Detection and Tracking of Large Number of Targets in Wide Area Surveillance

Vladimir Reilly, Haroon Idrees, and Mubarak Shah
vsreilly@eecs.ucf.edu,haroon.idrees@knights.ucf.edu,
shah@eecs.ucf.edu

Computer Vision Lab, University of Central Florida, Orlando, USA
http://www.cs.ucf.edu/~vision

Abstract. In this paper, we tackle the problem of object detection and tracking in a new and challenging domain of wide area surveillance. This problem poses several challenges: large camera motion, strong parallax, large number of moving objects, small number of pixels on target, single channel data and low framerate of video. We propose a method that overcomes these challenges and evaluate it on CLIF dataset. 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, we divide the scene 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. We use these constraints as part of cost function to solve the tracking problem which allows us to track fast-moving objects in low framerate videos. In addition to that, we manually generated groundtruth for four sequences and performed quantitative evaluation of the proposed algorithm.

Keywords: Tracking, Columbus Large Image Format, CLIF, Wide Area Surveillance

1 Introduction

Recently a new sensor platform has appeared on the scene, allowing for persistent monitoring of very large areas. The dataset examined in this paper is Columbus Large Image Format or CLIF dataset. In CLIF, the sensor consists of six cameras with partially overlapping fields of view, mounted on an aerial platform flying at 7000 feet. All six cameras simultaneously capture 4016x2672 intensity images at 2 frames per second. See Figure 1(a) for an example of global camera mosaic.

CLIF 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

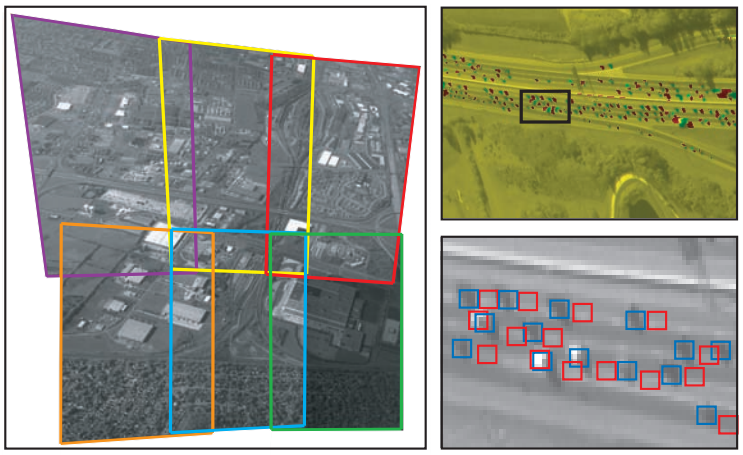

Fig. 1: (a) CLIF data - all six cameras. (b) top shows two consecutive frames overlayed in two different color channels: red is frame t, green is frame $t+1$. (b) bottom shows how far vehicles move between consecutive frames. Red boxes show vehicle positions in previous frame and blue boxes show vehicle positions in next frame.

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.

Data obtained from such a sensor is quite different from the standard aerial and ground surveillance datasets, such as VIVID and NGSIM, which have been used in [1, 2], as well as aerial surveillance scenario [3–5]. First, objects in WAS data are much smaller, with vehicle sizes ranging from 4 to 70 pixels in grayscale imagery, compared to over 1500 pixels in color imagery in the VIVID dataset. Second, the data is sampled only at 2 Hz which when compared against more common framerates of 15-30 Hz is rather low. Third, the traffic is very dense comprising thousands of objects in a scene compared to no more than 10 objects in VIVID and no more than 100 in NGSIM.

The first issue makes object detection difficult, but more importantly it disallows the use of shape and appearance models for objects during tracking as in [3, 1, 5, 6] 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.

