MULTIPLE TARGETS DETECTION AND TRACKING IN WIDE AREA SURVEILLANCE

Under the guidance of:

MR. ARSHAD JAMAL

Scientist- 'E'

Centre for Artificial Intelligence and Robotics

Defence Research & Development Organization,

Bangalore, India

Submitted by-

KULDEEP PUROHIT

B.Tech (E.Engg.)

INDIAN INSTITUTE OF TECHNOLOGY MANDI

Himachal Pradesh

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout this project work.

I take immense pleasure in thanking **Dr. Subrata Rakshit**, Scientist 'F', Head, Computer Vision Group, CAIR, Defense Research & Development Organization, Bangalore for allowing me to carry out the project in Computer Vision Group.

I wish to express our deep sense of gratitude to my project guide **Mr. Arshad Jamal,** Scientist 'E', for devoting his precious time. His esteemed guidance and expertise has helped to complete the project in time. He has always been a source of inspiration.

I also thank **Mrs. Babitha** for her help and support during the course of the project. Finally, I thank all the staff of Computer Vision Group for their support and encouragement.

I also thank all my friends who supported me in completing this project.

-Kuldeep Purohit

TABLE OF CONTENTS

ABSTRACT

In this project, the problem of object detection and tracking in a new and challenging domain of wide area surveillance has been tackled. This problem poses several challenges: large camera motion, strong parallax, large number of moving objects, and small number of pixels on target, single channel data and low frame-rate of video. The method implemented here overcomes these challenges and evaluate it on UAV imagery. We use median background modeling which requires few frames to obtain a workable model. We remove false detections due to parallax and registration errors using gradient information of the background image. In order to keep complexity of the tracking problem manageable, the scene can be divided into grid cells, solve the tracking problem optimally within each cell using bipartite graph matching and then link tracks across cells. Besides tractability, grid cells allow us to define a set of local scene constraints such as road orientation and object context. The method uses these constraints as part of cost function to solve the tracking problem which allows us to track fast-moving objects in low frame-rate videos. The implementation of the method on UAV videos demonstrates the successful tracking and overcoming the problems mentioned above.

INTRODUCTION

Recently a new sensor platform has appeared on the scene, allowing for persistent monitoring of very large areas. This type of dataset belongs to the domain of Wide Area Surveillance (WAS), which could be used to monitor large urban environments, as an aid in disaster relief, as well as traffic and accident management. Monitoring such a large amount of data with a human operator is not feasible, which calls for an automated method of processing the data. An initial step for such a system would be the detection and tracking of moving objects such as vehicles moving on highways, streets and parking lots.

PROBLEM STATEMENT

This problem poses several challenges: large camera motion, strong parallax, large number of moving objects, and small number of pixels on target, single channel data and low frame-rate of video. First, objects in WAS data are much smaller, with vehicle sizes ranging from 4 to 70 pixels in gray-scale Imagery Second, the data is sampled only at 2 Hz which when compared against more common frame-rates of 15-30 Hz is rather low. Third, the traffic is very dense comprising thousands of objects in a scene.

The first issue makes object detection difficult, but more importantly it disallows the use of shape and appearance models for objects during tracking and necessitates an accurate velocity model. However, issues two and three make initialization of a velocity model extremely difficult. High speed of vehicles on highway combined with low sampling rate of the imagery results in large displacement of objects between frames. This displacement is larger than spacing between objects; making proximity based initial assignment produce incorrect labeling which results in incorrect velocity model.

SOLUTION PROPOSED

As a solution, High speed 60Hz cameras have been used to address this problem in dense scenarios, where the high sampling rate makes initial proximity based assignment meaningful. Instead, method leverages structured nature of the scene to obtain a set of constraints and use them in the tracking function. Specifically, it derives road orientation and traffic context constraints to help with initial assignment.

Method also uses median background modeling which requires few frames to obtain a workable model. It removes false Detections due to parallax and registration errors using

gradient information of the background image. In order to keep complexity of the tracking problem manageable, the scene is divided into grid cells; tracking problem is solved optimally within each cell using bipartite graph matching and then tracks are linked across cells. Besides tractability, grid cells allow us to define a set of local scene constraints such as road orientation and object context. We use these constraints as part of cost function to solve the tracking problem which allows us to track fast-moving objects in low frame-rate videos.

CONCEPTS INVOLVED

IMAGE REGISTRATION

Image registration is the process of transforming different sets of data into one coordinate system. Data may be multiple photographs from different times, or from different viewpoints. Registration is necessary in order to be able to compare or integrate the data obtained from these different measurements.
The methods discussed in the rest of the report are based on feature point extraction method.

SIFT DETECTOR AND DESCRIPTOR

Scale-invariant feature transform (or **SIFT**) is an algorithm in computer vision to detect and describe local features in images. It is based on scale space representation using Gaussian filter.

