OSU Dept. of Computer Science and Engineering Technical Report OSU-CISRC-12/06-TR79

A Method for 3D Tracking Using Multiple Cameras

Ambrish Tyagi[1], Mark Keck[1], James W. Davis[1], Gerasimos Potamianos[2]
[1] Dept. of Computer Science and Engineering, Ohio State University, Colombus, OH 43210
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
Emails: {tyagia,keck,jwdavis}@cse.ohio-state.edu, gpotam@us.ibm.com

Abstract

We present a computer vision system to robustly track an object in 3D by combining evidence from multiple calibrated cameras. Its novelty lies in the proposed unified approach to 3D kernel based tracking, that amounts to fusing the appearance features from all available camera sensors, as opposed to tracking the object appearance in the individual 2D views and fusing the results. The elegance of the method resides in its inherent ability to handle problems encountered by various 2D trackers, including scale selection, occlusion, view-dependence, and correspondence across different views. We apply the method on the CHIL project database for tracking the presenter's head during lectures inside smart rooms equipped with four calibrated cameras. As compared to traditional 2D based mean shift tracking approaches, the proposed algorithm results in 35% relative reduction in overall 3D tracking error and a 70% reduction in the number of tracker re-initializations.

1. Introduction

A complex scene is characterized by several moving objects such as people, animals, vehicles, etc. In order to perform higher level tasks, as demanded by typical surveillance or monitoring applications, one needs to first identify and track various objects of interest. Although the human visual system is extremely capable and efficient in temporally associating, or *tracking*, objects, the task has proven surprisingly difficult in computer vision.

Traditionally, tracking algorithms are based on filtering and data association techniques, ranging from simple Kalman filter models [2] to more complicated particle filters [7] and their variants. More recently, kernel based techniques for object tracking have also gained popularity. In particular, mean shift tracking has attracted attention in the vision community due to its computational efficiency and ease of implementation [5].

Most of these tracking algorithms have only been applied to the two-dimensional (2D) image plane. However, as multiple camera systems and data are becoming more affordable and available [14, 9, 3], the interest of the community has expanded into tracking the position of an object through the three-dimensional (3D) space. A typical approach to attack this problem is to combine results from 2D trackers in order to obtain 3D information by exploiting geometric constraints between the camera views [15, 16]. Such a framework however suffers from the typical 2D tracking limitations, namely scale selection, occlusion, view-dependence, and correspondence across the available views.

In this paper, we present a novel approach to the 3D tracking problem, by tracking the desirable object directly in the 3D space instead. The proposed method is inherently capable of addressing the aforementioned limitations of 2D tracking algorithms. This is achieved by exploiting the available 2D camera views through an appearance based feature level fusion approach, instead of traditional, decision level fusion. In particular, the proposed technique is kernel based, and it constitutes a 3D generalization of the 2D mean shift tracking algorithm [5], by employing

a joint histogram representation of the 3D object, fused over its 2D appearance features resulting from the object projection onto the available camera views. Naturally, the proposed framework is indifferent to scale variation in the 2D camera views, as well as view-independent and robust to occlusions, assuming that the object is non-deformable and that enough camera views are available.

We further strengthen the above claims by applying the proposed approach to 3D head tracking inside smart rooms. The task is central to the lecture scenario within the EU-funded CHIL project ("Computers in the Human Interaction Loop"), where a presenter interacts with meeting participants inside smart rooms equipped with a number of sensors, including four calibrated cameras located at the room corners. In particular, we demonstrate the superior performance of the proposed kernel based 3D tracking algorithm, as compared to a baseline that extracts 3D information by combining 2D tracking results from the available cameras views. Notice that since the paper does not address object track initialization and drift detection, but rather the tracking component alone, an appropriate experimental paradigm is adopted to allow the necessary algorithmic comparisons.

The remainder of the paper is organized as follows: We begin with a brief overview of related work on single- and multiple-camera object tracking in Section 2. The object tracking framework, including the baseline system and the proposed 3D algorithm, are described in Section 3. Subsequently, we present extensive experimental evaluation of these algorithms in Section 4. Finally, we summarize the paper in Section 5.

2. Related Work

Filtering and data association methods for tracking have been used previously. Kalman filter based tracking assumes that the object being tracked is driven by a linear process with a Gaussian state distribution [2]. The object location is optimally predicted, and new observations are assimilated during the tracking process. The Extended Kalman filter can be used to find a suboptimal solution for a non-linear process, by approximating it as a linear process [11]. A more general particle filtering based tracking framework was proposed by Isard and Blake [7].

More recently, kernel based object tracking approaches have gained popularity. Mean shift object tracking has been widely used in the computer vision community since it was proposed in [5]. It has been successfully employed to track non-rigid objects in 2D images in the presence of clutter and occlusions. The method is popular due to its succinct description and real-time implementation. Kernel based tracking approaches do not assume any knowledge of shape or motion model of the object. Various enhancements to handle particular problems arising from this algorithm such as scale selection [4], feature fusion [13], etc. have been proposed. Mean shift tracking and particle filters were used together for hand tracking in [12].