Highspeed 60Hz cameras have been used to address this problem in dense scenarios [7, 8], where the high sampling rate makes initial proximity based assignment meaningful. Instead, we leverage structured nature of the scene to obtain a set of constraints and use them in our tracking function. Specifically,

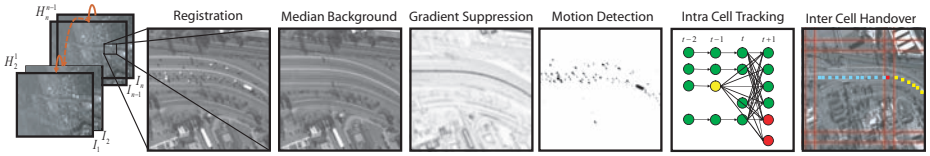

Fig. 2: This figure shows different stages of our pipeline. First, we remove global camera motion using point based registration, then we model the background using a 10 frame median image, perform background subtraction and suppress false positives due to parallax and registration errors. We track objects within individual grid cells, then perform handover of tracks between grid cells.

we derive road orientation and traffic context constraints to help with initial assignment. We cannot define context based on appearance of neighboring objects and background as has been done in [9], instead, we define a descriptor for the geometric relationship of objects with their respective neighbors.

2 Method

Our proposed method consists of the following modules (see figure 2 for reference). 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 makes possible the use of $O(n^3)$ 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. The contribution of our paper is a method for performing object detection and tracking in a new and challenging Wide Area Surveillance dataset characterized by low framerate, fast camera motion and a very large number of fast moving objects. In rest of the paper, we describe how we address all of the challenges and provide details for the individual modules.

2.1 Registration

Prior to motion detection in aerial video, we remove global camera motion. The structured man-made environment in these scenes and large amount of detail yields itself nicely to a point-matching based registration algorithm. It is also much faster than 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^{t+1} using RANSAC, that describes the transformation between top 200 matches. Once homographies

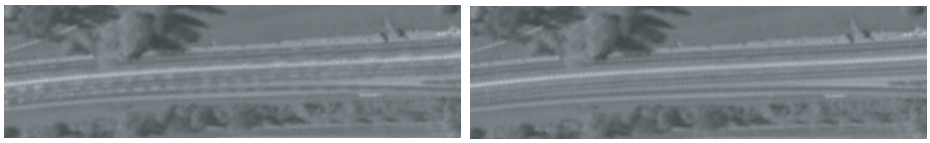

Fig. 3: Left shows a background model obtained using mean which has many ghosting artifacts from moving objects. Right shows background model obtained using median with almost no ghosting artifacts.

between individual frames have been computed, we warp all the images to a common reference frame by concatenating the frame to frame homographies.

2.2 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, then moving objects can be considered as outliers with respect to the background. Probabilistic modeling of the background as in [10] has been popular for surveillance videos. However, we found these methods to be inapplicable to this data. In the parametric family of models, each pixel is modeled as either a single or a mixture of Gaussians. First, there is problem with initialization of background model. 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 as shown in figure 3. In the case of single Gaussian model, besides 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 (figure 3). 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.

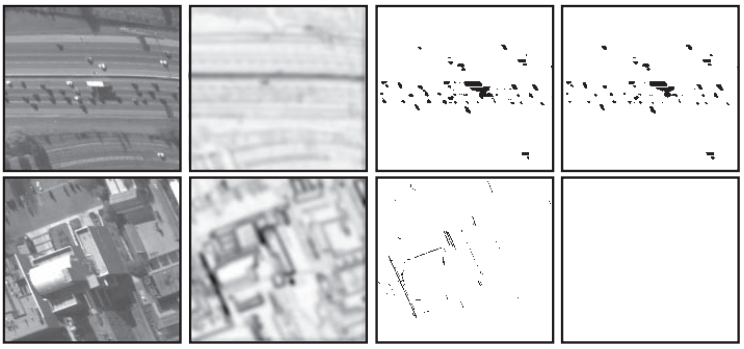

Fig. 4: Left to right: Section of original image, gradient of the median image, motion blobs prior to gradient suppression, motion blobs after gradient suppression. Bottom row shows an area of image that has false motion detections due to parallax and registration errors, top row shows a planar area of the image.

