DETECTION AND TRACKING OF
MULTIPLE TARGETS IN CROWDED SCENES

JOSEPH MICHAEL DEGOL
Spring 2012

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computer Engineering and General Mathematics
with honors in Computer Engineering.

Reviewed and approved* by the following:

Dr. Robert Collins
Associate Professor
Thesis Supervisor

Dr. Lee Coraor
Associate Professor
Honors Adviser

* Signatures are on file in the Schreyer Honors College.
Abstract

In this work, we propose a method for detecting and tracking groups of individuals in crowded scenes. Our method begins with a detection phase where we use KLT to extract feature points from the scene. Feature Points that are stationary or have infeasible trajectories are then pruned leaving only points representing moving people. These points are then connected to form a graph structure using Delaunay Triangulation. Edges of the graph are assigned weights based on the proximity and coherency of motion of connected feature points. Edge weights above a given threshold are then pruned, resulting in blobs of connected feature points. These blobs represent our groups of individuals that we want to track; this concludes the process of detecting groups of individuals. After the detection phase is complete, we use mean shift belief propagation (MSBP) to track the feature points. This method allows connected feature points to influence the likelihood estimation for each point, resulting in an estimation that is based on the movement of the entire group of individuals. In the past, MSBP has been used for tracking of rigid lattice graphs, but not for general graph structures. Thus, much of the novelty of this approach comes from the application of MSBP to track connected feature points which are represented by general graph structures. Additionally, this paper demonstrates a vision system that combines established techniques in a novel way for both detection and tracking.
# Table of Contents

1. **Introduction** .................................................. 1

2. **Detecting People in Crowded Scenes** .................. 3
   2.1 Detecting Features ........................................ 3
   2.2 Clustering Features into People ......................... 4

3. **Tracking People in Crowded Scenes** .................. 6
   3.1 Background Mathematics .................................. 6
      3.1.1 Markov Random Fields .............................. 6
      3.1.2 Belief Propagation ................................ 7
      3.1.3 Sum-Product and Max-Product .................... 8
   3.2 Mean Shift Belief Propagation .......................... 9

4. **Results** ..................................................... 11
   4.1 Phase 1: Tracking with Gaussian Sampling .......... 11
   4.2 Phase 2: Including Mean Shift Belief Propagation .. 12

5. **Conclusion** .................................................. 15

**References** .................................................... 16

**Academic Vita** ................................................ 17
1 Introduction

Tracking a large number of individuals in a crowded scene is an interesting problem that presents many challenges. When people are within close proximity of each other, partial occlusion can frequently occur. This occlusion makes it difficult to differentiate individuals within a group, and makes standard appearance and model based detectors inadequate methods of detecting many people in crowded scenes. Therefore, other techniques have been devised to detect individuals in crowds. Ge [2] uses background subtraction coupled with Reversible Jump Markov Chain Monte Carlo (RJMCMC) to extract a set of overlapping rectangles that best represents the desired targets. This method, however, becomes less effective as crowd density increases. Sugimura [1] and Rabaud [3] use the KLT tracker to extract keypoints from the scene. They then consider the correlation of each feature point to its neighbors in a temporal and spatial framework. Using this correlation, similar feature points are grouped together to form individuals.

Once groups of individuals have been identified, the process of tracking them all simultaneously can be another challenging problem. In sparse environments, individuals tend to have very unique motion characteristics. This can make the challenge of tracking multiple targets easier. In densely populated environments, however, each individual’s motion is constrained by those around him or her. This tends to make the motion of neighboring people similar, making it more difficult to track each person. One possible solution demonstrated by [1] is to consider gait frequency and local appearance patches as a way of differentiating individuals within close proximity of each other. Another possibility is to track the entire group, making changes to the group composition as people enter or leave. Ge [2] and French [4] incorporate social effects and Gestalt psychology into their methods in an effort to define a group structure. By considering common goals, appearance similarities, long term proximity, and motion coherence, they attempt to group individuals into small groups. Another approach by Park [7][8] uses belief propagation to pass messages between each grouped object to estimate the joint likelihood of each tracked point. The mean shift algorithm is then used to perform hill climbing to find a new predicted state for each point. This process is repeated until convergence is achieved to within a small threshold.

