Improving Multiple People Tracking Using Temporal Consistency

Rania Y. Khalaf   Stephen S. Intille

House_n: the MIT Home of the Future Project
Massachusetts Institute of Technology
1 Cambridge Center, 4FL
Cambridge, MA 02142, U.S.A.
intille@mit.edu

Technical Report - July 2001

Abstract

This work presents a real-time multi-person or multi-object tracking algorithm that uses multiple hypothesis reasoning in time to enforce multi-person match constraints. The algorithm is intended to augment, not replace, existing multi-person tracking methods. We demonstrate how tracking systems that use inter-frame feature matching can be improved by enforcing contextual matching constraints throughout a 1-5 second temporal window. Robust and efficient multiple hypothesis reasoning in time is achieved for a useful class of tracking problems using a dynamic programming framework. Results are described for a dataset of 40 minutes of test video taken from a static, top-down camera and with two to four people moving about a small room.

1. Introduction and Related Work

Our target application is reliable real-time tracking for perceptual user interface applications of up to four people in a small room from a single overhead camera. We demonstrate that the multi-object tracking problem can be formulated in a dynamic programming framework, allowing for efficient multi-hypothesis temporal reasoning. Prior work in tracking multiple people has generally assumed that features such as color histograms, correlation patches, velocity estimation, and blob-distance metrics can be used to *independently* track each object of interest. In a few cases, explicit reasoning about the relationships between objects has been used to handle brief occlusion events (e.g. [7, 16, 12]) or to reason about contextual constraints between two adjacent frames [10]. Here we take a different approach. We show that when constraints are enforced and feature evidence is integrated over a relatively long temporal window instead of simply between frames, matching performance can be substantially improved; by considering all physically realizable situations a tracking algorithm can recover from errors that result from bad short-term feature matching.

The fundamental problem with tracking people is that, unlike rigid, geometric objects, there are few stable features to track over time. Color, size, velocity, and shape features of people all vary dramatically as people move about an indoor environment. Strengthening the model used for tracking using kinematic or dynamic models of human movement has so far proven most useful for tracking single people from high-resolution, side-camera views [19]. Instead of searching for more invariant features or constraints to use for matching in a short time window (i.e. between frames), we instead consider the use of temporal consistency to provide constraint for relatively weak features integrated over longer temporal intervals.

Most multi-person tracking work has used stereo or color blob-representations of people that are computed by clustering pixels that differ from statistical models of the "background" [18, 10, 13]. The recovered blobs can then be statistically characterized by shape and color features (e.g [18]). Generally the algorithms either assume a top-down view with little full-occlusion [9] or a side view with constant velocity heuristics to handle full frontal occlusion in pass-by situations [16].

Although blob features can be matched between adjacent frames directly, in practice these features are too weak to provide reliable multiple-object tracking. Prior work explores some additional sources of constraint. The EasyLiving tracking system [13] uses color stereo to locate people blobs, and when one person's blob occludes another a histogram intersection feature is used for matching. The W4 algorithm [6] tracks multiple people in outdoor scenes using second order motion models of recovered blobs and blob parts (i.e. body parts recovered using side-view heuristics) to predict each person's next location and constrain inter-frame matching. "Closed world" contextual information about a scene and the relationship between objects being tracked has been used to select the best pixels to match between individual frames [9]. Contextual information about how many people are in an enclosed space has been used to weight multiple blob features in a match score matrix to track rapidly-moving and interacting people in real-time [9].

Other blob tracking systems have used Kalman filter prediction to track positions of blobs between frames [5] and features such as adaptive templates that update each new frame [2]. While another approach uses separate trackers for different targets that combine multiple attributes (color and contour) of complex objects [15], it is not designed to track multiple articulated, non rigid, objects with similar contours such as people in a room.

None of the work mentioned above uses match consistency over a long temporal window to provide additional tracking constraints; decisions are generally made using just two adjacent frames and velocity estimation. Any of these algorithms could use the dynamic programming approach presented here to consider additional temporal constraints.