$I_d = |I - B|$. 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 of this is bottom row of figure 4 where we show a small portion of an image containing a tall building (left). Due to parallax error, the building produces false motion detections along its edges (third image from the left). We suppress these by subtracting gradient of the median image ∇B (second column) from the difference image i.e. $I_d^r = I_d - \nabla B$. The top row shows a planar section of the scene and contains moving objects. As evident from figure 4, 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.

2.3 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. When we have thousands of objects in the scene, an optimal solution for the entire scene is intractable. To overcome this problem, we break up the

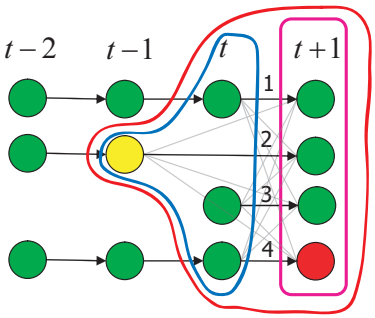

Fig. 5: The figure shows an example of the bipartite graph that we solve at every frame. Four different types of edges are marked with numbers.

scene into a set of overlapping grid cells (see figure 8). We solve the correspondence problem within each grid cell independently and then link tracks across grid cells. The use of grid has an additional advantage of allowing us to exploit local structured-scene constraints for objects within the grid cell, which will be discussed later.

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

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

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

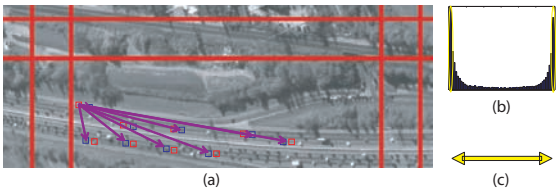

Fig. 6: This figure shows the process of estimating road orientation within a grid cell. Objects tracked in frame t are shown in red, objects detected in frame $t+1$ are shown in blue. (a) Obtain all possible assignments between objects in frame t and frame $t+1$. (b) Obtain a histogram of resulting possible velocities. (c) Take mean of velocities which contributed to the histogram peak.

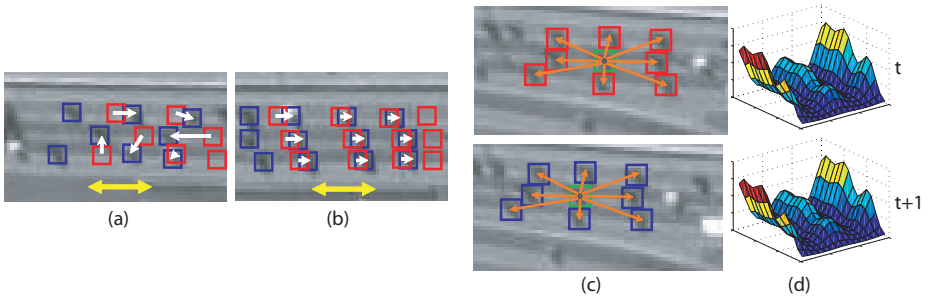

Fig. 7: Vehicles tracked at time t are shown in red while vehicles detected in frame $t+1$ are shown in blue. White arrows indicate the assignment of labels to objects based on proximity only and correspond to resulting velocities of objects. Yellow arrows indicate the road orientation estimate for this particular grid cell. (a) shows a case where road orientation estimate can be used to disambiguate the assignment of labels and (b) shows where it is not useful. To handle cases such as (b), we introduce a new constraint for context of each vehicle, shown in (c). At frames t and $t+1$ we compute vectors between vehicle of interest (green) and its neighbors (orange). We then compute a 2D histogram of orientations and magnitudes of the vectors shown in (c).

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

where $\boldsymbol{v}^t$ is the last observed velocity of an object, $\boldsymbol{v}^{t+1}$ 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 follows

$$w = \alpha C_v + (1-\alpha) C_p. \tag{3}$$

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.