All of the aforementioned algorithms were developed with the intent of tracking objects in 2D images. As previously discussed, the focus has been recently shifting to 3D object tracking. Multiple camera systems can make use of redundant information for resolving occlusions to provide 3D information about the objects and the scene.

Systems such as [15, 16] have recently attempted to fuse information from different sensors to obtain 3D tracking results. In [16], shape features are extracted from blobs in each camera view and they are associated from frame to frame. The individual tracks from each view are then fused into 3D tracks using an Extended Kalman filter. Zhang [15] presented a 2D color based mean shift tracking system that was bootstrapped by using motion cues and face detection. The 2D detection results were combined using the camera calibration information to obtain a 3D output. Other approaches such as [1, 8] make assumptions about the scene's planar structure in order to obtain object correspondence across multiple views and finally use Kalman filters or clustering to track objects.

In all the above cases, tracker performance depends on the limitations of 2D tracking approaches or the assumptions made by these systems (such as a planar scene structure). Also, in the past, [17, 10] have applied particle filters in 3D for joint audio-visual person tracking, but these systems lack the advantages of a kernel based tracking approach, as discussed earlier.

In this paper, we present a unified 3D kernel based object tracking framework that employs feature fusion to combine information from different sensors in a robust and consistent manner, while avoiding the problems encountered by 2D tracking algorithms. Results are presented on a large dataset of interactive video sequences and compared to a baseline 2D tracking approach.

3. Two Kernel-Based 3D Tracking Approaches

Tracking is a special case of image registration, where the given object template (feature description) is searched for in the target image. The search space is limited, due to the assumption that the object has a continuous motion trajectory, and it is generally expected to be found in the vicinity of its previous location. The search space can be spatially smoothed by masking the target with an isotropic kernel, and hence any gradient based search technique, such as the mean shift mode seeking algorithm, can be employed for optimization.

In the following, we describe in detail two approaches for kernel based 3D object tracking. The first, referred to as the baseline 2D method, employs 2D kernel based trackers independently on each camera view, and combines the obtained results using the camera calibration information; this approach is akin to a decision fusion framework. The second approach is our proposed algorithm that uses a feature fusion framework for kernel based tracking directly in the 3D space. Notation-wise, in the derivations of this section, vectors $\mathbf{X}$ and $\mathbf{Y}$ represent 3D space points; vectors $\mathbf{x}$ and $\mathbf{y}$ denote points in 2D camera views; and matrices $\mathbf{P}^i$ represent camera projection matrices that associate 3D space points $\mathbf{X}$ to 2D image plane points $\mathbf{x}^i$, for camera i; namely $\mathbf{x}^i = \mathbf{P}^i\mathbf{X}$, assuming that the effect of lens distortion has been appropriately corrected, if necessary.

3.1. Baseline 2D Tracking Based Approach

Conventionally, tracking is performed in the 2D image space using features such as color, texture, gradients, etc. As computer vision systems evolve, they should be able to identify and track objects through the 3D scene to accomplish higher level analysis tasks such as action recognition, intent analysis, etc. Multiple viewpoints of an object can be used to obtain this 3D information. A potential solution to obtain the 3D tracks of an object would be to use an existing 2D approach to track the object projections in different views and subsequently combine the results to obtain the 3D location. This is the prevalent approach, as adopted by [15, 16]. The mean shift algorithm constitutes a popular algorithm for object tracking in 2D, and is described below in detail, as applied in individual camera views.

Let $\mathbf{x}_t^i$ be the current location of the object in camera view $i \in \{1, \ldots, N\}$ at time instant t. The subsequent 2D object location, $\mathbf{x}_{t+1}^i$, is estimated using the mean shift algorithm, recursively, as follows:

$$\mathbf{x}_{t+1}^i = \frac{\sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x}_t^i)} \mathbf{y} k'(\mathbf{x}_t^i - \mathbf{y}) w(\mathbf{y})}{\sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x}_t^i)} k'(\mathbf{x}_t^i - \mathbf{y}) w(\mathbf{y})} , \tag{1}$$

where $k(\cdot)$ is the profile of a smooth isotropic kernel, whose derivative is $k'(\cdot)$, and $w(\cdot)$ is a weighting function.

In a sense, Eqn. 1 can be viewed as the weighted average of pixel locations $\mathbf{y}$ in the image neighborhood, $\mathcal{N}(\mathbf{x}_t^i)$, of $\mathbf{x}_t^i$. The kernel $k(\cdot)$ assigns smaller weights to locations farther away from its center, whereas $w(\cdot)$ assigns higher weights to locations that have features (e.g., color) similar to the given object model being tracked. More formally,

$$w(\mathbf{y}) = \sum_{u=1}^{m} \delta\left[b(\mathbf{y}) - u\right] \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\mathbf{y})}} , \tag{2}$$

where $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1,\ldots,m}$ and $\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1,\ldots,m}$ denote the probability density functions (pdf) of m-features for the given *target model* and the *target candidate* at location $\mathbf{y}$, respectively, and δ denotes the Kronecker delta

function. The function $b : \mathcal{R}^2 \rightarrow \{1, \ldots, m\}$ associates the pixel at location $\mathbf{y}$ to its corresponding bin $b(\mathbf{y})$ in the quantized feature space.