Dynamic programming has been used for sub-pixel sized target tracking to overcome the weakness of the signal [1] and for improving tracking of single objects [3]. It has also been used for tracking deformable contours of single objects [4]. These systems do not use dynamic programming to explicitly represent multi-person matching. Although the condensation algorithm [11] has been used to keep multiple hypotheses of object contours and for tracking similar objects [14], the framework has not been used to model inter-object constraints over time.

2 Modeling consistency over time

In this section we describe our general framework for tracking. Assume an algorithm exists that can track certain features between frames. To extend such an algorithm to multi-person tracking each person can be tracked independently from the others; in some cases the proximity of objects may be used to select features to track or overcome short occlusion situations using constant velocity assumptions [6, 9].

Here we are interested in exploiting the long-term temporal constraints *between* the objects being tracked. We would prefer that the algorithm defer making any short-term tracking commitment based on inter-frame feature matching of individual objects and instead wait to find a longer-term best hypothesis that simultaneously takes into account the long-term constraints and feature match evidence. We propose to use the following three temporal constraints. (1) *Spatial continuity*: People who enter a "visual merge"[1] are the same people who leave a merge; people cannot disappear unless there is a known explanation; (2) *Continuity in motion*: There is a limit to the amount of distance people can travel in given amounts of time, and to get from A to B requires that an object travel between the positions; (3) *Continuity in appearance*: Even though visual features can provide misleading information in any given frame or for short windows of time when tracking non-rigid objects, on average people will tend to look more similar to themselves than to others in the same space.

[1]We define a "visual merge" as any situation where the features being tracked can no longer distinguish the positions of two objects (e.g. segmentation merges the objects).

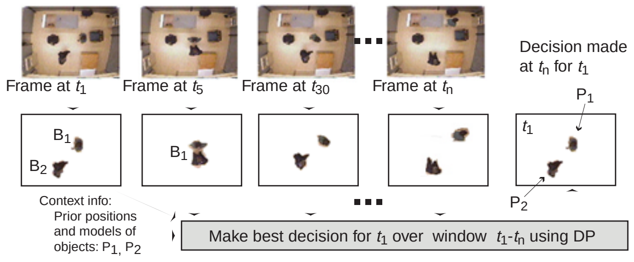

Frame at t_1
Frame at t_5
Frame at t_{30}
Frame at t_n
Decision made at t_n for t_1
B_1
B_2
B_1
t_1
P_1
P_2
Context info: Prior positions and models of objects: P_1, P_2
Make best decision for t_1 over window t_1-t_n using DP

Figure 1: The tracking problem. Feature blobs are computed for incoming sample frames. Contextual information (e.g. the number of people in the scene, models of each object) is used to compute the best possible person to blob assignment over a window of time given all features and constraints. A final decision about the positions of objects at t_1 is delayed until evidence from samples $t_1 - t_n$ is observed.

These constraints appear straightforward and most tracking algorithms enforce them in a local space-time window. However, here we enforce the constraints both for individual objects being tracked and for the entire group of objects being tracked simultaneously; we do this for a window of several seconds in length. Therefore, we can sometimes avoid the matching errors that would otherwise be made by algorithms using relatively weak inter-frame matching features.

Figure 1 illustrates the matching problem. Assume some algorithm identifies feature clusters (e.g. difference blobs, B) in incoming frames $t_1 - t_n$ and that the tracking algorithm has contextual knowledge such as the starting position of the objects in the scene and feature models of those objects. The goal is to assign each known person, P with some B but where it is assumed several P may be in one blob. Some B may actually be spurious due to noise or segmentation inaccuracies.

We are interested in selecting P to B matches that are globally consistent across the scene and that integrate feature evidence over the entire temporal window, $t_1 - t_n$, taking into account the constraint introduced by the patterns of movements of the group as a whole. Although the algorithm can make a preliminary best guess at assignment as soon as a new frame arrives, a final assignment is delayed until a temporal window is observed. Samples are not necessarily obtained from every incoming frame or at set intervals. In this work, we use temporal intervals of up to 5 seconds with 500 ms delays between samples.