Assigning labels based simply on proximity between object centroids is not meaningful in wide area scenario. Due to low sampling rate (2 Hz), high scene density and high speed of objects, proximity based assignment is usually incorrect (see figure 7). Therefore we use road orientation estimate and object context as constraints from the structured scene.

Road orientaion estimate $\boldsymbol{g}$ is computed for each grid cell in the following manner (see figure 6). First, we obtain all possible assignments between objects in frame t and $t+1$. This gives us a set of all possible velocities between objects at frames t and $t+1$. Next, we obtain a histogram of orientations of these velocities and take the mean of orientations that contributed to peak of the histogram. See **Algorithm 1** for a formal description.

Algorithm 1 Algorithm to compute global velocity for each cell in grid of size m x n using detections D_t and D_{t+1}.

procedure computeGlobalVelocity
  for $i \leftarrow 1, m$ **do**
    for $j \leftarrow 1, n$ **do**

      for all $d \in D_t^{i,j}$ **do**
        for all $d' \in D_{t+1}^{i,j}$ **do**
          $\theta = \tan^{-1}(d' - d)$
          Store θ in Θ
        end for
      end for

      $h = histogram(\Theta)$
      Find bin ψ s.t. $mode(h) \in \psi$
      $\theta' = mean(\theta | \theta \in \psi)$
      $\overrightarrow{g}(i,j) = [\cos(\theta') \ \sin(\theta')]$

    end for
  end for
end procedure

Algorithm 2 Algorithm to compute context $\Phi(O_t^a)$ for object a at frame t.

procedure computeContext
  for all c **do**

    if $\|O_t^c - O_t^a\|_2 < r$ **then**
      $\theta = \tan^{-1}(O_t^c - O_t^a)$
      $d = \|O_t^c - O_t^a\|_2$
      $\Phi = \Phi + \mathcal{N}(\mu, \Sigma)$
                ▷ $\mathcal{N}$ centered on (d, θ)
    end if

  end for
end procedure

Note that orientation of $\boldsymbol{g}$ essentially gives us orientation of the road along which vehicles travel, it does not give us the direction along that road. However, even without the direction, this information is oftentimes sufficient to disambiguate label assignment as shown in figure 7(a). When vehicles travel along the road in a checkerboard pattern, proximity based assignment will result in velocities which are perpendicular to $\boldsymbol{g}$. That is not the case when a number of vehicles are traveling in a linear formation as in Figure 7(b). Therefore, we introduce an additional formation context constraint (see figures 7(c) and 7(d)). If we are trying to match an object O_a 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

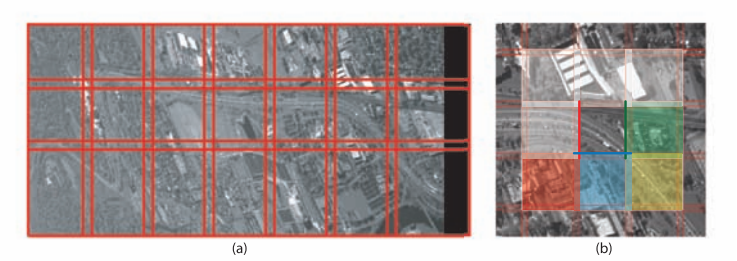

Fig. 8: (a) This figure shows an example frame with grid overlayed onto an image. (b) shows the grid cell search procedure for handing over tracks. The bold colored lines correspond to OLLeft, OLBottom, and OLRight, in counterclockwise direction. Only colored grid cells are searched, white cells are ignored.

equivalent, we make the kernel wrap around to other side of orientation portion of the histogram.