The target model and target candidate pdfs can be estimated as kernel-weighted histograms

$$\hat{q}_u = C \sum_{\mathbf{y}^* \in \mathcal{N}(\mathbf{0})} \delta\left[b(\mathbf{y}^*) - u\right] k(\mathbf{y}^*)\,, \tag{3}$$

$$\hat{p}_u(\mathbf{x}) = D \sum_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} \delta\left[b(\mathbf{y}) - u\right] k(\mathbf{y} - \mathbf{x})\,, \tag{4}$$

where C and D are appropriately chosen to ensure that $\sum_{u=1}^{m} \hat{q}_u = 1$ and $\sum_{u=1}^{m} \hat{p}_u(\mathbf{x}) = 1$, respectively. The given target model is assumed to be centered at $\mathbf{0}$ and scaled to have a unit aspect ratio. The same scaling is applied to all target candidate regions during pre-processing. Intuitively, they are similar to regular feature histograms, but the kernel weighting makes the density estimation more robust, since the peripheral pixels that are often unreliable (due to clutter or background interference) have a smaller contribution.

The kernel function $k(\cdot)$ can be chosen to be any function that is continuous, differentiable, isotropic, and symmetric. A popular choice in this framework is the Epanechnikov kernel that has the following profile

$$k(\mathbf{x}) = \begin{cases} \frac{d+2}{2h^2 c_d}(h^2 - \mathbf{x}^T\mathbf{x})\,, & \text{if } \mathbf{x}^T\mathbf{x} \leq h^2 \\ 0\,, & \text{otherwise} \end{cases}, \tag{5}$$

where d is the number of dimensions, and c_d is a constant equal to the volume of a d-dimensional unit sphere. For the 2D case, $d = 2$ and $c_2 = \pi$. The value $h = h_{2d}$ denotes the bandwidth of the kernel that defines the scale of the target candidate. The bandwidth is related to the area occupied by the object's projection in the image plane. In the 2D case, the object's projection can change with time in each view, and this makes the adaptation of h_{2d} a nontrivial process. Moreover, it should be noted that the kernel can have a different bandwidth in each dimension, but for this analysis we assume that they are all equal.

To summarize, Eqn. 1 is used recursively to find the new location of the target in *each view* (independently), such that the similarity between the target model and the target candidate is maximized.[1] At the end of each iteration, the individual 2D tracking results from each of the camera views are combined using the Direct Linear Transformation (DLT) method [6] for triangulation to obtain the 3D location $\mathbf{X}_{t+1}$. A non-linear bundle adjustment [6] is employed to further refine the triangulation results. The triangulation error can be used to determine whether or not the 2D points $\mathbf{x}_{t+1}^i$ are consistent.

The shortcomings of this approach are apparent. Imagine how this method would fail when a person's head rotates, and the camera that was tracking the front of the face now observes the back of the head, and so on. Since the individual 2D trackers in each view are independent of each other, drifts can occur easily due to feature corruption, occlusion, view changes, etc.

3.2. Feature Fusion for 3D Tracking

The approach described in Section 3.1 can be thought of as decision level fusion, where each 2D tracker outputs its belief about the target location in the corresponding image plane. The triangulation process then fuses the results to obtain the 3D point in space corresponding to the true object location. We believe that most of the problems in this approach can be resolved by adopting a feature level fusion technique that is robust to corrupted or missing features.

A more principled approach would be to instead track the object directly in the 3D space by obtaining features from the contributing views. Firstly, this ensures that there will be no inconsistency among the 2D image locations

[1] In this case the sample estimate of the Bhattacharyya coefficient between $\mathbf{p}$ and $\mathbf{q}$, $\rho(\mathbf{x}) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{x})\hat{q}_u}$, is maximized.

of the object, as they are obtained by projecting a single 3D point to the respective view. Secondly, if the features are allowed to be fused such that their view dependence can be eliminated, then the object can be tracked in a unified manner, making it more robust to view-point changes and occlusions.

The algorithm described in Section 3.1 can be carefully extended to track a given object directly in the 3D space. The new 3D location, $\mathbf{X}_{t+1}$, can be directly estimated as

$$\mathbf{X}_{t+1} = \frac{\sum_{\mathbf{Y}\in\mathcal{S}(\mathbf{X}_t)} \mathbf{Y} k'(\mathbf{X}_t - \mathbf{Y}) w(\mathbf{Y})}{\sum_{\mathbf{Y}\in\mathcal{S}(\mathbf{X}_t)} k'(\mathbf{X}_t - \mathbf{Y}) w(\mathbf{Y})} \,, \tag{6}$$

where the summation is now performed in the 3D neighborhood of $\mathbf{X}_t$, i.e., $\mathbf{Y} \in \mathcal{S}(\mathbf{X}_t)$. Unlike in 2D, where the space is already discretized into pixels, one needs to select a proper sampling rate, s_{3d}, of the 3D region for the summation to be defined in the neighborhood $\mathcal{S}$. This will depend on how fine the features are spread in the 3D space. For example, a low sampling rate is sufficient for solid textured objects such as a car, but a higher sampling rate may be required to track a highly textured object such as a clown.