2.1 Multiple hypothesis reasoning using DP

We cast the temporal continuity problem in a first-order Markovian framework that can be solved efficiently using dynamic programming (DP). Match decisions are made over a window of time. As opposed to making a match decision with each new incoming frame or keeping only a few top match hypotheses and assuming smooth trajectories[8], a graph is used to represent all physically realizable tracking decisions that could be made during this window *without any velocity filtering and prediction*. The graph is designed to enforce the constraints between the objects being tracked.

Figure 2a shows a portion of such a graph for two incoming samples from times, t_n and t_{n+s}, where s is an arbitrary sample time. For each sample, the number of objects in the scene is known, as is feature models of those objects. Also known are the clusters of match features, B_i, where i varies from frame to frame depending on the configuration of objects and noise. Each new sample adds a new "stage" to the graph, and each stage represents all possible object (P_i) to segmented blob (B_i) matches where each P must be assigned to a B. For example, stage t_n has 3 blobs and 3 people. Two of the nodes in Figure 2a are expanded to show two possible match situations. All together, if there are b blobs and p people objects there are b^p nodes in a stage. Stage t_n has 27 nodes and stage t_{n+s} has 8 nodes.

Adjacent stages are fully-connected with links that each have a transition cost. For example, the highlighted link in Figure 2a shows a transition from a state where P_1 is assigned to B_1 in t_n, P_2 and P_3 are assigned to B_2, and all three objects have moved to the blob labeled B_1 in t_{n+s}. A cost is associated with this transition that depends upon how physically realizable it is and how well the model features of each P_i match to the Bs in t_n and t_{n+s}.

Given a starting node, a dynamic programming algorithm can find the lowest cost path through the graph. With an appropriate cost function for the links encoding long-term constraints between match decisions in a Markovian way, this path will represent the best match across all the data observed in the entire window. The key observation is that the temporal continuity constraints can be represented as node transitions costs.

Figure 2b conceptually demonstrates the benefit of such an approach for a tracking problem where two people walk towards each other, walk adjacent for a while, and then walk away. The "people models" boxes represent the feature models for each object (i.e. the closer the color, the better the feature match). Stage t_n contains two people, P_1 and P_2, in known positions (assigned to positions of B_1, B_2) that have been committed. Typically when people are moving quickly, the observed features of each P vary as the person moves about the environment, but the features are still consistent over time.

The leftmost graph represents all possible match situations from the committed stage to stage 4 at t_{n+3s}; the DP best path considering stages 1-4 is highlighted. However, when stages 3, 4, and 5 are considered, evidence mounts that the match decision made at stage 3 was erroneous. The right graph illustrates how the original graph changes when the new stage 5 is added. The current best decision for stage 1 is committed and stage 0 (sample t_n) is dropped from the window. Stage 5 is connected to the end of the graph. Now the reasonable feature match from stage 4 and the good feature match from stage 5 overwhelm the bad feature match from stage 3. Even though by stage 5 physical continuity of motion might prohibit an instantaneous tracker from allowing a switch, our network stores all potential realizable situations and still allows a switch if it results in a physically consistent path over time. DP finds the new best path (highlighted). Using proximity, the network keeps track of when it was possible for assignment swaps to occur. While this example implies that color is the feature of interest, other features and weighted combinations of features can also be used.

