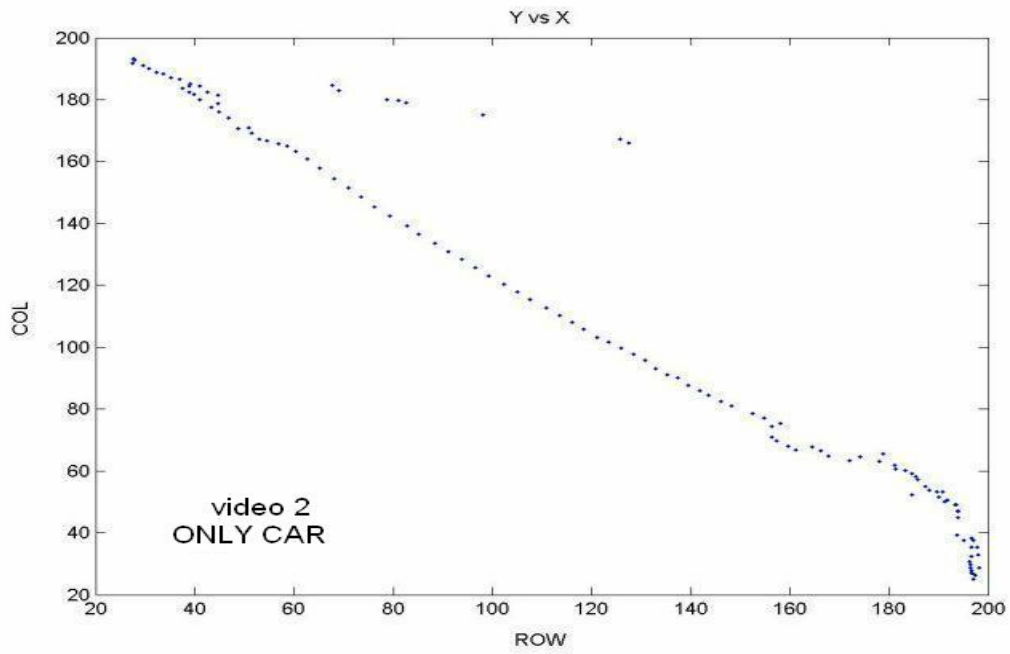


Figure 6: Video2 Y vs. X plot

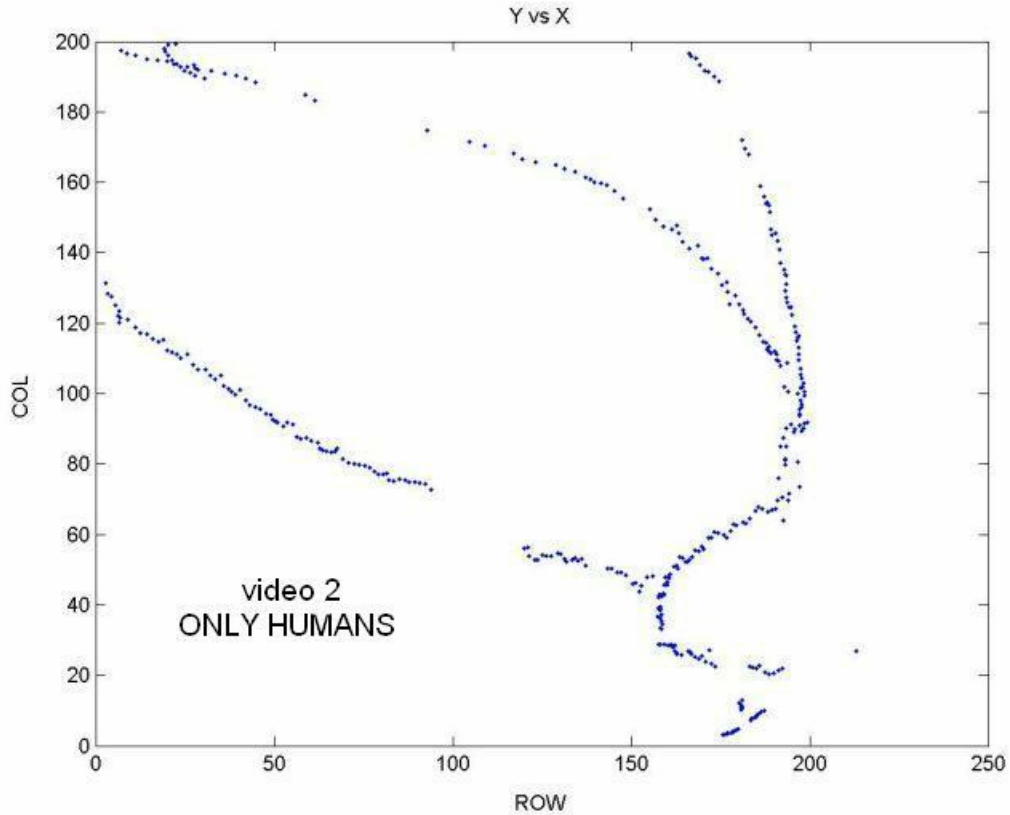


Figure 4: Video2 Y vs. X plot

3.3 Representing Trajectories

A 3D model of the scene was created using tape measurements of the building facades. Using MicroStation SE we created the design file using the actual measurements and raised the 2D top view to the height of the building thus making it as a 3D building. We choose a point at the end of one of the building corners to be the origin (0, 0, 0). In a similar way we chose the exact same point (figure 8) in the video frame to be the origin, so that the transformation of the trajectory points on to the 3D becomes easy and accurate. We rendered the 3D model with the images of the facades of the building taken before. Just to add a touch of the real parking lot we included parking lines as well.

The trajectories resulted out of minimum spanning tree algorithm, were in quasi 3D with image coordinates. A trajectory in this case is geometrically a polyline represented by a collection of image points that represent the centroid of the moving object with frame numbers. Better representation of trajectories would be a vector 3D model, so we used Microstation 3D design file to plot the trajectories in 3D environment. A projective transformation between image coordinate system and the 3D vector system was used. Figure 8 shows the control points that were used to map and arrive at the transformation parameters.



Fig 8: Control points in image

The video and the 3D model of the parking lot did not have too many points in common, that could be clearly distinguished in both the systems. The projective transformation used, preserves straight lines in both the systems, but doesn't necessarily maintain parallelism, so we chose control points that were far apart and clearly distinguishable in both systems.

Plotting the trajectory of the moving objects

Plotting the centroids of the moving objects on a 3D model is an effective tool to represent the trajectories. The centroids obtained from the statistical filtering approach (based on area) mentioned before were plotted on the rendered design file. The 3D model of the videos consisting of movements of a car and humans, and 2 cars is shown in figure 9 and figure 10 respectively. The trajectories were clearly marked up in these figures.