An inherent challenge with tracking changing groups is handling the creation and removal of links between individuals as they enter or leave the group. Lao’s work [5] addresses this problem. An appropriate object for a group meets three criteria: 1) frequent co-occurrence with others, 2) consistent
motion correlation, and 3) suitability for tracking. Then, using belief propagation within a Markov random field model, messages are passed between the objects to estimate likelihood. Finally, Grabner’s method [6] uses Generalized Hough Transform (GHT) voting to differentiate between strongly and weakly correlated motion.

In this paper, we present a method for detecting and tracking multiple targets in crowded scenes. This method begins by extracting feature points in each frame of the video sequence using KLT. From the feature point data, a graph is built using Delaunay Triangulation to connect all the feature points of the image. Edge pruning is then performed based on spatial proximity and motion coherency over an 11 frame window of time. This results in groups of feature points representing groups of individuals. This is our starting point for tracking. Data Driven Mean Shift Belief Propagation is then used to track the groups over time.
2 Detecting People in Crowded Scenes

The first phase of our approach is to detect people and group them into reasonable clusters for tracking. This task is performed by detecting features and then grouping them based on proximity and coherency of motion. These steps are explained in this section, and we show how this process works to transform a crowded scene into many trackable groups.

2.1 Detecting Features

KLT is a corner detection and tracking method that relies on a second moment matrix of image gradients to extract important locations in the image called feature points. Over the duration of the video sequence, KLT will add new feature points as needed to keep a constant number of extracted points for each frame. We begin our process by using the KLT tracker to extract many feature points over the length of the entire video sequence [1]. Figure 1-a shows the results of KLT on a single frame of a video sequence of Beaver Stadium after a Penn State football game. Notice in the figure that...
many points were generated that correspond to static objects in the scene. In addition, some feature points do not properly correlate to any meaningful objects in the scene. Therefore, the next step is to prune these undesired feature points.

After feature points have been extracted, we analyze their trajectories within a time window and prune trajectories that demonstrate unreasonably high velocities or very short durations [1]. In addition, we prune trajectories that remain stagnant because they typically represent background objects or individuals that are not moving; both of which are objects we are not trying to track [1]. Figure 1-b demonstrates our pruning process on the feature points extracted from the frame as shown in Figure 1-a. Notice how feature points along doorways, trees, benches, and other static objects are removed, leaving only the feature points of people we seek to track. The next step is to cluster these pruned feature points into groups.

2.2 Clustering Features into People

Using the set of pruned feature points, we construct our graph structure using the feature points as the nodes and by applying Delaunay Triangulation to obtain the edges. Figure 1-c shows the result of this step on the pruned feature points from Figure 1-b. Notice that the pruned feature points from Figure 1-b are nodes of our graph and edges were generated to link the nodes. In this current state, however, there are many edges connecting points that should not be grouped together.

The next step is to prune edges of our graph that are a poor representation of similar groups for tracking. For example, edges connecting feature points that are a large distance apart are probably not part of the same group and thus do not need an edge between them. To execute this pruning procedure, we assign weights to each edge. This weight is calculated based on two metrics: (1) spatial proximity and (2) coherency of motion[1].

Spatial proximity (1) is a legitimate metric because any two individuals who are walking close together and remain close together over a period of time should be tracked as a group. Conversely, if the spatial proximity is large, then these two feature points should probably not be grouped together. Spatial proximity, denoted $e^{xy}_{prox}$, is calculated by taking the maximum distance between the two points over a window of their trajectories [1].

The other metric used to determine edge weight, denoted $e^{xy}_{coh}$, is coherency of
motion. Coherency of motion (2) represents the similarity of the trajectories of the two individuals. It makes sense then to group individuals with similar motion. Coherency of motion is calculated by taking the standard deviation of the distance between the two trajectories over the time window.