In summary, even if feature evidence initially suggests an incorrect match, as more evidence is acquired over time, the best path is guaranteed to be selected. The path score is dependent upon the transition cost from node n_{t-1} to n_t:

$$\begin{aligned} c(n_{t-1}, n_t) &= \frac{1}{N}(aM + \\ &\sum_{i=0}^{People} f_{dist}(P_i, n_{t-1}, n_t, \Delta t) + \\ &\sum_{i=0}^{People}\sum_{j=0}^{Features} \alpha_j(f_j(P_i, n_{t-1}, n_t))) \end{aligned} \quad (1)$$

where N is the number of Ps in the room and normalizes the score, a is the total number of people merged minus the number of merges, M is a constant merge cost, $f_{dist}(P_i, n_{t-1}, n_t, t)$ is a distance score for the distance P_i moves from its blob in n_{t-1} to its blob in n_t including a penalty when this distance is too large for time t, α_j is a weighting constant for feature j, and $f_j(P_i, n_{t-1}, n_t)$ is the cost of matching a particular feature, f_j, from person i's blob in the node n_{t-1} with person i's blob in the new node n_t.

The merge cost, M, is a constant used to penalize the system for assigning multiple people to the same blob. If this penalty is not used, the algorithm may prefer to keep

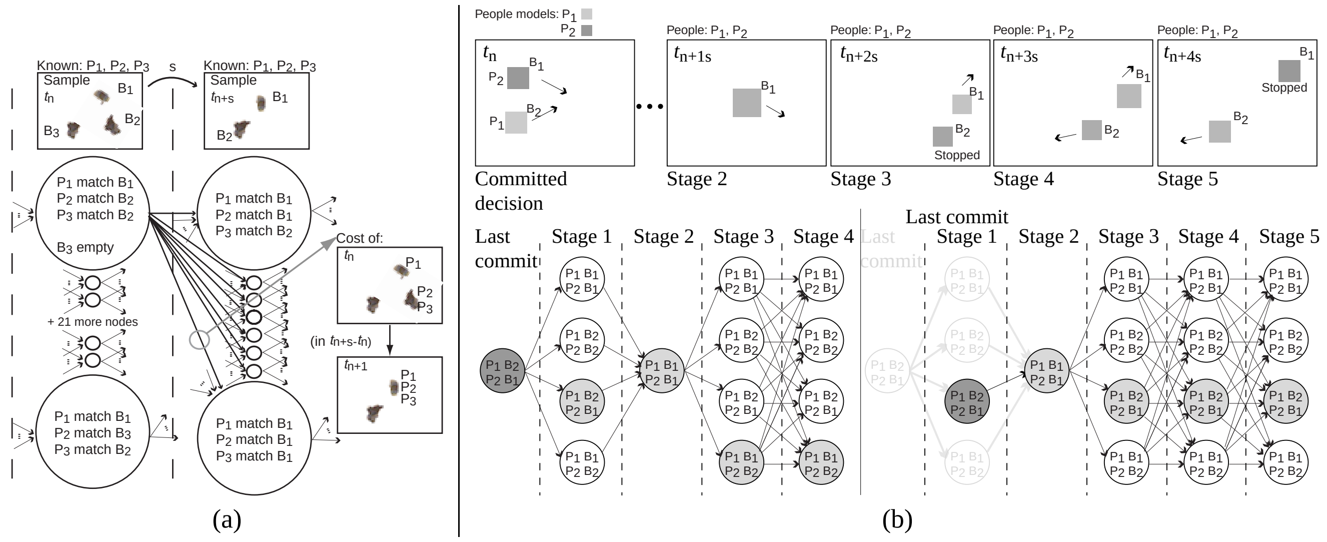


Figure 2: (a) Two stages of the temporal window graph, as described in the text. (b) By representing all possibly physically realizable hypotheses in a Markovian framework, feature evidence accumulated in time – even when objects are far apart – can lead to a revised hypothesis that corrects tracking mis-assignments, as described in the text.

people previously merged in the same merged position even when one person is actually moving away and should be assigned to another blob. It is preferable to penalize merges instead of empty blobs because a path should not be penalized for not assigning a P to a blob that resulted from noise.