We modify the target representations to combine features from N sensors as

$$\begin{aligned} \hat{q}_u &= C \sum_{i=1}^{N} \sum_{\mathbf{Y}^*\in\mathcal{S}(\mathbf{0})} R(\mathbf{P}_i\mathbf{Y}^*, u) k(\mathbf{Y}^*) \,, \\ \hat{p}_u(\mathbf{X}) &= D \sum_{i=1}^{N} \sum_{\mathbf{Y}\in\mathcal{S}(\mathbf{X})} R(\mathbf{P}_i\mathbf{Y}, u) k(\mathbf{Y} - \mathbf{X}) \,, \end{aligned}$$

where

$$R(\mathbf{x}, u) = \delta\left[b(\mathbf{x}) - u\right] \mathcal{V}(\mathbf{x}) \,, \tag{7}$$

and $\mathcal{V}(\mathbf{x})$ is a boolean function that evaluates to 1 if the point coordinates $\mathbf{x}$ are valid, i.e. the point lies within the image frame view. The weight function is modified to accommodate contributions from all N camera views such that

$$w(\mathbf{X}) = \sum_{i=1}^{N} \sum_{u=1}^{m} R(\mathbf{P}_i\mathbf{X}, u) \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\mathbf{X})}} \,. \tag{8}$$

In contrast to the 2D case, where the weighted average of the pixel locations guide the search for the new location according to Eqn. 1, the proposed algorithm instead operates in 3D, where each (discrete) point in space has a weight associated with it. The spatial 3D patches that are similar to the target model in the chosen feature space and are closer to the kernel center will have a higher weight. Note that Eqn. 6 now represents a weighted average of 3D locations.

Finally, the constants in Eqn. 5 become $d = 3$ and $c_3 = 4\pi/3$. The bandwidth $h = h_{3d}$ relates to the volume of the object being tracked. Unlike the 2D bandwidth h_{2d} that changes with time, the 3D bandwidth – once defined – remains constant for objects of fixed size. For example, an adult human head is roughly a sphere of 80 mm radius, whose volume does not change over time for a particular person. This is advantageous, because, after selecting the initial value for h_{3d} and s_{3d}, one does not need to worry about updating them for the rest of the algorithm. For example, to track a typical human head sampled at every 20 mm, a reasonable set of parameter values can be $h_{3d} = 80$ mm and $s_{3d} = 20$ mm.

3.3. Benefits of the Proposed Approach

The 3D algorithm presented in Section 3.2 is preferred over the baseline algorithm of Section 3.1 for various reasons. Firstly, unlike in [1, 8], the 3D algorithm does not assume any planar structure in the scene to solve

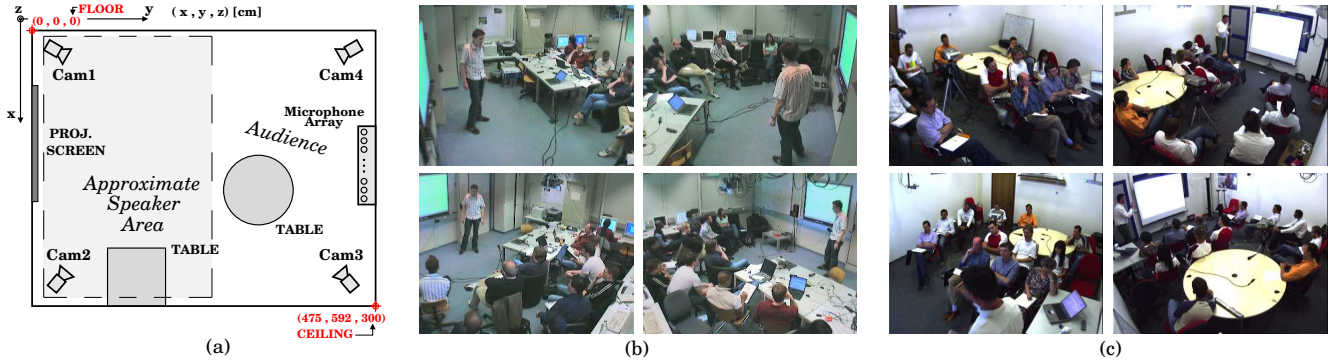

Figure 1. Brief overview of the CHIL lecturer video tracking task: (a) Schematic of the ITC smart room; (b,c) Examples of synchronous four camera views recorded at the (b) UKA and (c) ITC smart rooms during lectures.

for object correspondence between different views. Instead, the correspondence is automatically solved since the tracking is performed directly in the 3D coordinate space, as opposed to existing methods, where several 2D trackers are run independently on their respective target representations, and the final results are obtained by triangulation. The object location in each view can be directly obtained by projecting its 3D coordinates to 2D, using the camera projection matrices.