Once these values are computed, the final weight for an edge connecting nodes x and y is calculated as

\[ e_{x,y} = e_{x,y}^{\text{prox}} \cdot e_{x,y}^{\text{coh}} \]  

(1)

where \( e_{x,y}^{\text{prox}} \) represents spatial proximity and \( e_{x,y}^{\text{coh}} \) represents coherency of motion [1]. Note that if either of the metrics is equal to zero, this will force the weight to zero regardless of the value computed for the other metric. To avoid this, a predetermined \( e_{\text{min}} \) value is used to replace any metric calculation equal to zero [1]. Thus, equation (1) becomes

\[ e_{x,y} = \min(e_{\text{min}}, e_{x,y}^{\text{prox}}) \cdot \min(e_{\text{min}}, e_{x,y}^{\text{coh}}). \]  

(2)

Finally, we prune any edges that have edge weights above a given threshold; this results in small groups of feature points connected by edges. Figure 1-d shows the results of this edge pruning process on the graph structure of Figure 1-c.

Figure 2: Zoomed View of Sub-Graphs of Feature Points

The end result of this process is a set of clustered sub-graphs corresponding to individuals and groups of individuals moving in the scene. Figure 2 above shows a zoomed in view of these clustered sub-graphs.

The next step is to track these sub-graphs as they move through the scene.
3 Tracking People in Crowded Scenes

The second phase of our approach is to track the groups of people over time by considering the trajectory of each individual and the trajectory of surrounding grouped individuals. This is done by applying a method called Data Driven Mean Shift Belief Propagation (DDMSBP) to our general graph structures.

3.1 Background Mathematics

3.1.1 Markov Random Fields

A Markov Random Field (MRF) can be thought of as a graph of random variables $X$ where the conditional probability of each variable depends only on neighboring variables in the graph. For example, in a chain-like time series graph, the state of a random variable at time $\tau$ depends only on the variables at time $\tau - 1$ and $\tau + 1$. If it is a causal (directed) time series graph, the state at time $\tau$ depends only on variables at time $\tau - 1$. This is also sometimes referred to as the Markov property.

An MRF can be represented as an undirected graph $G = (x_i, e_{ij})$ where $x_i$ represents a random variable from the set $X$ and and edge $e_{ij}$ represents a statistical dependency between random variables $x_i$ and $x_j$ from $X$. Figure 3 demonstrates a basic MRF with five random variables.

![Figure 3: MRF With Five Random Variables](image)

One way to calculate the joint distribution of an undirected graph is to factor it based on the number of cliques in the graph. A clique in graph theory is a subset of vertices such that every two vertices in the subset are connected.
by an edge. So, to factor a graph $G$ with $n$ cliques using this method, the formula becomes

$$P(x_1, x_2, \ldots, x_n) = \frac{1}{Z} \prod_{k=1}^{n} \psi_{C_k}(x_i \in C_k)$$  \hspace{1cm} (3)$$

where $Z$ is a normalizing constant and $\psi_{C_k}$ is the clique potential of a clique $C_k$ in graph $G$ [7]. Note that clique potentials are functions that specify correlations between the values of the random variables in a clique.

Now, let us consider the MRF model shown in Figure 3. If we want to calculate the marginal density $p(x_1)$ without considering the dependencies (edges) of the graph, we must resort to

$$P(x_1) = \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} P(x_1 x_2 x_3 x_4 x_5)$$  \hspace{1cm} (4)$$

If we want to compute this for $V$ variables, each having $S$ possible states, we have $S^G$ possible states of $X = \{x_1 \ldots x_5\}$, resulting in exponential computation times and storage requirements [7]. Fortunately, belief propagation offers a faster solution to this computation.

3.1.2 Belief Propagation

Belief propagation allows us to compute the marginal distribution in a much faster way by considering the dependencies of the graph. These dependencies allow us to factor the joint distribution based on the clique potentials as shown:

$$P(x_1 x_2 x_3 x_4 x_5) = \frac{1}{Z} \psi(x_1, x_2) \psi(x_2, x_3) \psi(x_2, x_4) \psi(x_4, x_5)$$  \hspace{1cm} (5)$$

Thus, our computation of $p(x_1)$ then becomes

$$p(x_1) = \frac{1}{Z} \psi(x_1, x_2) \psi(x_2, x_3) \psi(x_2, x_4) \psi(x_4, x_5)$$  \hspace{1cm} (6)$$

Note now that we can use the fact that multiplication is distributed over addition to further factor our equation. Thus $p(x_1)$ becomes

$$p(x_1) = \frac{1}{Z} \sum_{x_2} \left( \psi(x_1, x_2) \sum_{x_3} \psi(x_2, x_3) \sum_{x_4} \left( \psi(x_2, x_4) \sum_{x_5} \psi(x_4, x_5) \right) \right)$$  \hspace{1cm} (7)$$
This factorization procedure is the foundation of belief propagation.

