Fusion of Local Appearance with Stereo Depth for Object Tracking

Feng Tang
University of California, Santa Cruz, California
HP Labs. Palo Alto, California
tang@soe.ucsc.edu

Michael Harville
CastTV Inc., San Francisco, California
HP Labs. Palo Alto, California
mike@casttv.com

Hai Tao
University of California, Santa Cruz, California
tao@soe.ucsc.edu

Ian N. Robinson
HP Labs. Palo Alto, California
ian.robinson@hp.com

Abstract

Object tracking methods based on stereo cameras, which provide both color and depth data at each pixel, find advantage in separating objects from each other and from background, determining the 3D size and location of objects, and modeling object shape. However, stereo tracking methods to date sometimes fail due to depth image noise, and discard much useful appearance information. We propose augmenting stereo-based models of tracked objects with sparse local appearance features, which have recently been applied with great success to object recognition under pose variation and partial occlusion. Depth data complements sparse local features by informing correct assignment of features to objects, while tracking of stable local appearance features helps overcome distortion of object shape models due to depth noise and partial occlusion. To speed up tracking of many local features, we also use a "binary Gabor" representation that is highly descriptive yet efficiently computed using integral images. In addition, a novel online feature selection and pruning technique is described to focus tracking onto the best localized and most consistent features. A tracking framework fusing all of these aspects is provided, and results for challenging video sequences are discussed. [1]

1. Introduction and related work

Object localization and tracking is an important problem in computer vision, with application in domains including surveillance, intelligent video conferencing, and human-computer interaction. To obtain good results, many significant challenges must be overcome, including segmentation of objects from cluttered backgrounds, discrimination of objects of interest from shadows and moving distractors, determination of 3D object location, and propagation of object identity assignments through close interactions or occlusions. These challenges grow when multiple cameras are needed to cover extended physical spaces. Many tracking methods based on monocular cameras have been proposed to address such problems, including techniques employing multiple simultaneous hypotheses of object location [4], competitive optimization of image pixel assignment to tracked layers [22], and Bayesian inference not only of object location but also of object number [10].

In recent years, substantial improvement in multi-object tracking performance has been enabled by the increasing practicality of real-time, dense (per-pixel) depth imagery from stereo cameras [1, 2]. The 3D information inherent in depth images affords better segmentation of objects from background, facilitates separation of objects undergoing close interaction and occlusion, enhances object models by including shape and scale as well as appearance, and simplifies estimation of 3D object location. While some methods have tracked objects directly in the depth images themselves [6], others have found advantage in first projecting the 3D scene onto the ground plane and then tracking in the resulting "plan-view" images [5, 23]. Because objects generally do not overlap much in the dimension normal to the ground plane in which they move, they are more easily separated and tracked in plan-view images than in the original "camera-view" images. 3D data allows plan-view object representations to be made invariant to object scale in camera-view, thus simplifying object model dynamics. Furthermore, projection of data from multiple stereo cameras into a single plan-view coordinate system, as in [5, 23], provides a straightforward means of integrating object tracks across large spaces covered by multiple cameras.

Despite its advantages, depth data produced by typical real-time stereo implementations is much noisier than the gray-scale or color imagery from which it is computed, and contains many pixels whose values have low confidence due

[1] Work done when Feng Tang and Michael Harville were in HP Labs.

978-1-4244-2340-8/08/$25.00 ©2008 IEEE

to stereo matching ambiguity at textureless regions or depth discontinuities. These problems increase with the distance of objects from the camera, and make it difficult to create depth-based models of tracked objects and features that are stable over long time periods. To address this, a few stereo-based tracking methods have added appearance components to their object models. In most of these, object appearance is modeled simply with color "blobs" or histograms. In [9], a plan-view image template is colored according to the highest pixel above ground at each plan-view location, resulting in an approximate rendering of tracked objects as if viewed from above. However, this representation depends heavily on the quality of the depth data, and hence is greatly affected by typical depth noise.

We wish to make better use of appearance information in stereo-based tracking. Much progress has recently occurred in development of invariant descriptors of local features [11, 13], and in applying these to object recognition. By modeling an object with a set of sparse local features, new observations of the object can be correctly identified under widely varying pose and scale, or during partial occlusion. Such superior recognition performance suggests that similar, feature-based object models might enable good tracking of objects under the same circumstances. Clustering of trajectories of feature motion has been applied to counting (but not tracking) independently moving objects in crowds [3, 15]. In [19], sparse local features are organized in the form of a graph for object tracking. In [14]'s feature harvesting work, the authors use randomized tree to classify the keypoints for rigid object detection and tracking.