Another common problem associated with 2D based tracking approaches is that of target scale selection. Since the scale of the target can change drastically over time (as the object moves closer or further from the camera), one needs to devise an optimal method to adjust the scale of the kernel used for tracking in individual views. In [5], a simple heuristic for selecting the best scale was suggested. For each frame, the tracking results were obtained at three different scales, and the one that produced the best match was selected. In [4] an alternate scale space based method for mean shift blob tracking was presented. Evidently, attempts to address this problem have been made in the past, but there exists no optimal solution.

A novel contribution of our proposed framework is its ability to eliminate the issues related to scale selection. During initialization, a 3D volume defining the object of interest is selected. Since the real size of the object (such as a human head) does not change over time, the volume containing it remains constant. The projection of this volume to individual views automatically selects the appropriate regions in 2D images, from where the appearance features are extracted for tracking. Hence, the problem of selecting the scale in each view, and at each iteration, is eliminated altogether. Moreover, as opposed to one mean shift iteration per view for the 2D algorithm, the 3D method requires only one unified mean shift iteration per time step. Finally, any enhancements that can be applied to the 2D algorithm such as background weighted histograms, Bayesian filters, etc., can be also applied to the 3D case.

4. Experiments

We now report our evaluation of the proposed 3D tracking algorithm (Section 3.2) and its comparison with the baseline 2D approach (Section 3.1). For each method, we experiment with the algorithmic parameters and demonstrate how they affect tracking performance. Finally, we compare the best overall results obtained by these methods and present a brief analysis of them.

4.1. Database and Experimental Paradigm

Recently, there have been efforts to develop formal metrics for evaluation of tracking methods, such as the Performance Evaluation of Tracking and Surveillance workshops, but mostly the quality of an algorithm is still

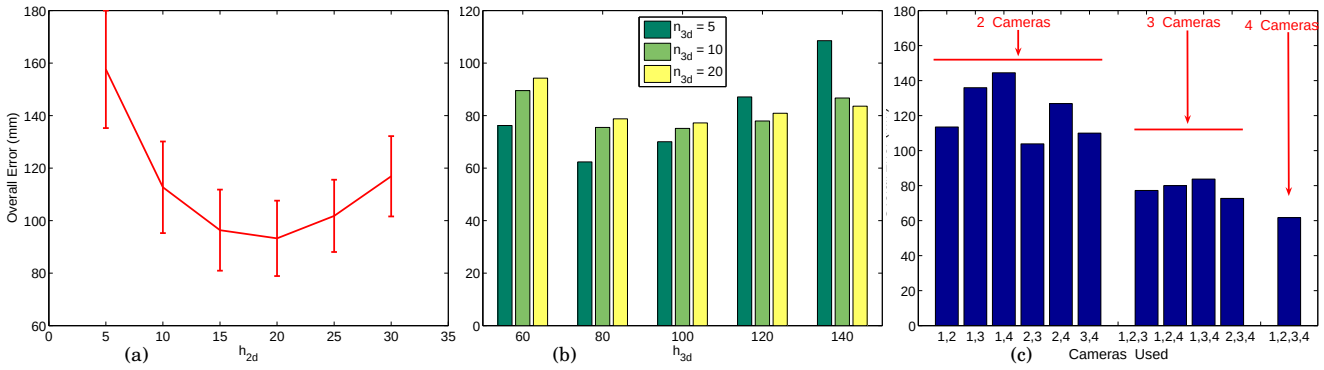

Figure 2. Overall tracking error (in mm) as a function of: (a) Bandwidth h_{2d} in the 2D algorithm; (b) Bandwidth h_{3d} and number of samples n_{3d} in the 3D approach (note: only the first one-third segment of each sequence is processed); (c) Number of cameras used in the 3D approach ($h_{3d} = 80$ mm, $n_{3d} = 5$ are used).

assessed by visual analysis of the results. If the ground truth can be made available then more quantitative results can be produced. For this reason we choose to evaluate our algorithms on a number of "interactive seminar" video sequences recorded and *manually* annotated as part of the CHIL ("Computers in the Human Interaction Loop") EU-funded project [3].

The evaluation set consists of 26 video sequences – 2 recorded at the smart room in the Istituto Trentino di Cultura (ITC), Italy, and the remaining 24 inside the smart room of the University of Karlsruhe (UKA), Germany. Each video segment contains approximately 4500 frames recorded at 15 Hz from 4 synchronized and calibrated cameras, i.e., a total of $4 \times 26 \times 4500 = 468$ k frames. Images are captured at a 640×480 and 800×600 pixel resolution for the UKA and ITC sequences, respectively. The camera calibration information (both intrinsic and extrinsic) is made available with the dataset. From this we simply derive the camera matrices $\mathbf{P}^i$ that are used for projection (3D $\rightarrow$ 2D) and triangulation (2D $\rightarrow$ 3D). Figure 1 shows representative camera views from the dataset and a schematic of one of the smart rooms.

The head location of the main speaker (lecture presenter) has been manually annotated every 1 sec for each sequence. For this reason, we focus on the lecturer 3D head position tracking. We evaluate the two algorithms in terms of how their output differs from the annotated ground truth head locations. In the following sections, *error* refers to the 3D Euclidean distance between the algorithm output and the ground truth at that frame. The error can only be calculated for the annotated frames. The sequence *mean error* is equal to the average error for all annotated frames. The average error produced by the tracking algorithm on all sequences is referred to as the *overall error*. We process every third frame of the full sequence to track the head location, i.e., the effective tracking frame rate is 5 Hz.