In practice, belief propagation is an iterative process that passes messages between nodes in order to compute likelihood estimates for each node. In general then, the marginal density \( p(x_1) \) computed by belief propagation is

\[
p(x_1) = \frac{1}{Z} \phi(x_i, y_i) \prod_{j \in \mathcal{N}(i)} m_{ji}(x_i)
\]

where \( m_{i\rightarrow j} \) is a message passed from node \( i \) to node \( j \). This message \( m_{i\rightarrow j} \) is defined as

\[
m_{i\rightarrow j}(x_j) = \sum_{x_i} \phi(x_i, y_i) \psi(x_i, y_j) \prod_{s \in \mathcal{N}(i) \setminus j} m_{s\rightarrow i}(x_i)
\]

Using this method, our computation is greatly reduced from \( O(S^V) \) to \( O(S^2V) \). With pair-wise functions, message passing between two nodes \( i \) and \( j \) is \( O(S^2) \) where \( S \) is the number of states for the random variables at nodes \( i \) and \( j \). Note that there are \( V-1 \) possible links, resulting in a total computation and storage requirement of \( O(S^2V) \) \[7\].

In this work, we are attempting to apply belief propagation to graphs with loops. This is typically referred to as loopy belief propagation. While loopy belief propagation has been used for many years with great success, one of the problems with loopy BP is knowing if it will converge \[7\]. Currently, precise conditions for convergence of loopy belief propagation are unknown. In order to handle this, our algorithm continues until convergence or until a maximum number of iterations has been reached.

### 3.1.3 Sum-Product and Max-Product

The sum-product algorithm is used to take a joint distribution \( P(X) \) and efficiently compute the marginal distribution over any component variables \( x_i \in X \). The message \( m_{i\rightarrow j}(x_j) \) passing from node \( i \) to \( j \) at each iteration is given by

\[
m_{i\rightarrow j}(x_j) = \sum_{x_i} \phi(x_i, z_i) \psi(x_i, x_j) \prod_{s \in \mathcal{N}(i) \setminus j} m_{s\rightarrow i}(x_i)
\]

where \( \mathcal{N}(i) \setminus j \) means all neighbors of node \( i \) except \( j \). Here, \( \phi_i \) is called the joint compatibility and \( \psi_{ij} \) is the pairwise compatibility function \[7\]. The
The final sum-product solution is then given by computing an expectation over the possible state of

\[ x_i = \sum_{x_i} x_i b_i(x_i); \quad b_i(x_i) = k \phi_i(x_i, z_i) \prod_{s \in \mathcal{N}(i)} m_{s \rightarrow i}(x_i) \]  

(11)

where \( b_i \) is the belief at node \( i \) and \( k \) is a normalizing constant.

While the sum-product algorithm lets us compute the marginal distribution over any \( x_i \in X \) for a joint distribution \( P(X) \), the max-product algorithm allows us to find the states of variables that yield the maximum probability.

### 3.2 Mean Shift Belief Propagation

After determining the feature points and grouping them into a graph structure, we use mean shift belief propagation to track the group as it moves through the scene. This process begins by computing the normalized cross correlation (NCC) between an image patch in the initial frame and the next frame. A Gaussian sampling is then performed and the NCC scores corresponding to each sampled point are used to compute a new weighted mean and variance. The new mean is given by

\[ \mu_{\text{new}} = \frac{\sum_{i=1}^{n} \text{NCC} (P_i, A) \times P_i}{\sum_{i=1}^{n} \text{NCC} (P_i, A)} \]  

(12)

and the new variance is given by

\[ \sigma^2_{\text{new}} = \frac{\sum_{i=1}^{n} \text{NCC} (P_i, A) [P_i - \mu_{\text{new}}]^T [P_i - \mu_{\text{new}}]}{\sum_{i=1}^{n} \text{NCC} (P_i, A)} \]  

(13)

where \( P_i \) represents a Gaussian sampled point from the next frame of the sequence, \( A \) represents the original feature point, and \( \text{NCC} (P_i, A) \) represents the normalized cross correlation score between image patches centered at the points \( P_i \) and \( A \).