In this paper, we combine plan-view projections of depth imagery with feature-based object appearance models to create a robust, real-time multi-object tracking system. The system architecture is shown in Figure 1. A fusion based tracking scheme is employed, in which many features are tracked individually in the camera-view, with the resulting trajectories being projected into plan-view along with dense foreground depth to create a substrate for multi-object tracking. Depth data and sparse local features prove highly complementary in a tracking context. By placing features in 3D, and in particular by projecting their locations into plan-view maps of dense depth data, a sound basis for assigning features to objects is obtained. At the same time, by enhancing depth-based object shape models with descriptions of associated salient appearance features, greater invariance of object representation to depth noise and partial occlusion is achieved. For example, if one or more appearance features can be tracked in the camera view through an occlusion, the identity of the associated object is more likely maintained through this occlusion. Local features also allow us to begin to exploit the recent progress in feature-based object recognition for linking identities of tracked objects across different camera views, complete occlusions, and scene exit and re-entry. Furthermore, if one feature fails, the system might still be able to rely on the other feature. For example, if an object cannot find enough appearance features for tracking, the depth information can still keep track.

To our knowledge, we describe here the first multi-object tracking method to use both dense depth imagery and local appearance features. Further, feature-based tracking is a popular and challenging area in computer vision, and we present two additional innovation improving the speed and robustness of such methods, whether or not they employ stereo. First, we use fast local features called "binary Gabors", in which the convolutions of a set of Gabor filters with an image location are approximated as sums of binary box features. In combination with integral images, this greatly speeds feature tracking in a search-based framework, while retaining good descriptive power. Second, we present an adaptive feature selection scheme that maintains only a few of the "best" features for each object model at each time step. Features are selected based on a combination of their trajectory properties and their distinctiveness relative to their surroundings.

In Section 2, we discuss building object representations from plan-view maps and appearance features, and from binary Gabor features in particular. Methods for online selection of reliable features for tracking are discussed in Section 3. The probabilistic tracking framework combining plan-view and appearance information is presented in Section 4. Experimental results are shown in Section 5, and we conclude in Section 6.

2. Object representation

As shown in Figure 1, the input to our algorithm is a video stream of color and registered depth data captured by a stereo camera. Depth is provided on a per-pixel basis, although some pixels may be marked as having "unknown" or low-confidence values. A background modeling module, such as in [18], is used to extract foreground areas most likely to contain objects of interest. With calibrated cameras, each foreground pixel with reliable depth can be projected to its location on a plan-view ground plane. Shape descriptors for tracking are built in this plan-view space as described in Section 2.1. At the same time, in the foreground areas of the "camera-view" images, local appearance features are selected and described as in Section 2.2.

2.1. Shape descriptors from stereo

Prior calibration of the stereo unit provides its location and orientation in a 3D world coordinate system, in which the XY-plane is aligned with the ground, and Z represents height above ground. Back-projection of foreground image pixels creates a point cloud in 3D space, which may be partitioned into Z-aligned vertical bins. We typically use bins intersecting the XY plane in a square grid, with 2-4cm ex-

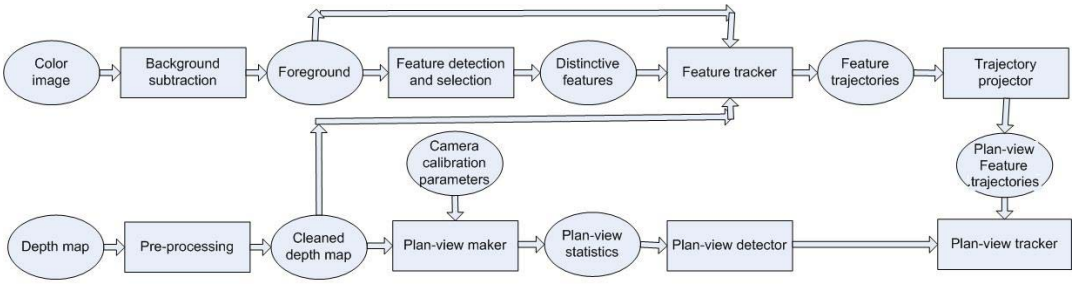


Figure 1. System architecture.

tent. A plan-view image, or map, contains a pixel for each vertical bin, with the value at each pixel being some statistic of the 3D points in the bin.

We create plan-view maps of two statistics, namely "height" and "occupancy". The height map contains the height of the highest point in each vertical bin that is above ground level and below a reasonable maximum height t which to find objects of interest. An occupancy map displays distance-weighted counts of the points in each bin, where the weights effectively convert the counts to estimates of the physical surface area (in cm^2) of foreground objects visible to the camera within each vertical bin [7]. Both maps may be computed together efficiently in a single pass through the input depth image. Plan-view image values based on too few 3D points are pruned from the plan-view maps, to avoid unreliable statistics. Examples of camera-view color, depth, and foreground images are shown in Figures 2(a-c), and the corresponding plan-view occupancy map is shown in Figure 2(d).