The interest points, which are called key points in the SIFT framework, are detected. For this, the image is convolved with Gaussian filters at different scales, and then the difference of successive Gaussian-blurred images is taken. Key points are then taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. Specifically, a DoG image $D(x, y, \sigma)$ is given by

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma),$$

where $L(x, y, k\sigma)$ is the convolution of the original image $I(x, y)$ with the Gaussian blur $G(x, y, k\sigma)$ at scale $k\sigma$, i.e.,

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$$

Hence a DoG image between scales $k_i\sigma$ and $k_j\sigma$ is just the difference of the Gaussian-blurred images at scales $k_i\sigma$ and $k_j\sigma$. For scale-space extrema detection in the SIFT algorithm, the image is first convolved with Gaussian-blurs at different scales. The convolved images are grouped by octave (an octave corresponds to doubling the value of σ), and the value of k_i is selected so that we obtain a fixed number of convolved images per octave. Then the Difference-of-Gaussian images are taken from adjacent Gaussian-blurred images per octave.

Once DoG images have been obtained, keypoints are identified as local minima/maxima of the DoG images across scales. This is done by comparing each pixel in the DoG images to its eight neighbors at the same scale and nine corresponding neighboring pixels in each of the neighboring scales. If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint.

This keypoint detection step is a variation of one of the blob detection methods developed by Lindeberg by detecting scale-space extrema of the scale normalized Laplacian, that is detecting points that are local extrema with respect to both space and scale, in the discrete case by comparisons with the nearest 26 neighbours in a discretized scale-space volume. The difference of Gaussians operator can be seen as an approximation to the Laplacian, here expressed in a pyramid setting.

KEYPOINT LOCALIZATION

After scale space extrema are detected (their location being shown in the uppermost image) the SIFT algorithm discards low contrast keypoints (remaining points are shown in the middle image) and then filters out those located on edges. Resulting set of keypoints is shown on last image.

Scale-space extrema detection produces too many keypoint candidates, some of which are unstable. The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

INTERPOLATION OF NEARBY DATA FOR ACCURATE POSITION

First, for each candidate keypoint, interpolation of nearby data is used to accurately determine its position. The initial approach was to just locate each keypoint at the location and scale of the candidate keypoint. The new approach calculates the interpolated location of the extremum, which substantially improves matching and stability. The interpolation is done using the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function, $D\left(x, y, \sigma\right)$ with the candidate keypoint as the origin. This Taylor expansion is given by:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

where D and its derivatives are evaluated at the candidate keypoint and $\mathbf{x} = (x, y, \sigma)$ is the offset from this point. The location of the extremum, $\hat{\mathbf{x}}$, is determined by taking the derivative of this function with respect to $\mathbf{x}$ and setting it to zero. If the offset $\hat{\mathbf{x}}$ is larger than 0.5 in any dimension, then that's an indication that the extremum lies closer to another candidate keypoint. In this case, the candidate keypoint is changed and the interpolation performed instead about that point. Otherwise the offset is added to its candidate keypoint to get the interpolated estimate for the location of the extremum. A similar subpixel determination of the locations of scale-space extrema is performed in the real-time implementation based on hybrid pyramids developed by Lindeberg and his co-workers

DISCARDING LOW-CONTRAST KEYPOINTS

To discard the keypoints with low contrast, the value of the second-order Taylor expansion $D(\mathbf{x})$ is computed at the offset $\hat{\mathbf{x}}$. If this value is less than 0.03, the candidate keypoint is discarded. Otherwise it is kept, with final location $\mathbf{y} + \hat{\mathbf{x}}$ and scale σ, where $\mathbf{y}$ is the original location of the keypoint at scale σ.

ELIMINATING EDGE RESPONSES

The DoG function will have strong responses along edges, even if the candidate keypoint is not robust to small amounts of noise. Therefore, in order to increase stability, we need to eliminate the keypoints that have poorly determined locations but have high edge responses.

For poorly defined peaks in the DoG function, the principal curvature across the edge would be much larger than the principal curvature along it. Finding these principal curvatures amounts to solving for the eigenvalues of the second-order Hessian matrix, **H**:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

The eigenvalues of **H** are proportional to the principal curvatures of D. It turns out that the ratio of the two eigenvalues, say α is the larger one, and β the smaller one, with ratio r = α / β, is sufficient for SIFT's purposes. The trace of **H**, i.e., $D_{xx} + D_{yy}$, gives us the sum of the two eigenvalues, while its determinant, i.e., $D_{xx}D_{yy} - D_{xy}^2$, yields the product. The ratio $\mathrm{R} = \mathrm{Tr}(\mathbf{H})^2 / \mathrm{Det}(\mathbf{H})$ can be shown to be equal to $(r+1)^2/r$, which depends only on the ratio of the eigenvalues rather than their individual values. R is minimum when the eigenvalues are equal to each other. Therefore the higher the absolute difference between the two eigenvalues, which is equivalent to a higher absolute difference between the two principal curvatures of D, the higher the value of R. It follows that, for some threshold eigenvalue ratio r_{th}, if R for a candidate keypoint is larger than $(r_{\mathrm{th}}+1)^2/r_{\mathrm{th}}$, that keypoint is poorly localized and hence rejected. The new approach uses r_{th} = 10.