The road orientation constraint component is defined as

$$C_g = \frac{1}{2} + \frac{|\boldsymbol{g} \cdot \boldsymbol{v}^{t+1}|}{2\|\boldsymbol{g}\|\|\boldsymbol{v}^{t+1}\|} \tag{4}$$

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

$$C_c = \sum_{p}^{Nbins} \sum_{q}^{Mbins} min(\Phi_a^{p,q}, \Phi_b^{p,q}) \tag{5}$$

Finally, weight for edge of type 3 is computed as follows,

$$w = \alpha_1 C_g + \alpha_2 C_p + (1 - \alpha_1 - \alpha_2) C_c \tag{6}$$

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. Next, we find and link tracks that have crossed the cell boundaries, using **Algorithm 3** utilizing the overlapping regions of the neighboring grid cells. (see figure 8 for reference).

2.4 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. But, this approach has a serious issue which arises from the fact that 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

Algorithm 3 Algorithm for object handover across grid cells. The size of grid is m x n. $S(i,j)$ represents all tracks for the sequence in the cell at i^{th} row and j^{th} column in grid.

procedure INTERCELLHANDOVER
  for $i \leftarrow 1, m$ **do**
    for $j \leftarrow 1, n$ **do**
      Calculate OLLeft, OLRight and OLBottom ▷ See figure 8
      for all $s^{i,j} \in S(i,j)$ **do**

        if $\exists\ k \mid s_k^{i,j} > OLRight$ **then**
          completeTrack($s^{i,j}, S(i+1,j)$)
        else if $\exists\ k \mid s_k^{i,j} > OLRight\ \wedge \exists\ k \mid s_k^{i,j} > OLBottom$ **then**
          completeTrack($s^{i,j}, S(i+1,j+1)$)
        else if $\exists\ k \mid s_k^{i,j} > OLBottom$ **then**
          completeTrack($s^{i,j}, S(i,j+1)$)
        else if $\exists\ k \mid s_k^{i,j} < OLLeft\ \wedge \exists\ k \mid s_k^{i,j} > OLBottom$ **then**
          completeTrack($s^{i,j}, S(i-1,j+1)$)
        end if

      end for
    end for
  end for
end procedure

procedure COMPLETETRACK(s, S) ▷ s=track to complete, S=tracks in neighboring cell
  for all $s' \in S$ **do**
    if $\exists\ (l,m) \mid s_l.detectionID = s'_m.detectionID \wedge s_l.t = s'_m.t$ **then**
      assign s and s' unique label
    end if
  end for
end procedure

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 [1, 11, 12, 6], 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 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 [13] i.e.

$$I'_C(x,y) = \beta I_C(x,y)^{\gamma}, \tag{7}$$

Fig. 9: This figure shows the result of multi-camera intensity equalization. Notice the seam in image on left which in not visible in equalized image on right.

where $I_C(x, y)$ is the intensity of the original image at location (x, y). We find β, γ by minimizing the following cost function:

$$\underset{\beta,\gamma}{\operatorname{argmin}} \sum_{(x,y)\in I_{C1}\cap I_{C2}} (I_{C1}(x, y) - I'_{C2}(x, y))^2, \tag{8}$$

where $I_{C1} \cap I_{C2}$ is the overlap between the two cameras. The cost function is minimized using a trust region method for nonlinear minimization. The approximate Jacobian matrix is calculated by using finite difference derivatives of the cost function. Transformation in equation 7 is then applied to each frame of the camera before generating concurrent mosaics. Results for this procedure are shown in figure 9.

3 Results

We validated our method on four sequences from CLIF 2006 dataset. Sequences 1 to 3 are single camera sequences while sequence 4 has multiple cameras. The average number of objects in these sequences are approximately 2400, 1000, 1200 and 1100 respectively. Objects in sequence 2 and 3 undergo merging more often than objects in the other two sequences. This is primarily due to oblique angle between highway and camera in these sequences as opposed to top view in sequences 1 and 4. Figure 10 shows some of the tracks from these sequences.

For quantitative evaluation, we manually generated ground truth for the four sequences. Due the sheer number of objects, smaller size and similar appearance, generating ground truth for each object is a daunting task. We selected one region from sequence 1,3 and 4 and two regions from sequence 2 for ground truth. Objects were randomly selected and most of them undergo merging and splitting. The number of objects for which ground truth was generated are 34 for sequence 1, 47 and 60 for sequence 2 and 50 each for sequences 3 and 4.