2.2 Typical complexity and model updating

Encoding constraints in first-order Markovian link costs allows an enormous number of hypotheses to be searched efficiently using dynamic programming techniques. Since each stage in the temporal window graph represents a full match state describing how all tracked objects match to all known blobs, there are b^p nodes, where b is the maximum number of blobs in any stage and p is the number of people known to be in the room. Therefore, since each stage of the graph is fully connected to adjacent stages, dynamic programming can find the best-cost path in $O(nb_{max}^{2p_{max}})$ where n is the number of stages and p_{max} is the maximum number of people ever in the room during a sequence and b_{max} is the maximum number of feature blobs in any sampled frame. For many useful people tracking problems in small rooms, $b_{max} < 6$ and $p_{max} < 5$. Typical case performance can be substantially better, as shown in Section 4.3.

Two issues remain. First, for each P known to be in the scene we need feature models. These can be static models, but a better approach is to update the models using the tracking algorithm. In this work we do so by updating models only in the following situation: (1) when a decision has been finalized because the entire temporal window has been considered, and (2) when the final tracked position of this object indicates that it is isolated from other objects; this situation presents a good opportunity to update features without segmentation confusion from other objects being tracked.

A second issue is initialization – keeping track of how many Ps are in the room. The temporal window graph can be extended so that it also represents contextual information such as P_i left the room and P_i entered the room if a special region of the room is designated for entry/exit (i.e. a door) (see [10] for an example of using a "door" in an instantaneous-decision tracker). For the results described here, however, we have manually initialized the starting positions of objects and no objects enter or leave the room during each test video clip.

3 Real-time implementation

We have implemented a real-time version of the multiple hypothesis temporal reasoning algorithm. Our scene contains a static camera observing a room from above (see image in Figure 4). We use motion-differencing to obtain object blobs and use color and distance metrics for inter-frame matching features.

Blob extraction proceeds as follows. The room is emptied of all people and a statistical model of the background is created using a YUV color space. Incoming images are compared to the background model using the Mahalanobis distance. "Foreground pixels" are dilated and contiguous pixels are merged into "blobs." The blobs correspond to a single person, multiple people standing close together, shadows, moved objects, or camera noise.[2] If two or more objects are close to each other, the background subtraction will yield one blob for all of them.

Each blob is assigned a position (centroid of the bounding box), a bounding box, a color histogram (computed using the foreground image pixels in the blob), and an arbitrary label (e.g. B_3).

The transition cost function in equation 1 requires that for any transition between two matching stages a distance and a feature cost be computed. Both computations must be fast because they are computed for all possible transitions between all possible matching states each time a new sample is received. We set, M, the merge penalty to be .35, which was found by trial and error.

3.1 Distance cost

We call $(f_{dist}(P_i, n_{t-1}), n_t, \Delta t)$ from equation 1 the *distance penalty*. This value is computed by imposing an inter-stage, time-dependent penalty on the distance score, d_s. d_s is the distance between the blob that person P_i is assigned to in node n_{t-1} and the blob that P_i is assigned to in node n_{t+s}. We use a fast heuristic to compute this distance and three cases must be considered. For clarity, imagine overlaying the two blobs on the same image, as in Figure 3a. If the bounding box of one blob completely encloses the bounding box of the second blob, the blobs are assumed to completely overlap, as shown in Figure 3b, and $d_s = 0$. If the bounding boxes of the blobs partially overlap as in Figure 3c, $d_s = C_{PO}$, where C_{PO} is a partial overlap penalty (6 cm in our implementation). Finally, if the bounding boxes do not overlap, as shown in Figure 3d, then $d_s = d + C_{NO}$, where C_{NO} is a no-overlap penalty (16 cm in our implementation), and d is an efficient approximation of the shortest distance between the contours of the two blobs estimated by finding the length of line e_1 to e_2, where line c_1, c_2 connects the centroids of the two blobs and e_1 and e_2 are on the blob contours.[3]

The C_{NO} and C_{PO} costs are used to bias the algorithm towards tracking assignments that keep objects in the same place unless good features are observed that overwhelm the penalties. Similarly, by setting C_{PO} to be less than C_{NO}, the algorithm is biased towards situations such as Figure 3c where two blobs are largely overlapping over those where blobs are very close but not overlapping.