This processing step for suppressing responses at edges is a transfer of a corresponding approach in the Harris operator for corner detection. The difference is that the measure for thresholding is computed from the Hessian matrix instead of a second-moment matrix (see structure tensor).

ORIENTATION ASSIGNMENT

In this step, each keypoint is assigned one or more orientations based on local image gradient directions. This is the key step in achieving invariance to rotation as the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.

First, the Gaussian-smoothed image $L(x, y, \sigma)$ at the keypoint's scale σ is taken so that all computations are performed in a scale-invariant manner. For an image sample $L(x, y)$ at scale σ, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, are precomputed using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

The magnitude and direction calculations for the gradient are done for every pixel in a neighboring region around the keypoint in the Gaussian-blurred image L. An orientation histogram with 36 bins is formed, with each bin covering 10 degrees. Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint. The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint. In the case of multiple orientations being assigned, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation.

Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, and match moving. This feature description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.

Another important characteristic of these features is that the relative positions between them in the original scene shouldn't change from one image to another.However, in practice SIFT detects and uses a much larger number of features from the images, which reduces the contribution of the errors caused by these local variations in the average error of all feature matching errors.

RANSAC

Random Sample Consensus (RANSAC) to simultaneously solve the correspondence problem and estimate the fundamental matrix related to a pair of stereo cameras. It is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers.

This algorithm works on a basic assumption that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, and "outliers" which are data that do not fit the model. In addition to this, the data can be subject to noise. The outliers can come, e.g., from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.

It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iteration are allowed. The method is best described by following pseudo code

```
Input:
    data - a set of observations
    model - a model that can be fitted to data
    n - the minimum number of data required to fit the model
    k - the number of iterations performed by the algorithm
    t - a threshold value for determining when a datum fits a model
    d - the number of close data values required to assert that a model fits well to data
Output:
    best_model - model parameters which best fit the data (or nil if no good model is found)
    best_consensus_set - data points from which this model has been estimated
    best_error - the error of this model relative to the data

iterations := 0
best_model := nil
best_consensus_set := nil
best_error := infinity
while iterations < k
    maybe_inliers := n randomly selected values from data
    maybe_model := model parameters fitted to maybe_inliers
    consensus_set := maybe_inliers

    for every point in data not in maybe_inliers
        if point fits maybe_model with an error smaller than t
            add point to consensus_set

    if the number of elements in consensus_set is > d
        (this implies that we may have found a good model,
        now test how good it is)
        this_model := model parameters fitted to all points in consensus_set
        this_error := a measure of how well this_model fits these points
        if this_error < best_error
            (we have found a model which is better than any of the previous ones,
            keep it until a better one is found)
            best_model := this_model
            best_consensus_set := consensus_set
            best_error := this_error

    increment iterations

return best_model, best_consensus_set, best_error
```

HOMOGRAPHY

In the field of computer vision, any two images, acquired from a pinhole camera, of the same planar surface are related by a 3x3 matrix known as homography matrix. For a set of point correspondences $\{x_i \leftrightarrows x_i'\}$ in two images there is a homography matrix H such that $x_i' = Hx_i$ where x represents a homogeneous image coordinate $(x, y, w)^T$ and H is a 3x3 matrix which has only 8 degrees of freedom as it is defined up to a scale factor. Different planes in the images have different homographys. If a homography is computed using at

least three points correspondences on the ground plane, other points on the ground plane are found using the same homography matrix. As the point correspondences are generally noisy, we use a RANSAC based robust estimation technique for accurate computation of the homography matrix.

AFFINE & PROJECTIVE HOMOGRAPHY

When the image region in which the homography is computed is small or the image has been acquired with a large focal length or the camera center is fixed during acquisition of two images, an *affine homography* is a more appropriate model of image displacements. Almost similar usage is there for Projective homography. These are special type of a general homography whose last row is fixed to [0 0 1] i.e. $h_{31}=h_{32}= 0$ and $h_{33} =1$.

DETECTION

MEDIAN IMAGE FILTERING

This is a background estimation method where a median image is calculated for a set of images (converted into a common frame using homography) and is subtracted from the actual image to eliminate the background and leaves only the objects under motion.

It has optimum usage in situations of similar kind where we have low frame rates so that median image has to be calculated for lesser number of frames. Another advantage over mean image is that it makes false motion detections (due to parallax and registration errors) less prominent.

GRADIENT SUPPRESSION

This technique is necessary for removing false motion detections due to parallax and registration errors. When we use homography for transformation, we essentially assume a planar surface but this assumption does not hold for tall building. Due to misalignment of the pixels belonging to these objects between the successive frames, they appear to move between even aligned frames.