Occupancy maps quantify the "amount" of foreground data at each floor location, while height maps describe their shape as viewed from above. Their combination helps compensate for errors in foreground extraction, by allowing easy discimination of shadows or incorrectly sized objects. Furthermore, the appearance of objects in these plan-view maps is invariant to object distance from the camera, except for depth noise. Stereo cameras allow construction of these maps without actually mounting cameras overhead, although the maps will describe only the portions of objects visible to the camera.

To model tracked objects, we use templates (small image patches) extracted from the height and occupancy maps at the estimated plan-view object locations. These non-parametric representations provide detailed object models that are well suited to short-term tracking, but that are too weak to enable reliable object recognition across changes in object pose and occlusion. Other methods have modeled object shape using plan-view blobs [5] and Gaussians [23].

2.2. Appearance features

Local features are extracted in the color imagery to complement the shape descriptors extracted in plan-view. A local feature consists of two components: 1) an image location determined by a feature detector, and 2) a descriptor of the local image appearance in a small neighborhood about the detection location. We use the FAST (Features from Accelerated Segment Test) [16] feature detector to determine feature locations in a highly efficient manner. For a descriptor, we would like to use the responses of a set of Gabor filters at the detection location. Such descriptors have been applied with much success to many image analysis and pattern recognition tasks, but are computed with convolution operations entailing many element-wise floating point multiplications. The search process employed during feature tracking will require these same convolutions to be applied at many image locations, which can be very computationally expensive.

(a) (b) (c) (d)

Figure 2. Example imagery: (a) input color, (b) input depth registered with (a); (c) extracted foreground; (d) plan-view occupancy map showing five foreground objects.

To speed up this process, we use a non-orthogonal binary expansion of Gabor filters [20] for feature descriptor extraction. This is inspired by the recent work of [21], in which general signals are represented to arbitrary precision with linear combinations of basis vectors in a non-orthogonal binary subspace (NBS). In our case, we represent each Gabor filter as a linear combination of Haar-like box feature. Box features selected to represent an example Gabor filter are shown in Fig. 3. The main advantage of this NBS expansion is that, after performing an integral image transform on the input image, the dot product between each box base vectors and a particular image location can be computed very efficiently using a few integer additions. In this paper, we use a binary Gabor filter bank with 4 scales and 6 orientations (24 filters), applied only to the luminance channel.

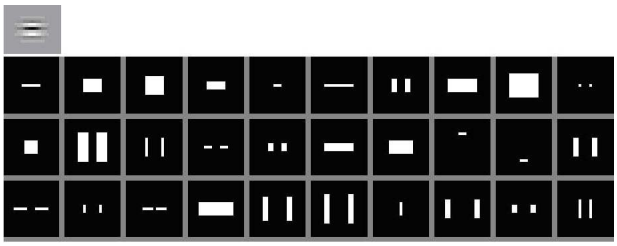


Figure 3. Non-orthogonal binary expansion of a Gabor filter for fast feature extraction.

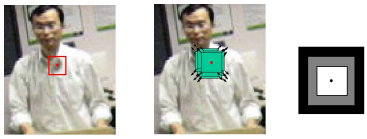
3. Feature tracking

Many current local feature based tracking methods e.g. [19, 12] employ a "feature match" framework with a two-stage solution: first, features (e.g. SIFT [13]) are detected in each individual frame; second, features are matched between two successive frames to obtain a one-to-one feature correspondence. The motion vector of the object is then computed as an integration of the local feature motion vectors. The main problem with this framework is that it relies on a "repeatable" feature detector. However, in many real scenarios, features may not be persistently detected in every frame due to image noise or object pose changes. Even for two adjacent frames with small motion, some features detected in one frame may not be detected in the next. Reliable matching between two sets of features with missing data remains a very hard problem in graph theory. Such methods usually involve complicated optimization procedures like relaxation labelling with dummy features [19], which are often slow and unstable. In this paper, we advocate use of a "feature search" tracking framework that can bypass the difficult matching problem. For each feature associated with an object in the previous frame, we search within a local window around the feature's predicted image location in the next frame for the image data most similar to the feature's appearance. Similarity is assessed as Euclidean distance between the binary Gabor coefficients representing the feature and those obtained at a camera image location. In the integrated object tracking framework of Section 4.3, these similarity scores are transformed to probabilities via a sigmoid function.

