Efficient Mean Shift Belief Propagation for Vision Tracking

Minwoo Park†, Yanxi Liu†* and Robert T. Collins†
†Department of Computer Science and Engineering, *Department of Electrical Engineering
The Pennsylvania State University, University Park, PA 16802
{mipark,yanxi,rcollins}@cse.psu.edu   http://vision.cse.psu.edu/msbp.htm

Abstract

A mechanism for efficient mean-shift belief propagation (MSBP) is introduced. The novelty of our work is to use mean-shift to perform nonparametric mode-seeking on belief surfaces generated within the belief propagation framework. Belief Propagation (BP) is a powerful solution for performing inference in graphical models. However, there is a quadratic increase in the cost of computation with respect to the size of the hidden variable space. While the recently proposed nonparametric belief propagation (NBP) has better performance in terms of speed, even for continuous hidden variable spaces, computation is still slow due to the particle filter sampling process. Our MSBP method only needs to compute a local grid of samples of the belief surface during each iteration. This approach needs a significantly smaller number of samples than NBP, reducing computation time, yet it also yields more accurate and stable solutions. The efficiency and robustness of MSBP is compared against other variants of BP on applications in multi-target tracking and 2D articulated body tracking.

1. Introduction

Many vision problems suffer from a multimodal likelihood and posterior, high dimensionality and inaccurate local evidence. However, when prior knowledge is well-encoded in a graphical model, the resulting constrained inference process becomes more accurate. Belief Propagation (BP) is a powerful tool for inference of marginal distributions in a graphical model. Because BP can approximately solve NP hard problems in polynomial time, it has been widely used in many areas. It is guaranteed to give an exact solution for acyclic graphs and has been empirically shown to give satisfactory results even on loopy graphs [4, 15, 16, 8].

As a sample vision problem using BP, consider tracking a group of people marching in formation (Figure 1). Here, each node in the graphical model corresponds to one person, and the hidden variables are the (x, y) location of that person in the image. Links between adjacent people specify spatial lattice constraints, which constrain the location of a person based on the locations of neighboring people. This constraint is specified by a distance based compatibility function. In addition to geometric constraints, the likelihood of a person being at a particular location in the image is measured by an appearance-based likelihood function that computes similarity between an intensity template model of the person and the image patch at that location.

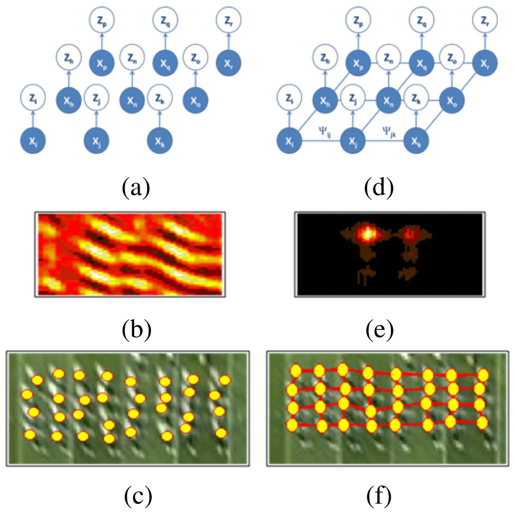


Figure 1. (a-c) Close spacing and similar appearance of people in low resolution video of a marching band yields a multimodal posterior distribution for the location of each individual when inference is based only on local evidence. (d-f) By encoding spatial geometric (lattice) constraints between people, the posterior distribution for each person's location becomes much more concentrated around a single mode. Note: (b) and (e) are positions for a single individual

As can be seen in Figure 1(a-c), when individuals are treated independently without neighbor compatibility links, the posterior distribution on location of any one person is highly multimodal. This is so because the posterior is dominated by the appearance likelihood function, which is multimodal due to the ambiguity of similar people at this resolution in this application. Compare this to Figure 1(d-f), where inference of the location of each person becomes unambiguous when the prior knowledge of the structured spatial layout of people is properly encoded into a graphical model.