The d_s is then scaled by the allowable distance, $D_a = V_a * \Delta t$, where V_a is 145 cm/s, just a bit less than a typical walking velocity. To get the value of the distance penalty, the scaled distance score is plugged into a function, $g(x)$, that returns x if the distance score is less or equal to the allowable distance; otherwise, it substitutes x into the penalty sigmoid. Therefore, $f_{dist}(P_i, n_{t-1}, n_t, \Delta t) = g(d_s/D_a)$, where $g(x)$ is

$$g(x) = \begin{cases} x & \text{if } x <= 1 \\ \frac{50}{e^{\frac{-x+2}{0.15}}+1} + 1 & \text{otherwise} \end{cases}$$

Intuitively, matches that assume that large objects move unrealistically quickly are heavily penalized.

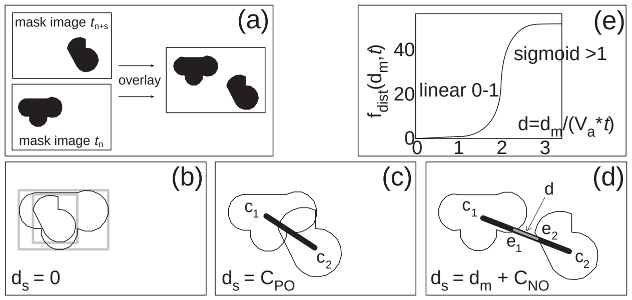


Figure 3: (a) Blobs from t_n and t_{n+s} are overlaid on the scene coordinate system. (b) When bounding boxes around blobs indicate that blobs are overlapping, there is no distance score. (c) When bounding boxes partially overlap, a partial-overlap penalty is used. (d) When bounding boxes do not overlap, a fast estimate of the distance between contours is computed and added to a no-overlap penalty.

3.2 Color feature cost

In the results presented here, we use only one feature – color histogram intersection [17]. Therefore, α_j in equation 1 is 1. Color histogram intersection returns a match score of how similar two blob image regions are based on their color distributions; it is linear in the number of elements in the histogram and robust to multiple views, scales, and occlusions [17].

We define $f_j(P_i, n_{t-1}, n_t)$ in equation 1 – the cost contribution for a state transition based on feature-j (color histogram matching) – to be simply the histogram intersection score found matching a stored histogram for person P_i to the blob that P_i has been assigned to in the next node, n_t.

Given a model histogram for a person, M, and a blob histogram I with n buckets each, the algorithm returns the number of corresponding pixels of the same color that are found between the image and the model, normalized by the number of pixels in the model. Therefore, the equation for the histogram intersection score between two blobs is:

$$H(I, M) = 1 - \frac{\sum_{j=1}^{n} min(I_j \frac{S_M}{S_I}, M_j)}{\sum_{j=1}^{n} M_j} \qquad (2)$$

[2]Our implementation actually uses an adaptive background algorithm that slowly adapts any unoccupied regions of the space, but this enhancement does not impact the results discussed here.

[3]Distances between edges of blobs is preferable to distance between centroids because when an object is merged into a multi-person blob and then splits the centroid of the two blobs can move more than the object itself physically could.

where S_M and I_M are the sizes of the model and blob image histograms respectively. A low score is a better match.

To make histogram intersection more robust to changes in illumination, in this work we construct UV histograms (no Y) with 8 bins on each axis. We also use a heuristic where each person is modeled with up to 9 histograms which are associated with sections in the room (as in [13]). The algorithm uses the histogram closest to the blob's location when performing matching, allowing for variation in appearance of a person due to lighting differences and viewpoints in different parts of the scene. If no histogram model is available for an object, a neutral score of .5 is selected.

Each object in the space needs a histogram model. This model is acquired using the tracking system by taking histogram samples when a person is not near any other people. Whenever the tracker commits a decision, the histogram of each isolated person from that time is acquired and used as the histogram model whenever each person is near that location of the scene again.