3.1. Feature selection

In general, many feature points are detected on each tracked object. It is not prudent to attempt to track all of them, not only because of computational cost, but also because some features are similar to their neighbors, leading to spurious matching and feature tracking drift. In [17], the authors select highly textured image regions as good features for tracking. In this paper, we propose a novel feature selection scheme targeted for tracking contexts employing sparse feature representations of objects. The basic intuition is that we wish to select features that are dissimilar to their image surroundings, and that move in a smooth, coherent fashion with others associated with the same object. Feature selection relies on four components:

Distinctiveness η_d: We consider a feature that is dissimilar to its surroundings to be highly "distinctive", in that it will likely be well localized during search in successive frames. To assess "distinctiveness", we compare the descriptor for each newly detected feature with those computed at image locations in a surrounding neighborhood. Ideally, the difference between the feature descriptors at the detection and neighboring locations increases rapidly with image distance. We divide the feature neighborhood into three regions (1=white, 2=gray, and 3=black in Fig. 4), and compute the differences between descriptors in each neighborhood and that at the feature detection location in the center. Denoting the minimal descriptor difference for region i as min_i, we define the distinctiveness of feature f as $\eta_d(f) = \frac{min_3 - min_1}{min_2 - min_1}$. If descriptor difference increases with image distance, $min_3 > min_2 > min_1$, and $\eta_d(f)$ is high.



(a) (b) (c)

Figure 4. Feature distinctiveness. (a) feature on a tracked person, with box indicating extent of feature descriptor; (b) neighboring (overlapping) windows used to compute distinctiveness; (c) 3 regions around detection location used to compute distinctiveness (see the text).

Matching error η_e: Let e_t be the difference (error) between a feature's appearance descriptor and the most similar image location found for it during search-based tracking, as in Section 3. If a feature has small matching error over the previous several frames, it has high confidence to predict the correct location in the current frame. We use a function of the average matching error over the previous n frames to help select features for tracking: $\eta_e = \exp(-[\frac{1}{n}\sum_{t=1}^{n^2} e_t]^2/(2\sigma^2))$, where the variance σ^2 of the matching error is determined empirically.

Trajectory smoothness η_s: At high system frame rates, it is reasonable to assume the features are moving continuously (smooth angular velocity and magnitude change), so that non-smoothness is likely caused by drifting. We denote the trajectory for feature j of object i as T_j^i, the kth point on the trajectory as $T_j^i(k)$, and the velocity vector at this point as $TV_j^i(k)$. We then define η_s as:

$$\eta_s = w[\frac{1}{m}\sum_k \frac{TV_j^i(k)\cdot TV_j^i(k-1)}{||TV_j^i(k)||\cdot||TV_j^i(k-1)||}]+ \\ (1-w)[\frac{1}{m}\sum_k \frac{2\sqrt{TV_j^i(k)\cdot TV_j^i(k-1)}}{||TV_j^i(k)||+||TV_j^i(k-1)||}] \quad (1)$$

The first term penalizes velocity direction change, where this change is measured as the cosine of the two consecutive feature velocity vectors. The second term penalizes velocity magnitude change. The weight w balances the relative importance of these two terms.

Trajectory congruency η_c: At a coarse level, trajectories of features associated with the same object should be more similar than those of features associated with different objects. We define a distance measure between two trajectories as:

$$d(T_j^i, T_k^i) = \frac{1}{N}\sum_{n=1}^{N}[T_j^i(n) - T_k^i(n) - mean(T_j^i, T_k^i)] \quad (2)$$

where $mean(T_j^i, T_k^i) = \frac{1}{N}\sum_{n=1}^{N}[T_j^i(n) - T_k^i(n)]$ is the mean Euclidean distance between the two trajectories. This is effectively the variance of distance of corresponding trajectory points over the time during which the trajectories overlap. If two trajectories are congruent, this variance should be small. It is zero if two trajectories are related by a simple translation in space. For a given feature, trajectory "congruency" η_c measures how similar its trajectory is to that of all other features associated with the same object:

$$\eta_s = exp(\frac{d_{avg} - d_{min}}{d_{max} - d_{avg} + \epsilon} - 1) \quad (3)$$

d_{avg} is the average trajectory distance between T_j^i and its neighbor features, d_{min} and d_{max} are the minimal and maximal of the trajectory distances. Trajectory congruency is near 1 if T_j^i is translationally related with all its neighbors, and it is low if the feature trajectories are dramatically different. ϵ is a small number to avoid division by zero in the case of perfect congruency. A similar technique has been used in [3] for the purpose of feature clustering.

During initialization of tracking of a new object, we compute the distinctiveness η_d for all detected foreground features and select only the top k most distinctive for tracking. Fig. 5 shows an example of feature selection. The selected features can usually be tracked until a significant pose, lighting, or occlusion change occurs.