The combination of graphical models and BP has proven to be effective in many vision problems such as stereo vision, multi-target tracking, and articulated body tracking [4, 12, 9, 8, 6]. However, some problems are not well-suited for belief propagation, because there is a quadratic increase in computation time with respect to the size of the hidden variable space, that is, the computation time is O(kn^2T) where k is the number of nodes, n is the size of the hidden variable space, and T is the number of iterations needed for convergence. BP is therefore no longer feasible in large hidden variable spaces, and the need for more efficient and robust algorithms arises. The goal of this paper is to develop an efficient belief propagation algorithm for vision tracking that produces accuracy comparable to discrete belief propagation. To validate the generality of the proposed method, our approach is applied to two challenging applications, multi-target tracking and 2D articulated body tracking.

1.1. Related Works

To improve the speed of inference using belief propagation, several approaches have been suggested. Ramanan & Forsyth [11] speed up BP by removing states that have low image likelihood value, thus reducing the size of input to the belief propagation framework. However, preprocessing is needed to compute the image likelihood for the entire space, which needs O(nT) pre-processing time, where n is the size of the hidden variable space and T is the time needed for evaluation of image likelihood. Moreover, this kind of pruning method is susceptible to error; once an important state is wrongly pruned in the pre-processing stage, the inferencing may not reach the correct solution. Coughlan & Shen [2] also speed up BP by state pruning. However the pruning method they use is dynamic quantization which allows addition and subtraction of states during the belief propagation process. The method of Coughlan & Shen has less risk than static pruning of the hidden variable space. However, neither approach is suitable for continuous hidden variable spaces, because they are based on DBP.

More recently, efficient BP methods for early vision have been developed in [3]. The computation time of discrete belief propagation (DBP) is reduced by several orders of magnitude using min convolution, bipartite graphs and multigrid methods. However, the use of min-convolution in [3] is only applicable because of the convexity of their compatibility function. Moreover, it is not feasible in high-dimensional spaces due to the need for uniform discretization.

To tackle problems with high dimensional continuous state spaces, several versions of continuous BP have been proposed recently [13, 7, 6]. By representing arbitrary density functions using particles, with each particle being the mode of a Gaussian in a mixture of Gaussian distribution, BP inference can be approximated for a continuous hidden variable space. It has been reported that 100 $\sim$ 200 Gaussians suffice for an arbitrary and accurate representation of the density [13, 6]. However, the standard NBP algorithm is slow due to the sampling process [6]. Recently, a more efficient version of NBP was proposed using sequential density estimation and mode propagation [6]. It is reported that this efficient NBP is 80 times faster than standard BP for tracking a 2D articulated body model.

1.2. Mean-Shift Belief Propagation

The methods discussed so far treat the entire hidden variable space as a whole. We have developed a more efficient and simple method called MSBP (Mean-Shift BP) that works iteratively with local samples and weights. We note that mean-shift is equivalent to finding a local mode within a Parzen window estimate of a density function, and use mean-shift as a non-parametric mode-seeking mechanism operating on weighted samples generated within the belief propagation framework. Geometrically, we can visualize this process as performing mean-shift on the implicit belief surface or marginal density generated by the belief propagation algorithm (Figure 2). Because the mean-shift algorithm only needs to examine the values of the belief surface within its local kernel window, we can avoid generating the entire belief surface, yielding great computational savings.

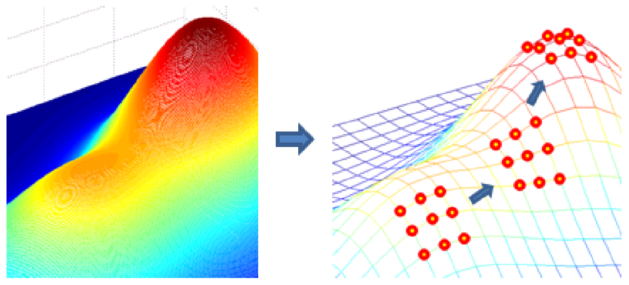


Figure 2. Illustration of a 2D marginal density (left) computed by the BP process and mean-shift hill-climbing on that belief surface (right). Only a small local grid of belief values within the mean-shift window need to be computed during any iteration, so the majority of the belief surface can remain implicitly defined (and thus not computed). Computational savings increase as the dimensionality of the belief surface increases.