4 Evaluation

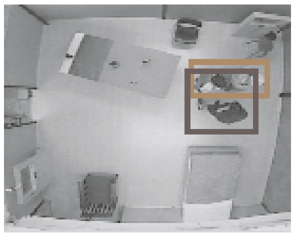
The real-time version of this algorithm was run on video clips of multi-person interaction acquired from a camera mounted above the 6m by 4m room shown in Figure 4.[4] The clips were acquired by asking between 2-4 people to walk around randomly and to meet and talk to each other as they walked around. For some sequences, people were asked to wear a bright color. In most of them, however, people walked in with whatever they were wearing the day the video was shot. The eight sequences vary in length between two to 10 minutes.

Generally, the activity is as follows: people meet in different parts of the room multiple times, in groups of two, three, or four. Some people sit on the floor or on the chairs, jump up and down, wave their arms about, and lean backwards and forwards as they pass under the camera. Clips of our test image sequences with tracking results overlaid can be viewed at [submitted file].

4.1 Error classifications

Single-person tracking is robust. Opportunities for error occur when multiple people interact. We classify the opportunities for tracking error by considering "merge" events, i.e. when the difference blobs of two or more people merge into a single blob because of proximity. The number of misassignments the tracker can make after a merge is proportional to the number of people in the merge (e.g. a 3-person merge can lead to a maximum of 3 misassignments). We classify an n-person merge as creating n "post-merge-assignment error" opportunities.

[4]Testing was performed on a 300 MHz dual processor Pentium II system with 512 MB RAM running unoptimized Java code.



Frames		
Grouped	MSE	GPE
632	45	17

Figure 4: The image shows a situation where three people have been improperly segmented into two blobs (indicated by bounding boxes) The table indicates how many "in-merge" errors were encountered in our dataset.

Two more errors can occur *during* merges, as illustrated by Figure 4, when the blob detection algorithm fails to segment blobs in a perceptually-meaningful way. One occurs when different parts of different people are marked as separate blobs: "merge segmentation errors" (MSE). The other case occurs when people are in a group and the algorithm picks up a few overlapping blobs in which one corresponds to a complete person from the group and that person is mislabeled: "group proximity errors" (GPE).

4.2 Results

The algorithm was tested with 1 (instantaneous decision), 2, 5, and 10 temporal window stages. 500 ms elapsed between each stage's sample, so the temporal windows were of length 1s, 2.5s, and 5s for the 2, 5, and 10 stage runs respectively. In the instantaneous case, the algorithm commits to a decision immediately upon receiving a new frame and ran at 5Hz.

As shown in Figure 5, the number of errors declines as the temporal window size is increased. Even using a temporal window of only 1s improves the results. Most mistaken assignments (12 out of 48 with the longest window) occurred when four people were in the room due to the size of the people relative to the size of the room. Four people had little space to move independently in the room and would often go from one merge to another quickly without giving the tracker a chance to get a clear shot of people leaving the group. The remainder of the errors result from three of the people in one sequence having similarly-colored clothing. Not surprisingly, when people look the same, the tracker is more likely to suffer from a post-merge assignment error.[5]

[5]E.g. One error is caused by one person wearing a navy and white shirt and navy pants and another person wearing navy pants and a white jacket with black stripes. The color feature, histogram intersection, uses no spatial information, so post-merge assignment errors between these two people are incurred on occasion.

		Post-Merge Assignment Errors			
In Room	Merges	1 stage	2 stage	5 stage	10 stage
2 people	4	3	0	0	0
3 people	54	22	3	0	0
4 people	48	19	17	14	12

Figure 5: The number of people mistakenly assigned after a merge for the 40 minutes of test video for window sizes of 1 (instantaneous decision), 2 (1s), 5 (2.5s), and 10 (5s).

Sequences	Min. edges per stage	Max. edges per stage	Ave. edges per stage
2 people	1	36	7
3 people	1	5398	780
4 people	1	50047	4290

Figure 6: Using distance pruning substantially reduces the number of links the DP algorithm must traverse from the max number (for our dataset) to the average number, improving performance.