This new mean represents the predicted location of a feature point in the next frame. This process can be repeated until the variance is below some threshold, resulting in an estimate of the feature point’s next location. Figure 4 demonstrates this process for a single feature point in our frame. Note also that this procedure can be used to track the feature points individually but does not take the grouping into account during tracking.
In order to consider the group structure during our tracking phase, we utilize mean shift belief propagation (MSBP). The belief propagation mechanism passes messages through the graph representing our connected feature points and returns a belief surface representing the marginal densities for each feature point. Then, mean shift is used to find the maximum of the belief surface. Figure 5 shows a depiction of a belief surface and the hill climbing procedure performed by mean shift. This procedure results in a new mean that represents the predicted location of the feature point in the next frame. This new mean takes into account both the influence of the feature point’s movement and the movement of the neighboring grouped points.
4 Results

We tested our approach on data taken from an overhead camera at Penn State’s Beaver Stadium at the conclusion of a football game. This testing was done in two phases. The first phase was to track the people leaving the game by Gaussian sampling on each point and calculating new means as described in section 3.2. The second phase of testing incorporated the edges connecting neighboring points and MSBP.

4.1 Phase 1: Tracking with Gaussian Sampling

We tested our method on a 149 frame sequence of people exiting Beaver Stadium after a Penn State football game. The first phase of our tracking method was to use Gaussian sampling with normalized cross correlation to calculate likelihoods for future feature point locations. This approach is described in more detail in Section 3.2 and demonstrated by Figure 6 and Figure 7.

Figure 6: Tracking Two Feature Points
Figure 6 above shows the result of this method for two neighboring feature points. The figure is split into 8-frame intervals to demonstrate the tracking procedure over time. Note that by frame 97 the tracker has lost the desired targets; because of this, the remainder of the sequence after frame 113 has been truncated.

Our phase I process is again demonstrated in Figure 7. In this figure, however, we have two additional feature points that we are considering and the sequence is split into 4-frame intervals. While the images shown only use 2 and 4 feature points, this process works for any number of feature points and these parameters and images were chosen to illustrate the overall results.

![Figure 7: Tracking Four Feature Points](image)

Notice how, in Figure 7, the additional feature points begin to wander from the target much sooner than the original points did in Figure 6. This could happen for several reasons including image noise, occlusion, and abrupt motion changes. However, while some of the points begin to wander from the target, not all the points lose the target. This is a useful fact for our phase II tracker that considers how the grouping of points can be used to influence likelihood estimates of each feature point’s location. In order to use the feature point groupings in our process, we incorporate MSBP into our algorithm.

### 4.2 Phase 2: Including Mean Shift Belief Propogation

The second phase of our tracking method added MSBP into the likelihood calculations of our phase I tracking process. This was described earlier in
section 3.2. We again demonstrate our process on the 149 frame sequence of people exiting Beaver Stadium after a Penn State football game.

Figure 8: Tracking Two Feature Points with MSBP

Figure 8 above demonstrates this method on two neighboring feature points. The figure is split into 10-frame intervals to demonstrate the tracking procedure over time. Note that, as opposed to phase I tracking, we were able to successfully track these feature points over the entirety of the sequence. This is a noticeable improvement over the results produced by our phase I tracking method.

We then considered additional feature points. Figure 9 below shows how our phase II tracking method performed with four neighboring feature points. We also have this figure split into 10-frame intervals. Notice how our phase
II tracker was able to successfully follow the points throughout the entire video sequence. This is a significant improvement over the results demonstrated by the phase I tracker which treated four neighboring feature points independently.

These results demonstrate that the grouping of neighboring feature points is a useful tool for tracking. In particular, our results show that the additional information gathered by using MSBP to consider linked feature points is able to noticeably improve the tracking results. These are promising results that provide preliminary validation of our method and motivation for further investigation into the effectiveness of MSBP for tracking multiple feature points that are grouped into a general graph structure.

Figure 9: Tracking Four Feature Points with MSBP
5 Conclusion

We have proposed a method for detecting and tracking multiple individuals in crowded scenes. Our proposed detection method begins by utilizing the KLT tracker to find and track feature points in our scene. These feature points are then pruned by considering each one’s trajectory and removing those points with trajectories representing static objects. These pruned features are then connected into a graph structure through the use of Delaunay Triangulation. We then prune away edges by considering the correlation of motion and spatial proximity between each connected set of feature points. This leaves us with groups of feature points representing individuals and groups of individuals for tracking.