During tracking, a confidence measure c is computed for each feature as the product $\eta_e\eta_s\eta_c$ of matching error, trajectory smoothness, and trajectory congruency. Features are deleted from tracking when this confidence falls below a threshold. When the number of active features for an object is less than a threshold N_{min}, feature detection is reapplied, and the top k most distinctive features are selected as the new feature set. To aid consistent object tracking, we constrain the new feature set to contain at least one old feature that was already being tracked on this object. Together with the feature search framework, the feature selection scheme makes tracking both reliable and efficient.

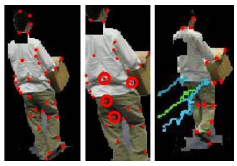


Figure 5. Feature selection example. Left: image of a person in which each red dot is a detected feature. Middle: zoomed image with features selected according to "distinctiveness" circled. Right: trajectories of the tracked selected features.

4. Tracking framework

The tracking method uses a dynamic Bayesian network to estimate the best joint probability of the multi-object state sequence and observation sequence. The hidden state includes information about the plan-view object location, velocity, plan-view shape, and visual appearance of each object. The multiple object configuration at time t is denoted as $X_t = \{x_t^1, ..., x_t^N\}$. The object i at time t is represented as $x_t^i = \{p_t^i, v_t^i, \mathcal{O}_t^i, \mathcal{H}_t^i, \mathcal{F}_t^i\}$, where p_t^i is the object location in the plan-view, v_t^i is the object velocity, $\mathcal{O}_t^i$ and $\mathcal{H}_t^i$ are the plan-view occupancy and height templates for object i, and $\mathcal{F}_t^i = \{f_{i,t}^1, ..., f_{i,t}^{K_i}\}$ is the set of local features associated with object i. Each local visual feature $f_{i,t}^j$ is represented by its location $p_{i,t}^j$ in the camera view, its depth $d_{i,t}^j$, and a binary Gabor camera-view descriptor $g_{i,t}^j$ as described in Section 2.2. Given measurement data Z_t derived from the input imagery observed at time t, we attempt to find the maximum *a posteriori* (MAP) multiple-object configuration X_t according to:

$$P(X_t|Z_t) \propto P(Z_t|X_t)P(X_t) \quad (4)$$

4.1. Prior state probability

The prior distribution $P(X_t)$ is determined from prediction applied to the estimated previous multi-object configuration state. In our tracking framework, the object has a two-level representation: 1) the object shape (as described in the plan-view maps), and 2) the sparse local appearance features. As described in section 1, both methods have complementary strengths and weaknesses, so we propose a joint prior formulation which incorporates both planview shape and local visual information.

Let $A(t)$ be the set of the indices of the active objects, i.e., the objects that appear in the view, at time t. Assuming independent movement of each object, $P(X_t|X_{t-1})$ can be factored as:

$$P(X_t|X_{t-1}) \propto \prod_{i \in A(t) \bigcap A(t-1)} P(x_t^i|x_{t-1}^i) \quad (5)$$

where

$$P(x_t^i|x_{t-1}^i) = P(p_t^i|p_{t-1}^i)P(\mathcal{O}_t^i|\mathcal{O}_{t-1}^i)P(\mathcal{H}_t^i|\mathcal{H}_{t-1}^i) \quad (6)$$

The state transition dynamics $P(x_t^i|x_{t-1}^i)$ describe how object location and appearance change in the next frame. At high system frame rates, it is reasonable to predict that the plan-view shape descriptors and local feature appearance components of the state can be well predicted as remaining constant across successive frames. For simplicity, the 3 terms in Eq. 6 are assumed to be Gaussians. We predict object location, however, based not only on the motion of the plan-view data, but also on the motion of individual appearance features. Specifically, the object location prior $P(p_t^i|p_{t-1}^i)$ is a Gaussian centered at a predicted plan-view object location μ_t, which in turn is a linear combination of the shape template motion prediction $\mu_{s,t-1}^i$ and the plan-view projections of the predicted locations $\mu_{j,t-1}^i$ of tracked appearance features f_j^i:

$$\mu_t^i = \alpha \times \mu_{s,t-1}^i + (1-\alpha) \times \frac{1}{N_i} \sum_{j=1}^{N_i} c_{j,t-1}^i \mu_{j,t-1}^i, \quad (7)$$

All of these predicted locations are based on a constant velocity model. The $c_{j,t-1}^i$ are the feature confidence factors described in Section 3.1

4.2. Plan-view object detection

After the plan-view maps are built as in Section 2.1, we compute "object detection" measurements by looking for significant "piles of pixels" in the occupancy map. More precisely, we convolve the occupancy map with a box filter and find the maximum value of the result. If this peak value is above a threshold θ_{occ}, we regard its location as that of a candidate object. The box filter size is a physically-motivated parameter, with width and height equal to an estimate of twice the average object size. This detection process can be done very quickly by using an integral image, and produces a set of potential object locations denoted as Z_t^d. Detection is performed not only to initialize tracking, but also on every frame to aid in determining candidate locations of objects already being tracked. Similarly, the trajectory tracker [8] also uses detection result as a measurement for tracking, but their detector is a pre-trained camera-view person head detector.