More sophisticated features that use body shape [6] might alleviate some of these ambiguities.

The segmentation and proximity errors, counted in Figure 4, did not impact tracking assignment. These errors result in small shifts in position – jitter – during merge events.

4.3 Complexity

The search complexity can be reduced by pruning the graph. Instead of simply assigning a large cost when an object match requires an unnaturally large movement of the object, the links can be removed from the graph entirely. In the best case, when all objects are far apart in the scene, the graph can be reduced down to a single chain of nodes. In the worst case, however, the cost remains $O(nb_{max}^{2p_{max}})$ as described in Section 2.2. Figure 6 shows the effect of pruning on the 40 minutes of test video: in the case of 4 people, the number of links the DP algorithm must consider is reduced by 91%.

The results in Figure 5 for the four person runs were obtained with distance pruning, which ensured that samples could be gathered at 500 ms intervals. The algorithm does accommodate a variable sampling rate, allowing for processing of data only when it is most informative. This affords another opportunity for limiting complexity by only adding stages without "fragmented" blobs that are not either definitely split or definitely merged. Avoiding processing frames just before a merge or after a split event can reduce spurious blobs that increase b_{max}.

5 Summary

The algorithm presented here is meant to augment, not replace, existing multi-person and object tracking algorithms. We have shown how to efficiently layer temporal consistency constraints over inter-frame feature matching processes, which can lead to more robust tracking performance by explicit reasoning about object interactions. The algorithm can be implemented in real-time for a useful class of tracking problems. We believe that this approach can be extended to incorporate more features and contextual constraints.

References

[1] J. Arnold, S. Shaw, and H. Pasternack. Efficient target tracking using dynamic programming. *IEEE Trans. on Aerospace and Electronic Systems*, 29(1), January 1993.

[2] D. Beymer and K. Konolige. Real-time tracking of multiple people using stereo. In *Proc. of the IEEE Frame Rate Workshop*, Corfu, Greece, 1999.

[3] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenswalb. Plan-view trajectory estimation with dense stereo background models. AI Memo 2001-001, MIT, February 2001.

[4] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic programming for detecting, tracking, and matching deformable contours. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17:294–302, March 1995.

[5] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. Computer Vision and Pattern Recognition*, pages 22–29, 1998.

[6] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8), August 2000.

[7] D.P. Huttenlocher, J.J. Noh, and W.J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proc. Int. Conf. Computer Vision*, pages 93–101, May 1993.

[8] V.S.S. Hwang. Tracking feature points in time-varying images using an opportunistic selection approach. *Pattern Recognition*, 22(3):247–256, 1989.

[9] S.S. Intille and A.F. Bobick. Closed-world tracking. In *Proc. of the Fifth Int. Conf. on Computer Vision*, pages 672–678, June 1995.

[10] S.S. Intille, J.W. Davis, and A.F. Bobick. Real-time closed-world tracking. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 697–703, Los Alamitos, CA, June 1997. IEEE Computer Society Press.

[11] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. of Computer Vision*, 29(1):5–28, 1998.

[12] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. European Conf. Computer Vision*, volume 1, pages 189–196, May 1994.

[13] J. Krumm, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for EasyLiving. In *Proc. of the 3rd IEEE Int. Work. on Visual Surveillance*, July 2000.

[14] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc. Int. Conf. Computer Vision*, pages 572–578, 1999.

[15] C. Rasmussen and G. Hager. Joint probabilistic techniques for tracking multi-part objects. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 16–21, 1998.

[16] R. Rosales and S. Sclaroff. 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 117–123, June 1999.

[17] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7:11–32, 1991.

[18] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[19] Y. Yacoob and L.S. Davis. Learned models for estimation of rigid and articulated human motion from stationary or moving camera. *Int. J. of Computer Vision*, 36(1):5–30, 2000.