So, we subtract the gradient of the median image from the difference image to get rid of all unwanted errors, including the registration errors also since they too manifest along gradients.

TRACKING

BIPARTITE GRAPH MATCHING

In the mathematical field of graph theory, a **bipartite graph** (or **bigraph**) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; that is, U and V are independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles.

Any graph with no odd cycles is bipartite. As a consequence of this:

- Every tree is bipartite.
- Cycle graphs with an even number of vertices are bipartite.
- Any planar graph where all the faces in its planar representation consist of an even number of edges is bipartite. Special cases of this are grid graphs and square graphs, in which every inner face consists of 4 edges.

These graphs are constructed taking objects in adjacent frames as vertices. Weight is calculated by adding various weighing components (Position, Velocity, Context and the Global velocity components). Every object in frame 't' is mapped to every object in frame 't+1' and the bipartite graph thus formed is used in Hungarian Algorithm for Tracking.

HUNGARIAN ALGORITHM

The **Hungarian method** is a combinatorial optimization algorithm which solves the assignment problem in polynomial time. The time complexity of the original algorithm was $O(n^4)$, however, later, it was noticed that it can be modified to achieve an $O(n^3)$ running time.

The bipartite graph weights obtained are represented in matrix form and then given as an input argument to this function. The function returns a sparse matrix with same size as the input Weight matrix but values 1 at indices with correct matching (Maximum Weight assignment).

i.e. Given a nonnegative $n \times n$ matrix, where the element in the i-th row and j-th column represents the cost of assigning the j-th job to the i-th worker. The algorithm finds an assignment of the rows to the columns that has minimum cost. If the goal is to find the assignment that yields the maximum cost, the problem can be altered to fit the setting by replacing each cost with the maximum cost subtracted by the cost.

For a square matrix, this algorithm seeks for assignment, but first applies appropriate row or column operations on it as follows:-

Subtracting the smallest element from each element of the corresponding row/column gives at least one zero in the line. If all the zeros are corresponding to distinct row/column indices, the assignment is solved, but if they aren't, further subtraction for a few steps ultimately leads to the optimized matching match.

WORKING PROCEDURE

The implemented method consists of the following modules. First, we register images using a point correspondence based alignment algorithm. Then we perform motion detection via a median image background model. We perform gradient suppression of the background difference image to remove motion detection errors due to parallax and registration. Once we have moving object blobs, we divide the scene into a number of grid cells and optimally track objects within each grid cell using Hungarian algorithm. The use of overlapping cells is a novel idea which can make possible the use of $O(n3)$ Hungarian algorithm in a scene containing thousands of objects and provides a way to define a set of structured scene constraints to disambiguate initialization of the algorithm.

IMAGE REGISTRATION

Prior to motion detection in aerial video, we remove global camera motion. We can use point based registration algorithms, and it is much faster than the direct registration method. We detect **Harris corners** in frames at time t as well as at time $t + 1$. Then we compute **SIFT descriptor** around each point and match the points in frame t to points in frame $t+1$ using the descriptors. Finally, we robustly fit a **homography** H^{t+1}_{t} using **RANSAC**, that describes the transformation between top 200 matches. Once homographies between individual frames have been computed, we warp all the images to a common reference frame by concatenating the frame to frame homographies.

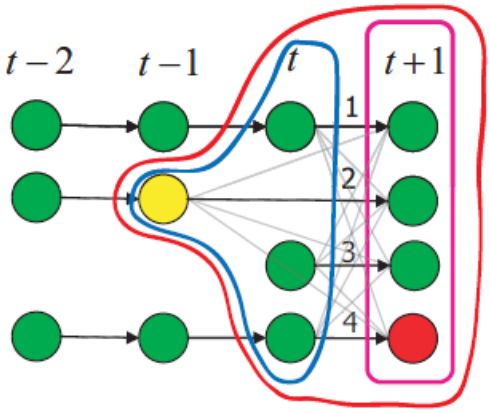
DETECTION

After removing global camera motion, we detect local motion generated by objects moving in the scene. To perform motion detection, we first need to model background, and then moving objects can be considered as outliers with respect to the background. Since it is always that objects are moving in the scene, we do not have the luxury of object-free initialization period, not even a single frame. Additionally, since the cameras move, we need to build the background model in as few frames as possible; otherwise our active area becomes severely limited. Furthermore, high density of moving objects in the scene combined with low sampling rate makes the objects appear as outliers. These outliers can be seen as ghosting artifacts affecting the mean, the large number of outliers make the standard deviation high, allowing more outliers to become part of the model, which means many moving objects become part of the background model and are not detected. A mixture of Gaussians makes background modeling even more complex by allowing each pixel to have multiple backgrounds. This is useful when background changes, such as in the case of a moving tree branch in surveillance video. This feature, however, does not alleviate any of the problems we highlighted above. Therefore, we avoid probabilistic models in favor of simple median image filtering, which learns a background model with less artifacts using fewer frames. We found that 10 frame median image has fewer ghosting artifacts than mean image. To obtain a comparable mean image, it has to be computed over at least four times the number of frames which results in smaller field of view and makes false motion detections due to parallax and registration errors more prominent. We perform motion