Methods to automatically bootstrap (initialize) a tracker and to detect when it fails are generally independent of the tracking algorithm itself but, nevertheless, important. The performance of the tracking algorithm will depend on how accurate these two other components are. Typically, complex detection schemes such as face detection, body part detection, 3D reconstruction, motion cues, etc., are employed in such tasks.

The scope of this evaluation is limited to comparing the performance of the proposed 3D tracking approach to the 2D baseline, and hence the effects of initialization and drift detection mechanisms need to be filtered out in our evaluation. For our experiments to be unbiased, we need to provide the same initial conditions for both trackers. For this purpose, we initialize the trackers based on the ground truth data available. In addition, tracker drift detection is achieved by comparing the tracking results to the ground truth. In particular, we introduce a drift threshold T on 3D error, identical to both trackers. Whenever the tracking error is higher than this predefined threshold, we re-initialize the target model for the respective tracking algorithm. The *number of re-initializations* required per sequence is hence another metric useful to tracking algorithm evaluation. A superior tracking method

is expected to require less re-initializations than competing approaches.

4.2. Baseline 2D Tracking Performance

One of the critical parameters in the traditional 2D mean-shift tracking framework is the kernel bandwidth, h_{2d}. In our experiments, the main speaker is roughly 2-3 meters from the cameras, and hence the head size does not change drastically within each view. For the baseline algorithm we have to choose the appropriate bandwidth such that the head region is properly tracked. The bandwidth controls the size of the 2D area that the object region is expected to occupy in the image. Ideally, the bandwidth should be adapted in some optimal manner to account for the object scale changes, but no satisfactory solution for scale adaptation exists. To simplify the comparison we used the same bandwidth across all views. In this experiment, we analyze the effect of the bandwidth parameter, h_{2d}, on the accuracy of the baseline algorithm. Figure 2(a) shows how the overall 3D error for the entire dataset depends on the bandwidth. The error is high for small and large values of h_{2d} and the minimum overall error of 93.2 mm is obtained for $h_{2d} = 20$ pixels. As seen in the plot, the overall error for the dataset ranges between 93 and 157 mm for various h_{2d} values. The error bars denote the error range (minimum to maximum) on the 26 database sequences.

4.3. Proposed 3D Tracking Performance

The two parameters for the 3D algorithm are the bandwidth h_{3d} and the spatial sampling s_{3d}. Once these parameters are decided for the object that needs to be tracked, then there is no need to update them. Objects that change their size over time are excluded from this claim, but most objects such as humans, animals, vehicles, etc., have fixed volumes over time.

For a given spatial sampling rate s_{3d} in each dimension, we can divide a 3D cube of side $2h_{3d}$ into $(n_{3d})^3$ partitions, where $n_{3d} = 2h_{3d}/s_{3d}$. We experimented with various values of the bandwidth h_{3d} and the number of samples n_{3d} (in each dimension) to determine how the overall tracking error rates are affected (Figure 2(b)). Note that we only used the initial one-third segment of each sequence to generate these results. It should be mentioned again that the human head is roughly a spherical volume, and we wish to examine how smaller and larger kernel bandwidths would affect the tracking results. The best result is obtained for $h_{3d} = 80$ mm and $n_{3d} = 5$ achieving an overall 3D error of 62.3 mm. As expected, a smaller number of samples are preferred for smaller bandwidths, while a larger volume (bandwidth) demands more samples to produce better results. Of course, the higher the number of samples, the higher will be the computational cost of the algorithm.

Another interesting question can be asked about the effect of the number of cameras "fused". Figure 2(c) confirms the intuition that by adding more cameras, and hence more information, the tracking error is reduced. The first set of bars in the figure denote the overall errors when various combinations of two (out of the four available) cameras are used. Evidently, adding a third camera significantly reduces error, as seen in the second cluster of bars. Finally, the minimum overall error of 61.7 mm is obtained by using all four cameras. If more cameras are added, we speculate that the error will decrease further until it tails off at a level where incorporating additional information becomes redundant.

4.4. Comparison of the two Approaches

In this section we first analyze the effect of parameters that are common to both 2D and 3D algorithms, and subsequently compare the best results obtained by each method.