4.3. Likelihood

The likelihood $P(Z_t|X_t)$ (second term in Eq. 4) measures how likely the observations agree with the hypothesized object state. The shape, appearance, and plan-view object detection observations are denoted as $Z_t = \{Z_t^s, Z_t^a, Z_t^d\}$. Assuming independence between the observation modalities, the likelihood term can be decomposed as:

$$P(Z_t|X_t) = \prod_i P(Z_t|x_i) = \prod_i P(Z_t^s|x_i) P(Z_t^a|x_i) P(Z_t^d|x_i) \quad (8)$$

$P(Z_t^s|x_i)$ is the likelihood of the stereo plan-view map features given the hypothesis that object i is at location p_i with the height map as $\mathcal{H}_t^i$ and occupancy map as $\mathcal{F}_t^i$. These are evaluated by first computing the "sum of absolute differences" (SADs) of $\mathcal{H}_t^i$ and $\mathcal{F}_t^i$ with the plan-view data at location p_i, and then transforming the SADs to likelihoods via sigmoidal functions fitted to training data.

The appearance feature likelihood term $P(Z_t^a|x_i)$ is computed as the product of each feature likelihood:

$$P(Z_t^a|x_i) = \prod_j P(\widehat{f}_j|f_j) \quad (9)$$

$P(\widehat{f}_j|f_j)$ is the likelihood that feature j is observed as $\widehat{f}_j$. It is computed as a sigmoid function of the Euclidean distances between binary Gabor model features and the observed features in the camera view.

The detection likelihood $P(Z_t^d|x_i)$ represents the support of the object state prediction from the plan-view object detection described in Section 4.1:

$$P(Z_t^d|x_i) = \prod_j P(Z_t^{d,j}|x_i) \prod_{k \neq i} (1 - P(Z_t^{d,j}|x_k)) \quad (10)$$

The first term is the likelihood that an object detected at a given location is object i. It is assumed a Gaussian distribution of the distance between the predicted location and the detected location. The second term is an "exclusion" term that discourages estimation of different objects as being at the same location.

The MAP multi-object state estimate is determined by computing the priors and measurement likelihoods of Eq. 4-Eq. 10 at all locations within local windows around predicted object locations. The maximum MAP location for each object is selected independently, in sequence according to the duration of time the object has been tracked, and is used as the location x_t^i for that object in the current frame. Plan-view template state $\mathcal{O}_t^i$ and $\mathcal{H}_t^i$ is updated for each object by replacement with the plan-view image data at the new object locations. Feature appearance state $\mathcal{F}_t^i$ is updated similarly using camera image data at the estimated locations returned from the feature search process of Section 3 and Eq. 9.

5. Experiments

An implementation of our system runs at 15 Hz on average on a dual-processor 2.8GHz PC, with the color-with-depth video being provided by a Point Grey Triclops camera. We evaluated our method on long sequences (more than 1000 frames) captured at 12-15Hz and 320x240 resolution, with the camera typically mounted with a view like that of Figure 6.

One application of interest is tracking people and items (boxes in our sequence), where those items are carried, deposited and picked up by the people. This scenario can

be applied, for example, to applications involving loading docks at warehouses or baggage handling at airports. In our algorithm a person entering the scene with a box is detected and tracked as a single object. The number of plan-view detections around object tracks is continuously checked to detect a "splitting" event caused by a person depositing a box. If there is a new detection very close to an existing track, it is identified as a child of the original object and a splitting event is triggered. Stationary objects continue to be tracked (i.e. they are not absorbed into the background). A "merge" event occurs when a stationary object ceases to be detected coincident with the presence of another moving tracked object, corresponding to a person picking up an object. Identities can be assigned to the tracked objects depending on the needs of the application, e.g. whether tracking people or items is the priority. In the sequences shown the identities of the moving objects - people - are maintained, whilst new identities are generated for the deposited boxes. These new identities persist until the object is picked up.

The sequences contain objects of different types, including people, boxes, under challenging occlusion events, close interactions, and varying illumination. Since the camera is fixed, background modeling is used to detect moving objects and foreground pixels are used to generate stereo depth and initialize the object of interest.