detection in the following manner. For every 10 frames we compute a median background image B, next we obtain difference image i.e. $Id = |I - B|$. Prior to thresholding the difference image, we perform gradient suppression. This is necessary to remove false motion detections due to parallax and registration errors. Since we fit a homography to describe the transformation between each pair of frames, we are essentially assuming a planar scene. This assumption does not hold for portions of the image that contain out of plane objects such as tall buildings. Pixels belonging to these objects are not aligned correctly between frames and hence appear to move even in aligned frames. Additionally due to large camera motion, there may be occasional errors in the alignment between the frames. An example, consider a small portion of an image containing a tall building. Due to parallax error, the building produces false motion detections along its edges. We suppress these by subtracting gradient of the median image ∇B (second column) from the difference image i.e. $Ir_d = I_d - \nabla B$. The top row shows a planar section of the scene and contains moving objects. This procedure successfully suppresses false motion detections due to parallax error without removing genuine moving objects. Also, the method has the advantage of suppressing false motion detections due to registration errors, since they too manifest along gradients. Note that above method works under an assumption that areas containing moving objects will not have parallax error which is valid for roads and highways.

TRACKING

After detecting moving objects, we track them across frames using bipartite graph matching between a set of *label* nodes (circled in blue) and a set of *observation* nodes (circled in magenta). The assignment is solved optimally using the Hungarian algorithm which has complexity $O(n^3)$ where n is the number of nodes.



For each grid cell in every pair of frames we construct the following graph. Figure 5 shows an example graph constructed for assigning labels between frames t and t+1. We add a set of nodes for objects visible at t to the set of *label* nodes. A set of nodes for objects visible at t + 1 are added to the set of *observation* nodes, both types are shown in green. Since objects

can exit the scene, or become occluded, we add a set of occlusion nodes to our *observation* nodes, shown in red. To deal with the case of reappearing objects, we also add *label* nodes for objects visible in the set of frames between $t-1$ and $t-p$, shown in yellow. We fully connect the *label* set of nodes to the *observation* set of nodes, using four types of edges.
1. Edge between label in frame t and an observation in frame $t+1$.
2. Edge between label in frame $t-p$ and an observation in frame $t+1$.
3. Edge between a new track label in frame t and an observation in frame $t+1$.
4. Edge between a label and an occlusion node.

We define edge weights in the following manner. Weight for edge of type 3 is simply a constant δ. Weights for edges of type 1 and 2 contain velocity orientation and spatial proximity components. Spatial proximity component C_p is given by

$$C_p = 1 - \frac{\|x^{t-k} + v^{t-k}(k+1) - x^{t+1}\|}{\sqrt{S_x^2 + S_y^2}},$$

where x is the position of the object, S_x and S_y are the dimensions of the window within which we search for a new object and k is the time past since last observation of the object. Velocity orientation component C_v is given by Detection

$$C_v = \frac{1}{2} + \frac{\boldsymbol{v}^t \cdot \boldsymbol{v}^{t+1}}{2\|\boldsymbol{v}^t\|\|\boldsymbol{v}^{t+1}\|},$$

where v^t is the last observed velocity of an object, v^{t+1} is the difference between x^{t+1}, the position of observation in current frame, and x^{t-k}, the last observed position of object at frame $t-k$.

We define the weight for edges of type 1 and 2 as $w = \alpha C_v + (1-\alpha)C_p$.

We found these to be sufficient when object's velocity is available. If on the other hand, velocity of the object is unavailable as in initial two frames or when new objects appear in the scene, we use structured scene constraints to compute weights for edges.

Therefore, we introduce an additional formation context constraint. If we are trying to match an object in frame t (or $t-k$) to an observation in frame $t+1$, we compute object context as a 2 dimensional histogram of vector orientations and magnitudes between an object and its neighbors. In order to account for small intra-formation changes, when computing the context histograms Φ_a and Φ_b, we add a 2D Gaussian kernel centered on the bin to which a particular vector belongs. Furthermore, since 0° and 360° are equivalent, we make the kernel wrap around to other side of orientation portion of the histogram. The road orientation constraint component is defined as

$$C_g = \frac{1}{2} + \frac{|\boldsymbol{g} \cdot \boldsymbol{v}^{t+1}|}{2\|\boldsymbol{g}\|\|\boldsymbol{v}^{t+1}\|}$$

The purpose of this constraint is to prevent tracks from traveling across the road. The context constraint is the histogram intersection between histograms Φ_a and Φ_b:

$$C_c = \sum_{p}^{Nbins} \sum_{q}^{Mbins} min(\Phi_a^{p,q}, \Phi_b^{p,q})$$

Finally, weight for edge of type 3 is computed as follows,
$w = \alpha_1 C_g + \alpha_2 C_p + (1 - \alpha_1 - \alpha_2) C_{c.}$
We solve the resulting bipartite graph using Hungarian algorithm. We track all objects within each grid cell by performing the above procedure for all frames.