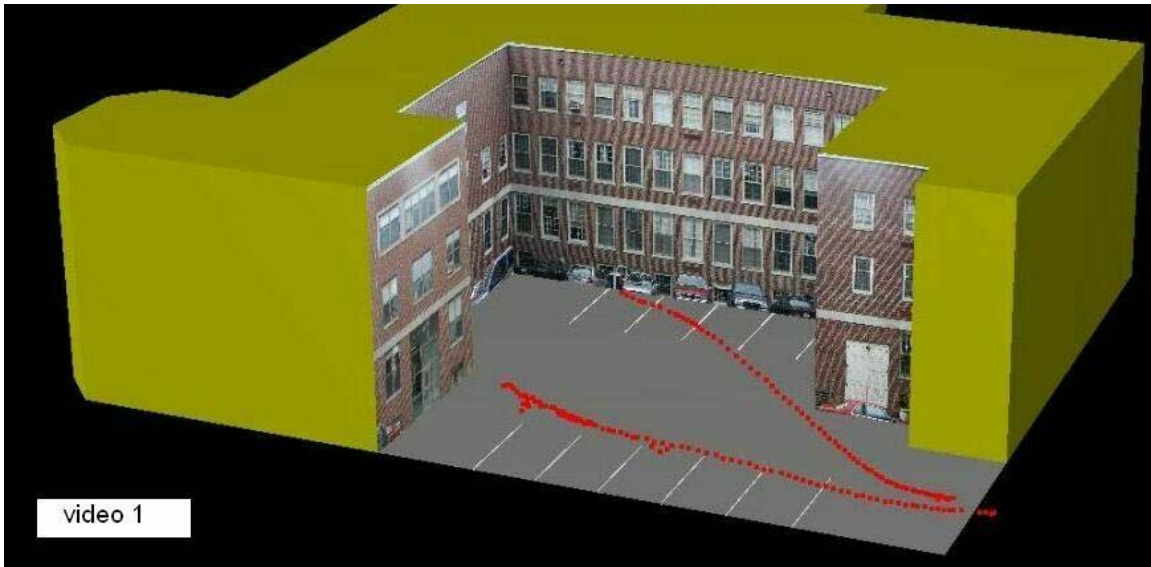


Figure 9: The 3D model showing the trajectories of 2 cars in Video 1



Figure 10: The 3D model showing the trajectories of 2 cars in Video 1

4. Future Work

As an extension to tracking objects using the MST, we propose an algorithm based on optical flow. Optical flow algorithm involves tracking an object based on velocity of the object in x and y direction.

Definitions:

Speed at which the object moves across frames is given by the first derivative of the distance between the centroids of the objects' position between frames. If (r_1, c_1, t_1) represents the object's position at frame t_1 and (r_2, c_2, t_2) represents the object's position at t_2 , then,

$$\text{Speed} = ((r_1 - r_2)^2 + (c_1 - c_2)^2)^{1/2} / (t_2 - t_1) \text{ pixels/frame}$$

Slope between object's positions with respect to the image's horizontal plane is given by,

$$\text{Slope} = ((r_1 - c_1) / (c_1 - c_2))$$

Optical flow object tracking involves the following steps:

1. Calculate the distance between objects' centroid in $(i-1)^{\text{th}}$ frame with every other objects' centroid in i^{th} frame and pick the object that has least distance as the next point in trajectory. This is analogous to extending minimum spanning tree temporally.
2. Calculate the slope between the two selected centroids in $(i-1)^{\text{th}}$ frame and i^{th} frame
3. Calculate the distance and slope between objects' centroid in i^{th} frame with every other objects' centroid in $(i+1)^{\text{th}}$ frame.
4. Calculate the slope difference between the slope calculated in step 2 and step 3.
5. Pick the object in $(i+1)^{\text{th}}$ frame that has least distance and slope difference between the centroids in i^{th} frame as third point in the trajectory.
6. Do steps 1 to 5 for every object in $(i-1)^{\text{th}}$ frame and continue this for all frames in the video to build the trajectories.

Optical flow object tracking is a greedy algorithm that chooses the local best in every step it proceeds and assumes the trajectories of moving objects are directionally consistent. The algorithm is simplistic in the sense, that it only tracks objects that move at a constant velocity. Objects while slowing down to stop (approaching zero velocity) or speeding up (starting from zero velocity) could not be tracked with this approach. Including the algorithm by including an acceleration parameter will further refine the approach to track objects that move at different speeds throughout the trajectory. The approach also assumes a constant camera and accumulative frame differencing can be incorporated by supporting the whole image to compensate for camera motion.

References:

- [1]. MPEG-4 Natural Video Coding - An overview, Touradj Ebrahimi and Caspar Horne, In *Signal Processing: Image Communication*, vol. **15**, pp. 365-385, Elsevier, 2000.
- [2]. Digital Image Processing (2nd Edition), Rafael C. Gonzalez and Richard E. Woods
- [3]. Minimum spanning tree, The Stony Brook Algorithm repository
<http://www.cs.sunysb.edu/~algorith/files/minimum-spanning-tree.shtml>