Our proposed tracking method begins by Gaussian sampling around each feature point and then performing normalized cross correlation to calculate an estimate of the point’s location in the next frame. We ran tests demonstrating that this is a sufficient method for tracking each feature point separately, but it does not allow grouped feature points to influence each other’s location estimate. In order to incorporate the grouping structure into our tracking procedure, we used Mean Shift Belief Propagation, and we ran experiments demonstrating how the grouping was able to improve the results of the multi target tracking.

Future work includes extending the algorithm to allow for breaking and joining edges between feature points over time. For example, as two connected feature points begin to diverge from each other in a scene, it becomes necessary to eventually break the link between them. This could happen if two people were walking together originally, and then, after some time, they head off in two separate directions. Conversely, if two feature points do not originally have a link, but then begin to follow similar trajectories within close range of each other, it may be appropriate to add an edge connecting them. A realistic example of this would be when two friends are walking from separate locations to meet each other. Originally, there is no link between them because they are far away from each other and walking towards a location from different directions. Once they reach their meeting location, however, they begin to walk together and, thus, share a similar trajectory.
References


Education

B.S. Computer Engineering With Schreyer Scholar Honors, May 2012
B.S. Mathematics, May 2012
Physics Minor, May 2012

The Pennsylvania State University, University Park, PA 16802
Computer Engineering Student Marshal

Striving for Ph.D in Computer Science, Starting Fall 2012
University of Illinois, Urbana-Champaign

Research Interests

Robotics, Artificial Intelligence, Computer Vision, Machine Learning, HCI

Publications


Current Academic and Research Experience

Working in Penn State Comp Vision Lab with Dr. Robert Collins, Fall 2010 – Present
- Current tracker can track multiple targets in a lattice formation
- Goal: to extend current algorithm to track multiple targets in any formation
- Final Result: Schreyer Honors College Thesis

Past Academic and Research Experience

TI for the iPhone Programming Course with Dr. John Hannan, Fall 2011
- Create assignments and hold office hours for students to seek help
- Create and give lectures about iPhone programming concepts

Participated in SROP at University of Michigan with Dr. Ryan Eustice, Summer 2011
- Learned computer vision concepts: camera calibration and planar object detection
- Implemented a planar pattern detector that used Fern based classification
- Tested the tracker with video footage from an autonomous quadrotor
- Presented my work at CIC Summer Research Conference at Ohio State

Participated in SURE program at Georgia Tech with Dr. Patricio Vela, Summer 2010
- Conducted research dealing with data clustering of flight patterns
- Learned machine learning techniques such as PCA, KPCA, mean shift, k-means
- Learned forward and inverse kinematics for maneuvering a robotic arm
- Participated in talks and lectures dealing with effective research strategies
- Gave a presentation and wrote a paper about my summer research

Participated in REU at Virginia Tech with Dr. Scott McCrickard, Summer 2009
- REU with computer science department in HCI
- Used Flex, ActionScript, and WAMP to create a storyboarding tool web application
- Became familiar with research in a university environment
- Participated in talks and lectures dealing with effective research strategies
- Presented my work at a symposium
- Paper published in CHI 2011 titled “Don’t drop it! Pick it up and storyboard”

Worked in Penn State physics lab with Dr. Damon Resnick, Fall 2008, Spring 2009
- Used lab equipment to measure magnetic effects on different samples
- Became acquainted with LabVIEW which we used to graph hysteresis loops
- Allowed me to gain significant hands on experience in a lab environment

Honors, and Awards

Penn State Schreyer Honors Scholar, Fall 2009 – Present
Dean’s List, All Semesters
Penn State Altoona Honors Member, Fall 2007 – Spring 2009
Lockheed Martin Scholarship Award, Fall 2009
President’s Freshman Award, Fall 2007 – Spring 2008

Extracurricular Activities

Penn State Altoona NCAA Men’s Soccer, Fall 2007 – Spring 2009
Residential Assistant: Nittany Pointe Housing Communities, Fall 2008 – Spring 2009