MATLAB_IMPLEMENTATION

%%IMAGE REGISTRATION AND OBJECT DETECTION

```
                    %%Video reading.and generating image sequence

        vdo=aviread('seq01.avi')
        frames=size(vdo,2)                          % NUMBER OF FRAMES
        for i=1:frames
        I(:,:,i)=rgb2gray(vdo(i).cdata);            % I- (GRAYSCALE IMAGE)
        end
        for i=1:frames
        I_(:,:,i)=imcrop(I(:,:,i),[0 0 320 240]);   % SETTING FRAME SIZE
        end
        i=1;

        while(i<50)
        c=i+2;
        [match1 match2]=siftMatch(I_(:,:,c),I_(:,:,i));    %SIFT DESCRIPTOR MATCHING

        H(:,:,i)=vgg_H_from_x_lin(match1',match2');         %Homography matrix for two
                                                            %alternate
frames
        i=c;
        end


for c=2:58


        j=c+2;
       count=1;
        while(j<(c+10))
        i=j;
        Hfk=[1 0 0 ; 0 1 0 ; 0 0 1];

while(i>=(c+2))
        Hfk=Hfk*Hio;                    % calculating homography matrices for
        i=i-2;                          % 5 pairs of frames
end

Hio=Hfk;

for k=1:240
for l=1:320
if((Hio(1,3)>0)&&(Hio(2,3)>0))
        x=Hio(1,1)*k+Hio(1,2)*l+Hio(2,3);
        y=Hio(2,1)*k+Hio(2,2)*l+Hio(1,3);
        c1=ceil(x);
        c2=ceil(y);
        image2(c1,c2)=I_(k,l,j);            % image2 is the transformed image
        end
```

```
end
end

Ix2=imcrop(image2, [0 0 320 240]);              % cropping to the grid size
MAT(:,:,count)=Ix2;                             % adding homography images to a 3D array MAT
count=count+1;
j=j+2;

end

for p=1:240
for q=1:320
        MAT(p,q,:)=ksort(MAT(p,q,:));           % sorting 3d array for each pixel
end
end

J(:,:,c)=I_(:,:,c)-MAT(:,:,3);                  % MAT(:,:,3)- the median difference image
[imx imy]=gaussgradient(MAT(:,:,3),1.0);        % x and y gradients
J(:,:,c)=J(:,:,c)-imx;
J(:,:,c)=J(:,:,c)-imy;

figure, imshow(J(:,:,c))                        % final image without background
end
```

OBJECT TRACKING

```
vdo=aviread('uav_out.avi');                     % READING THE VIDEO WITH
                                                % BACKGROUND SUBTRACTED
frames=size(vdo,2);

for i=1:50
I(:,:,:,i)=vdo(i).cdata;
end

for i=1:49
Ig(:,:,i)=rgb2gray(I(:,:,:,i));
end
for i=1:49
I_(:,:,i)=imcrop(Ig(:,:,i),[321 0 640 240]);
end

for count=1:49
L_(:,:,count)=bwlabel(I_(:,:,count));           % L- image with each object labeled
end
```

%PROXIMITY COMPONENT (FIRST FRAME):-

%%%% HERE, 't' REFERS TO FRAME NUMBER

```
sa=regionprops(L_(:,:,t),I_(:,:,t),{'Centroid'});          % array containing indices of
                                                              %centroids of all objects in
frame 't'
sb=regionprops(L_(:,:,t+1),I_(:,:,t+1),{'Centroid'});      %- (same for frame t+1)

for i=1:numel(sa)                                  % numel(s1)- number of objects stored in s1
for j=1:numel(sb)

        x2=sb(j).Centroid(1);
        y2=sb(j).Centroid(2);
        x=sa(i).Centroid(1);
        y=sa(i).Centroid(2);
        xn=x-x2;
        yn=y-y2;

Num=sqrt(xn*xn+yn*yn);
Den=sqrt(( size(I_(:,:,t),1) * size(I_(:,:,t),1)) + (size(I_(:,:,t),2) *size(I_(:,:,t),2)) );
```

$$C_p = 1 - \frac{\|x^{t-k} + v^{t-k}(k+1) - x^{t+1}\|}{\sqrt{S_x^2 + S_y^2}},$$

```
                                       %%
Cp(i,j,t)=1-(Num/Den);                 % proximity component for object 'I' in frame t
                                             % and object 'j' in frame t+1

end
end
```