For our experiments, we used color histograms in the *rgb* color space as features to track the human head. Figure 3(a) demonstrates how the choice of number of bins per color channel affect the performance of each algorithm. Evidently, a higher number of bins provide a better resolution for the features to be tracked, but this

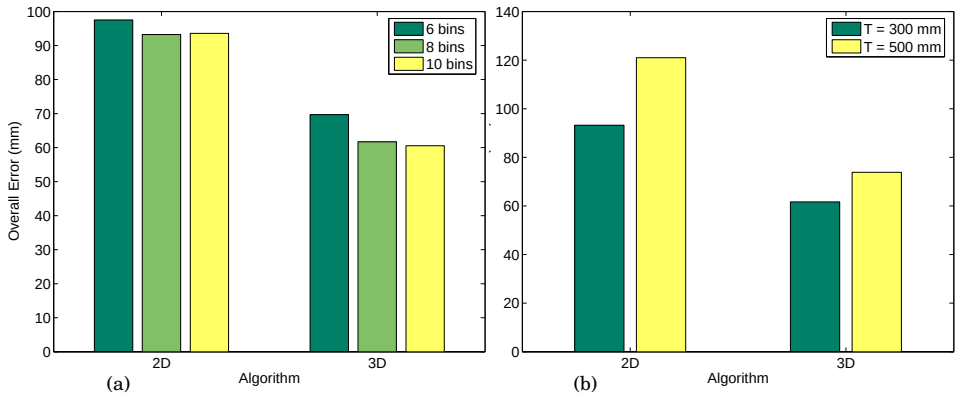

Figure 3. Overall error of the 2D and 3D tracking algorithms, depicted versus various choices of: (a) number of bins per color used in the histogram estimation; (b) re-initialization threshold T.

results in a tradeoff requiring higher computational cost. For both algorithms the performance was satisfactory when 8 bins per channel (i.e. $8 \times 8 \times 8 = 512$ bins) were used.

Another common parameter is the re-initialization threshold, T, as defined in Sect. 4.1. For each algorithm we re-initialize the target model whenever the 3D Euclidean tracking error exceeds T. Figure 3(b) depicts the results for both 2D and 3D tracking algorithms versus different values of T. The overall error for the 3D algorithm drops from 73.8 mm to 61.7 mm (16% relative reduction), while the 2D algorithm drops from 121.0 mm to 93.3 mm (23% relative), as T is changed from 500 mm to 300 mm. The large difference between the two errors for the 2D algorithm, as compared to that of the 3D method, suggests that the former has a higher occurrence of large errors than the latter.

For the various values of the algorithmic parameters, the general range of overall errors on the entire dataset for the 2D algorithm is around 93–121 mm, while that for the 3D approach is 60–74 mm, when all ground truth frames are used for evaluating the errors. The 3D algorithm clearly outperforms the 2D baseline approach in terms of overall error and number of re-initializations required.

We now compare the best overall results obtained from each method and present the conclusive results. The minimum overall error induced by the 2D algorithm is 93.2 mm as opposed to a 60.5 mm error achieved by the unified 3D tracking method. Therefore, a significant 35% relative reduction in the overall error can be obtained by using the proposed 3D tracking approach over the baseline 2D method. In addition, the average number of re-initializations required per sequence also gets reduced from 14.54 for the 2D approach to 4.46 for the proposed 3D algorithm (a 70% relative reduction). This implies that while the 2D method has to be re-initialized every 20 sec on an average, the 3D technique requires re-initialization only every 67 sec.

The parameters that produce the best overall results for each algorithm are $h_{2d} = 20$ pixels, $h_{3d} = 80$ mm, and $n_{3d} = 5$. In Figs. 4(a,b), we compare the mean error and number of re-initializations required per sequence for both methods, using these parameters. In both cases, we re-initialize the tracker if the error exceeds $T = 300$ mm. As shown, the 3D algorithm results in a lower mean error per sequence for most cases, and it also requires less re-initializations, as compared to the 2D method. Noticeably, there is no need to re-initialize the 3D algorithm even once for some sequences (each sequence was around 5 min long).

Finally, we present (in Fig. 4(c)) the representative tracking results per frame for one of the 26 sequences, as produced by both the 2D and 3D algorithms. The 3D tracker has smaller tracking errors as compared to the 2D tracker for most of the sequence. It can also be seen that most errors from the 3D tracker are below the re-initialization threshold of $T = 300$ mm, and hence it requires just 3 re-initializations as opposed to 22 required by the 2D baseline. For this particular sequence, the mean errors are measured at 105.3 mm and 47.3 mm for the 2D and 3D methods.

Currently the un-optimized algorithm prototypes are implemented in Matlab achieving close to real time per-