In Figure 6, one person enters the scene with a box and drops it on the ground, while a second person enters with a second box before the first person exits. After crossing each other, the second person drops his box on the ground and the first person walks into the scene again. Despite the complex interactions, occlusions, and non-rigid motions, our tracker can robustly track all of the objects.

Figure 7 is another example of tracking multiple objects under complex occlusion and interactions. Most of the time, there are 3-4 people walking around in a small area and they are occluding each other, doing different actions. However, as can be observed in the plan-view image on the third row, all these objects are successfully identified and tracked. Good performance was consistently obtained in our test sequences.

6. Conclusions

We have presented a tracking system that combines local appearance features and stereo depth data for robust multiple object tracking. The combination of these two types of information nicely compensates for the weaknesses of each individually, and provide a rich enough substrate that relatively straightforward, real-time tracking algorithms can perform well. We have also presented methodologies for feature representation and selection that apply to multi-object tracking contexts independent of stereo data. We are working to compare our tracker's performance to that of related methods, and to design new algorithms for detecting addition and removal of objects from scenes. In the future, we also plan to experiment with different types of features, with the aim of extending our system to perform object recognition and pose estimation.

References

[1] http://www.canesta.com. 1

[2] http://www.tyzx.com. 1

[3] G. Brostow and R. Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *IEEE Computer Vision and Pattern Recognition*, pages I: 594–601, 2006. 2, 5

[4] I. Cox and S. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *PAMI*, 18(2):138–150, Feburary 1996. 1

[5] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb. Plan-view trajectory estimation with dense stereo background models. In *ICCV*, pages 628–635, 2001. 1, 3

[6] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *IJCV*, 37(2):175–185, June 2000. 1

[7] M. Han, W. Xu, H. Tao, and Y. Gong. An algorithm for multiple object trajectory tracking. In *CVPR*, pages (I) 864–871, 2004. 3

[8] M. Han, W. Xu, H. Tao, and Y. Gong. An algorithm for multiple object trajectory tracking. In *CVPR*, pages (I) 864–871, 2004. 6

[9] M. Harville. Stereo person tracking with short and long term plan-view appearance models of shape and color. In *IEEE International Conference on Advanced Video and Signal based Surveillance*, pages 522–527, 2005. 2

[10] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *ICCV*, pages II 34–41, 2001. 1

[11] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *CVPR*, pages II: 90–96, 2004. 2

[12] M. Leordeanu and R. Collins. Unsupervised learning of object features from video sequences. In *CVPR*, pages 1142–1149, 2005. 4

[13] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. 2, 4

[14] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *ECCV06*, volume 3953, pages 592–605, 2006. 2

[15] V. Rabaud and S. Belongie. Counting crowded moving objects. In *CVPR (1)*, pages 705–711, 2006. 2

[16] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006. 3

[17] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, June 1994. 4

[18] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, pages (II) 246–252, 1999. 2