%COMPUTING GLOBAL VELOCITY FOR EACH FRAME

```
%input parameter- t
for t=1:48
index=1;
s1=regionprops(L_(:,:,t),I_(:,:,t),{'Centroid'});
s2=regionprops(L_(:,:,t+1),I_(:,:,t+1),{'Centroid'});
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%<u>***PSUEDO CODE***</u>

for all *d* ∈ *Di;jt* **do**
for all *d′* ∈ *Di;jt*+1 **do**
θ = tan−1 (*d′* − *d*) % direction of motion for each pair of objects
Store *θ* in *Θ*
end for
end for
h = *histogram*(*Θ*) % histogram for various angles obtained
Find bin *ψ* s.t. *mode*(*h*) ∈ *ψ* *% select bin with maximum height*
θ= *mean*(*θ*|*θ* ∈ *ψ*)
→*g* (*i, j*) = [cos(*θ*) sin(*θ*)] % global velocity for frame 't'

%%

%ROAD CONSTRAINT COMPONENT:-

```
for t=1:46
sc=regionprops(L_(:,:,t),I_(:,:,t),{'Centroid'});
sd=regionprops(L_(:,:,t+1),I_(:,:,t+1),{'Centroid'});
xg=g(1,t);
yg=g(2,t);

for i=1:numel(sc)
for j=1:numel(sd)

x1=sc(i).Centroid(1);
y1=sd(j).Centroid(2);
xv=sd(j).Centroid(1)-sc(i).Centroid(1);
yv=sd(j).Centroid(2)-sc(i).Centroid(2);

Num=abs(xg*xv+yg*yv);                    % calculating dot product of object velocity and
                                         %Road constraint velocity 'g'

Den=2*(sqrt(xg*xg+yg*yg))* (sqrt(xv*xv+yv*yv));

Cg(i,j,t)=0.5+(Num/Den);                 %%

end
end
end
```

$$C_g = \frac{1}{2} + \frac{|g \cdot v^{t+1}|}{2\|g\|\|v^{t+1}\|}$$

%COMPUTE CONTEXT

```
for t=1:100

s1=regionprops(L_(:,:,t),I_(:,:,t),{'Centroid'});        % s1 SET OF LABELS FOR ALL
                                                          %OBJECTS IN THE FRAME 't'
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**_PSUEDO CODE_**

for all *c in s1* **do**

if ||*Oct* – *Oat*||2 < *r* **then** **%** '*Phi*' is calculated for object % 'a' with respect to all the objects %in s1

θ = tan−1 (*Oct*− *Oat*)

d = ||*Oct*− *Oat*||2

Φ = *Φ* + *N*(*μ*,*Σ*)

◃*N* centered on (*d*, *θ*)

end if

% N(mu,sigma) is the 2D Gaussian kernel added to make the histogram more continuous.

%%

%CALCULATING CONTEXT CONSTRAINT BY HISTOGRAM INTERSECTION (FRAME T)

```
for t=1:100

sk=regionprops(L_(:,:,t),I_(:,:,t),{'Centroid'});
sl=regionprops(L_(:,:,t+1),I_(:,:,t+1),{'Centroid'});

sm=zeros(10,10,52);

for i=1:numel(sk)
for j=1:numel(sl)

        Q=hist3(Phi(:,:,i,t),[5 5]);
        R=hist3(Phi(:,:,j,t+1),[5 5]);

for temp1=1:5
for temp2=1:5
        sm(i,j,t)=sm(i,j,t)+min(Q(temp1,temp2),R(temp1,temp2));        % intersection of histograms
                                                                        %Phi(:,:,:,t) and
Phi(:,:,:,t+1)
end
end

sm(i,j,t);
```

$$C_c = \sum_{p}^{Nbins} \sum_{q}^{Mbins} min(\Phi_a^{p,q}, \Phi_b^{p,q})$$

```
Cc(i,j,t)=sm(i,j,t)/10;                                %
                                                        % Cc(:,:,t) is contribution to weight  of
                                                                % context constraint in overall mapping


end
end
end
end
```

%WEIGHT CALCULATION CODE(FOR FIRST FRAME)

```
t=1;
sa=regionprops(L_(:,:,t),I_(:,:,t),{'Centroid'});
sb=regionprops(L_(:,:,t+1),I_(:,:,t+1),{'Centroid'});

Cgn(:,:,t)=isnan(Cg(:,:,t));                        % FOR ENSURING THAT MATRICES DON'T
Cpn(:,:,t)=isnan(Cp(:,:,t));                        % NaN VALUES (not a number).

for i=1:numel(sa)
```

```
for j=1:numel(sb)
        if((Cpn(i,j,t)==0)&&(Cgn(i,j,t)==0))
        Weight(i,j,t)=0.2*Cg(i,j,t)+0.4*Cp(i,j,t)+0.4*Cc(i,j,t);
end
if((Cpn(i,j,t)==1)&&(Cgn(i,j,t)==0))
        Weight(i,j,t)=0.3*Cg(i,j,t)+0.7*Cc(i,j,t);
end

if((Cpn(i,j,t)==0)&&(Cgn(i,j,t)==1))
        Weight(i,j,t)=0.5*Cp(i,j,t)+0.5*Cc(i,j,t);
end

end
end
```