Our method for evaluation is similar to [2] and measures performance of both detection and tracking. We compute the following distance measure between generated tracks and ground truth tracks:

$$D(T_a, G_b) = \frac{1}{|\Omega(T_a, G_b)|^2} \sum_{t\in\Omega(T_a,G_b)} \|\mathbf{x}_t^a - \mathbf{x}_t^b\|^2, \tag{9}$$

where $\Omega(T_a, G_b)$ denotes the temporal overlap between T_a and G_b, $|.|$ denotes cardinality while $\|.\|$ is the Euclidean norm. A set of pairs are associated i.e. $(a, b) \in A$ iff T_a and G_a have an overlap. The optimal association,

$$A^* = \underset{A}{\operatorname{argmin}} \sum_{(a,b)\in A} D(T_a, G_b) \text{ subject to } \Omega(T_a, T_c) = \emptyset \quad \forall (a, b), (c, b) \in A \tag{10}$$

is used to calculate the performance metrics. Abusing notation, we define

$$A(G_b) = \{T_a | (a, b) \in A\}. \tag{11}$$

The first metric *Object Detection Rate*, measures the quality of detections prior to any association:

$$ODR = \frac{\#\text{ correct detections}}{\#\text{ total detections in all frames}}. \tag{12}$$

We cannot compute ODR for each track and then average, because that would bias the metric towards short tracks as they are more likely to have all detections correct. Further notice that, it is not possible to detect false positives as the number of ground truth tracks is less than number of objects. A related metric, *Track Completeness Factor*,

$$TCF = \frac{\sum_a \sum_{T_b \in A(G_a)} |\Omega(T_b, G_a)|}{\sum_a |G_a|}, \tag{13}$$

measures how well we detect an object after association. TCF will always be less than or equal to ODR. The difference between ODR and TCF is the percentage of detections that were not included in tracks. Finally, *Track Fragmentation* measures how well we maintain identity of the track,

$$TF = \frac{\sum_a |A(G_a)|}{|\{G_a | A(G_a) \neq \emptyset\}|}. \tag{14}$$

Weighing the number of fragments in a track with length, we get *Normalized Track Fragmentation*,

$$NTF = \frac{\sum_a |G_a| \cdot |A(G_a)|}{\sum_{a|A(G_a)\neq\emptyset} |G_a|}. \tag{15}$$

which gives more weight to longer tracks as it is more difficult to maintain identity for long tracks than short ones.

We compare our method with the standard bipartite matching using greedy nearest-neighbor initialization. Initial assignment is done based on proximity while linear velocity model is used for prediction. Standard gating technique is used to eliminate unlikely candidates outside a certain radius. The same registration and detection methods were used for all experiments. The values of parameters for our tracking method were $\alpha = 0.5$ (eq. 3) and $\alpha_1 = \alpha_2 = 0.33$ (eq. 6). Table 1 shows the comparison between both methods:

Table 1: Quantitative Comparison

		Our Method			GreedyBIP		
	ODR	**TCF**	**TF**	**NTF**	**TCF**	**TF**	**NTF**
Seq1	0.975	0.716	2.471	2.506	0.361	13.06	13.11
Seq2	0.948	0.714	2.894	2.895	0.489	12.55	12.55
Seq3	0.972	0.727	2.76	2.759	0.583	8.527	8.53
Seq4	0.984	0.824	1.477	1.48	0.638	6.444	6.443

As can be seen from table 1, our method achieved better TCF and TF because unique characteristics of WAS demand the use of scene-based constraints which were not leveraged by the standard bipartite matching. We derived road orientation estimate and object context using only the image data, which allowed for better initialization and tracking performance.

4 Conclusion

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. These preliminary steps pave way for more in-depth exploitation of this data such as scene modeling and abnormal event detection.

Acknowledgments. This work was funded by Harris Corporation.