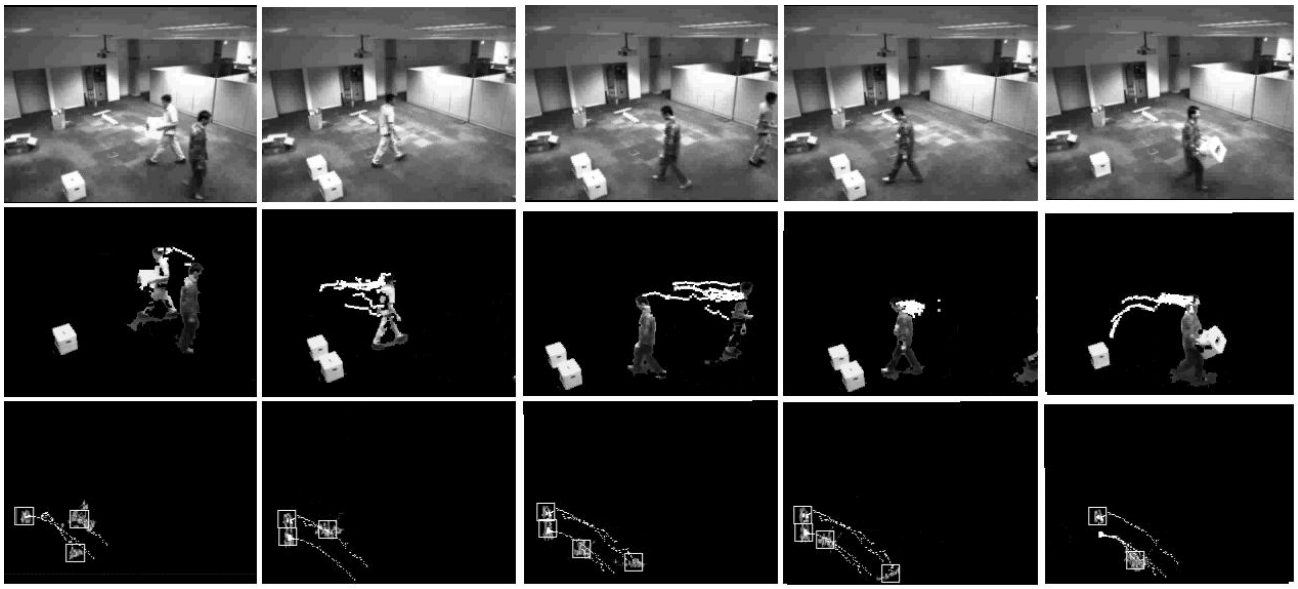


Figure 6. Tracking results. First row: the foreground image; Second row: the feature tracking result on the intensity image; Third row: the plan-view tracking result.

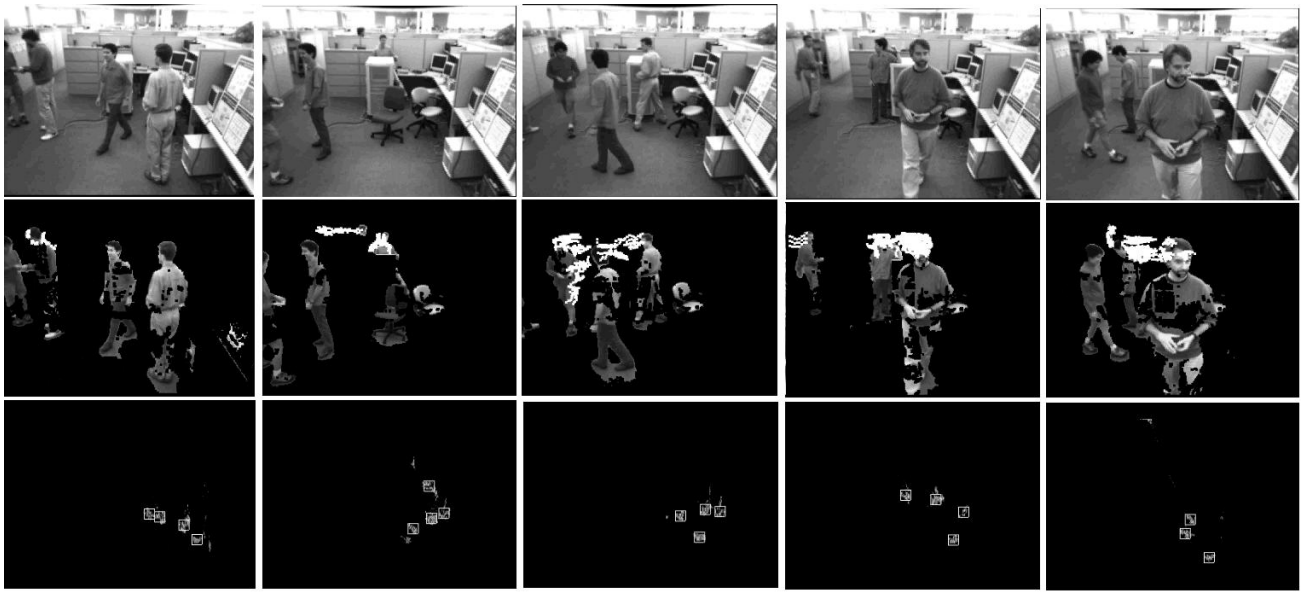


Figure 7. Tracking results. First row: the foreground image; Second row: the feature tracking result on the intensity image; Third row: the plan-view tracking result. Note: In order not to let the complicated interaction and long trajectories mess up the viewing, trajectories are not shown in the figure.

[19] F. Tang and H. Tao. Object tracking with dynamic feature graph. In *VS-PETS*, pages 25– 32, 2005. 2, 4

[20] F. Tang and H. Tao. Non-orthogonal binary expansion of gabor filters with the application in object tracking. In *IEEE Workshop on Motion and Video Computing*, 2007. 3

[21] H. Tao, R. Crabb, and F. Tang. Non-orthogonal binary subspace and its applications in computer vision. In *ICCV*, pages (I) 864– 870, 2005. 3

[22] H. Tao, H. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *PAMI*, 24(1):75–89, January 2002. 1

[23] T. Zhao, M. Aggarwal, R. Kumar, and H. Sawhney. Real-time wide area multi-camera stereo tracking. In *CVPR*, pages (I) 976–983, 2005. 1, 3