%ASSIGNMENT CODE FOR FIRST FRAME-

```
[match1 temp]=Hungarian(ones(10,10)-Weight(:,:,t))      % Hungarian algorithm for
                                                        %matching for maximum
weight in                                               %the matrix Weight(:,:,t)

count=1;
for x=1:numel(sa)
for y=1:numel(sb)
if(match1(x,y)==1)
        Chain(1,1,count)=sa(x).Centroid(1);
        Chain(2,1,count)=sa(x).Centroid(2);
        Chain(1,2,count)=sb(y).Centroid(1);
        Chain(2,2,count)=sb(y).Centroid(2);
       count=count+1;
end
end
end
```

%%%%%%%%%%%%%% Chain is a 2xFxN matrix which stores the x,y indices %%%% %%%%%%%%%%%%% of the 'N' objects tracked for each frame 'F'.
THIS IS BASE OF THE CODE FOLLOWED FOR ALL THE FRAMES IN SEQUENCE TO GENERATE THE FINAL TRACKING DATA OF OBJECTS ***"Chain".***

EXPERIMENTAL RESULTS

The method's tracking ability was tested by executing it on several real UAV videos.

Some of the Snapshots taken for single frames of a particular UAV imagery are shown below. Each snapshot is showing the actual input video in the left half and the objects detected after background subtraction in the right half. The actual input video contains the tracks for objects in the video in shape of a circle with color specific to the corresponding object.

FRAME: 3

FRAME: 5

FRAME: 10

FRAME: 90

The evaluation method for measuring performance of both detection and tracking involves computing the following distance measure between generated tracks and ground truth tracks:

$$D(T_a, G_b) = \frac{1}{|\Omega(T_a, G_b)|^2} \sum_{t \in \Omega(T_a, G_b)} \|\mathbf{x}_t^a - \mathbf{x}_t^b\|^2,$$

where $\Omega(T_a,G_b)$ denotes the temporal overlap between T_a and G_b, $|.|$ denotes cardinality while $\|.\|$ is the Euclidean norm.

CONCLUSIONS

We analyzed the challenges of a new aerial surveillance domain called Wide Area Surveillance, and proposed a method for detecting and tracking objects in this data. Our method specifically deals with difficulties associated with this new type of data: unavailability of object appearance, large number of objects and low frame rate. We evaluated proposed method and provided both quantitative and qualitative results.

FURTHER EXTENSION

After tracking all objects within each grid cell by performing the above procedure for all frames, we can find and link tracks that have crossed the cell boundaries, utilizing the overlapping regions of the neighboring grid cells. This technique enables this algorithm for tracking objects in very wide area surveillance videos containing thousands of objects.

HANDLING MULTIPLE CAMERAS

There can be several possible frameworks for tracking objects across overlapping cameras which employ inter-camera transformations. One possible way is to establish Correspondences at the track level where objects are detected and tracked in each camera independently, and afterwards, tracks belonging to the same object are linked. Background for a particular frame of a camera can only be modeled on overlapping region of all frames used for background. This reduces the area of region where objects can be detected. When objects are detected in cameras separately, reduction in detection regions results in the loss of overlap between two cameras. While methods for matching objects across non-overlapping cameras exist low resolution and single channel data disallow the use of appearance models for object hand over, and reacquisition based on motion alone is ambiguous. The increased gap between cameras arising from detection adds further challenge to a data already characterized by high density of objects and low sampling rate of video.

In order to avoid above problems, we can perform detection and tracking in global coordinates. We first build concurrent mosaics from images of different cameras at a particular time instant using the Registration method in §2.1 and then register the mosaics treating each concurrent mosaic as a single image. One problem with this approach, however, is that cameras can have different Camera Response Functions or CRFs. This affects the median background, since intensity values for each pixel now come from multiple cameras causing performance of the detection method to deteriorate. To overcome this issue, we adjust the intensity of each camera with respect to a reference camera using the gamma function, i.e.

$I'_C(x, y) = \beta I_C(x, y)^\gamma$

We can determine β, γ by minimizing the following cost function:

$$\underset{\beta,\gamma}{\operatorname{argmin}} \sum_{(x,y)\in I_{C1}\cap I_{C2}} \left(I_{C1}(x,y) - I'_{C2}(x,y)\right)^2$$

BIBLIOGRAPHY

The major part of this project implemented the registration and tracking methods discussed in the following resources

- Research paper- Reilly, Vladimir, Haroon Idrees, and Mubarak Shah. "Detection and tracking of large number of targets in wide area surveillance." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2010.
- Book - "Digital Signal and Image Processing Using MATLAB" - Gerard Blanchet & Maurice Charbit

Other internet sources include

- en.wikipedia.org
- www.gigapedia.com
- www.mathworks.com
- www.eetimes.com
- www.aquaphoenix.com