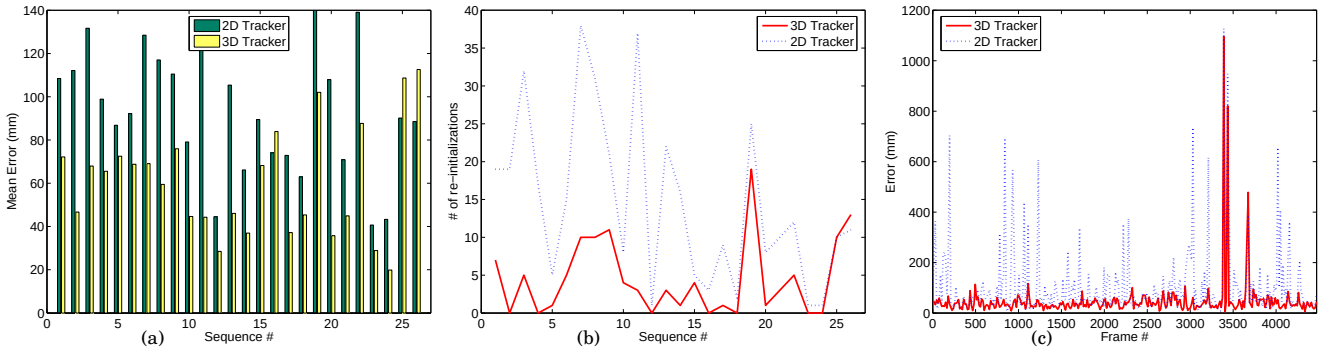

Figure 4. Comparison of the baseline 2D and the proposed 3D approach on the CHIL evaluation set: (a) Mean error for all 26 test sequences; (b) Number of tracker re-initializations over all 26 sequences; (c) Tracking error over time (frame) for one of the test set sequences. Error peaks over 300 mm trigger tracker re-initialization.

formance. The 3D algorithm is roughly 1.5–1.7 times slower than the 2D version, however in practice this can be compensated by the fact that less initializations are required, as well as the fact that tracking drift detection need not be performed as frequently as in the 2D baseline, due to the superior 3D accuracy. In any case, the proposed algorithm is expected to perform real-time, once optimized and implemented in C/C++.

5. Summary and Conclusion

We presented a unified approach to object tracking in 3D, based on a feature fusion scheme of multiple calibrated camera views. The kernel based tracking approach, when applied directly in 3D, helps resolve the scale selection and view correspondence issues that effect traditional 2D tracking algorithms. The proposed feature fusion scheme combines evidence from different views and is robust to feature corruption, occlusion, and view dependence. The experimental results demonstrate that the proposed algorithm achieves a significantly lower error rate and dramatically reduces tracker re-initialization, as compared to a baseline 2D kernel based tracking approach.

The area of 3D tracking is becoming popular, and useful information can be extracted from such algorithms to be used in higher level analysis, for example action recognition. The presented work is still in progress, and we plan to further investigate the related problems of automatic tracker initialization and drift detection, necessary to complement our proposed 3D tracking algorithm. Another area of future work includes the evaluation of this framework in other applications, such as outdoor tracking for visual surveillance.

References

[1] J. Black and T. Ellis, “Multi camera image tracking,” *Image and Vision Comp.*, 24(11):1256–1267, 2006.

[2] Y. Boykov and D. Huttenlocher, “Adaptive Bayesian recognition in tracking rigid objects,” In *Proc. Comp. Vis. and Pattern Rec.*, pp. 697–704, 2000.

[3] The CHIL consortium web-site:
`http://chil.server.de`

[4] R. Collins, “Mean-shift blob tracking through scale space,” In *Proc. Comp. Vis. and Pattern Rec.*, pp. 234–240, 2003.

[5] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. Patt. Analy. and Mach. Intell.*, 25(5):564–577, May 2003.

[6] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[7] M. Isard and A. Blake. “Condensation - conditional density propagation for visual tracking,” *Int. J. of Comp. Vis.*, 29(1):5–28, 1998.

[8] S. M. Khan and M. Shah, “A multiview approach to tracking people in crowded scenes using a planar homography constraint,” In *Proc. European Conf. Comp. Vis.*, 2006.

[9] A. Mittal and L. Davis. “M2Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo,” In *Proc. European Conf. Comp. Vis.*, pp. 18–36, 2002.

[10] K. Nickel, T. Gehrig, H.K. Ekenel, J. McDonough, and R. Stiefelhagen, “An audio-visual particle filter for speaker tracking on the CLEAR’06 evaluation dataset,” (In Press), *Proc. CLEAR Evaluation Wksp.*, Southampton, United Kingdom, 2006.

[11] R. Rosales and S. Sclaroff, “3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions,” In *Proc. Comp. Vis. and Pattern Rec.*, pp. 117–123, 1999.

[12] C. Shan, Y. Wei, T. Tan, and O. Ojardias, “Real time hand tracking by combining particle filtering and mean shift,” In *Proc. Int. Conf. on Auto. Face and Gesture Recognition*, pp. 669–674, 2004.

[13] K. She, G. Bebis, H. Gu, and R. Miller, “Vehicle tracking using on-line fusion of color and shape features,” In *Proc. Int. Conf. on Intelligent Transportation Sys.*, Washington DC, Oct. 2004.

[14] A. Tyagi, J. Davis, and M. Keck, “Multiview fusion for canonical view generation based on homography constraints,” *ACM-MM Work. on Video Surveillance and Sensor Networks*, pp. 61-69, Oct. 2006.

[15] Z. Zhang, G. Potamianos, A. Senior, S. Chu, and T. S. Huang. “A joint system for person tracking and face detection,” *Proc. IEEE Int. Work. Human Comp. Interaction*, Beijing, 2005.

[16] Q. Zhou and J. K. Aggarwal, “Object tracking in an outdoor environment using fusion of feature and cameras,” *Image and Vision Comp.*, 24(11):1244–1255, 2006.

[17] D. Zotkin, R. Duraiswami, and L. Davis, “Joint audio-visual tracking using particle filters,” *EURASIP J. Applied Signal Process.*, 2002(11), 2002.