References

1. Javed, O., Shafique, K., Shah, M.: Appearance modeling for tracking in multiple non-overlapping cameras. In: CVPR. (2005)
2. Perera, A., Srinivas, C., Hoogs, A., Brooksby, G., Hu, W.: Multi-object tracking through simultaneous long occlusions and split-merge conditions. In: CVPR. (2006)
3. Ali, S., Shah, M.: Cocoa: tracking in aerial imagery. Volume 6209., SPIE (2006)
4. Xiao, J., Cheng, H., Han, F., Sawhney, H.: Geo-spatial aerial video processing for scene understanding and object tracking. In: CVPR. (2008)
5. Kang, J., Cohen, I., Yuan, C.: Detection and tracking of moving objects from a moving platform in presence of strong parallax. In: ICCV. (2005)
6. Arth, C., Leistner, C., Bischof, H.: Object reacquisition and tracking in large-scale smart camera networks. In: ICDSC. (2007)
7. Betke, M., Hirsh, D.E., Bagchi, A., Hristov, N.I., Makris, N.C., Kunz, T.H.: Tracking large variable numbers of objects in clutter. In: CVPR. (2007)
8. Wu, Z., Hristov, N.I., Hedrick, T.L., Kunz, T.H., Betke, M.: Tracking a large number of objects from multiple views. In: ICCV. (2009)

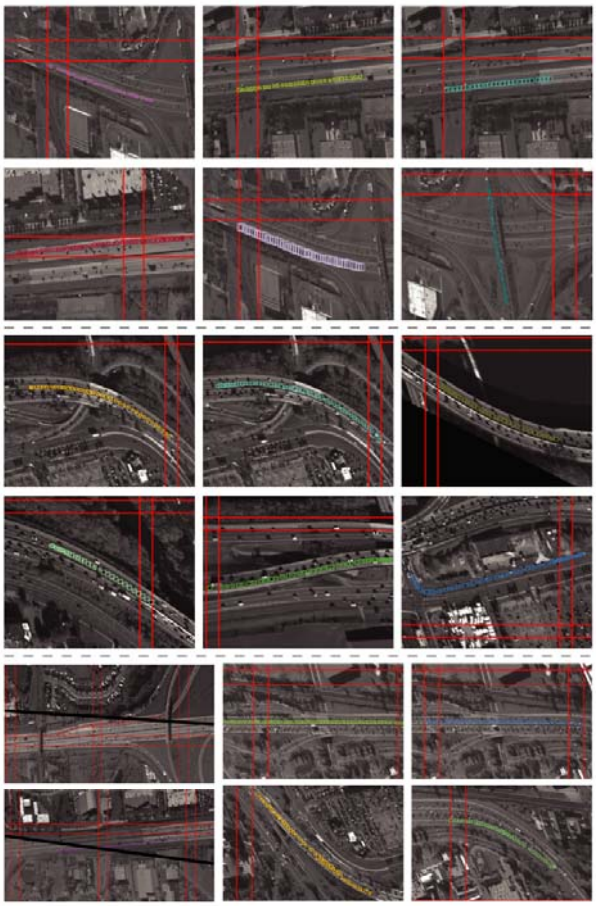

Fig. 10: This figure shows a number of results for different sequences. Top group is for sequence 1, second group is for sequence 2. In the bottom group, first column is from multiple camera sequence (camera boundary is shown in black), next two columns are from sequence 4.

9. Nguyen, H.T., Ji, Q., Smeulders, A.W.M.: Spatio-temporal context for robust multitarget tracking. IEEE PAMI (2007)
10. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. In: CVPR. (1999)
11. Pflugfelder, R., Bischof, H.: Tracking across non-overlapping views via geometry. In: ICPR. (2008)
12. Kaucic, R., Perera, A., Brooksby, G., Kaufhold, J., Hoogs, A.: A unified framework for tracking through occlusions and across sensor gaps. In: CVPR. (2005)
13. Farid, H.: Blind inverse gamma correction. IEEE Trans. Image Processing (2001)