Since a significantly smaller number of samples is needed than particle filtering, computation time is reduced as compared to NBP. If we were doing mean-shift on the weights and samples generated by image likelihood alone, then they might settle on a local mode that is of no interest, caused by clutter or noise. However, this is not likely in our case; although there might be many local modes in the image likelihood, the pairwise neighborhood compatibility constraints in BP can reduce false local modes in the posterior(Figure 1).

2. Mathematical Background

A Markov Random Field (MRF) specifies a factorization of the joint probability of a set $\mathbf{X}$ of random variables. An MRF can be represented as an undirected graph $\mathbf{G} = (\mathbf{N}, \mathbf{E})$, where each node in $\mathbf{N}$ represents a random variable in set $\mathbf{X}$ and each edge in $\mathbf{E}$ represents a statistical dependency between random variables in $\mathbf{X}$. The set of state values of the collection of variables is called the hidden variable space, since it is not observable directly but must be inferred through measurement and statistical dependence information. Brute force evaluation of the marginal distribution of a single random variable in the set $\mathbf{X}$ leads to $O(n^k)$ computation time where k is the number of nodes in the graph and n is the size of the hidden variable space, and thus it is $O(kn^k)$ for all random variables in a set $\mathbf{X}$. Computation of the marginal distribution of a single random variable x_j is given by

$$p(x_j) = \sum_{1} ... \sum_{N \setminus j} = p(x_1, x_2, ..., x_N, z_1, z_2, ..., z_N) \quad (1)$$

where z_i is an observable or measurable quantity given the state of x_j and $N \setminus j$ means the nodes in set N except for node j. However, the joint probability over the state x and measurement z in a MRF can be factored as

$$p(x_1, ..., x_N, z_1, ..., z_N) = \prod_{(i,j) \in E} \psi(x_i, x_j) \prod_{s \in N} \phi(x_s, z_s) \quad (2)$$

where Ψ and Φ are pairwise compatibility and joint compatibility functions, respectively. For vision problems, the joint compatibility function $\phi(x_j, z_j)$ is a function governing statistical dependency between x_i and z_i. The compatibility function $\psi(x_i, x_j)$ is a function governing the assumed geometric structure between neighboring nodes, x_i and x_j. For acyclic graphs (e.g. chains or trees), the above equations can be evaluated efficiently through belief-propagation. The term "belief" simply means a marginal probability that is computed approximately. Cost of computation is thus reduced from brute force evaluation of $O(kn^k)$ to $O(kn^2)$.

2.1. Discrete Belief Propagation

One of the advantages of BP on discrete spaces is that messages and compatibility functions can be expressed as arrays. Therefore, computation of message products and beliefs can be performed using only element-wise product of arrays and linear algebraic product of two arrays. There are two different message update rules:

2.1.1 Sum-Product rule

The sum-product rule corresponds to a Minimum-Mean-Square-Error (MMSE) solution. MMSE yields the correct posterior marginal probability for graphs without loops. Even in loopy graphs, it gives correct means for Gaussian processes [4]. The message $m_{i \to j}(x_i, x_j)$ passing from node i to j at each iteration is given by

$$m_{i \to j}(x_j) = \sum_{x_i} \phi_i(x_i, z_i) \psi_{ij}(x_i, x_j) \prod_{s \in N(i) \setminus j} m_{s \to i}(x_i) \quad (3)$$

where $N(i) \setminus j$ means all neighbors of node i except j, ϕ_i is the joint compatibility and ψ_{ij} is the compatibility function. The MMSE solution is given by computing an expectation over the possible state of

$$x_i = \sum_{x_i} x_i b_i(x_i), b_i(x_i) = k\phi_i(x_i, z_i) \prod_{s \in N(i)} m_{s \to i}(x_i) \quad (4)$$

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

2.1.2 Max-Product rule

The max-product rule corresponds to the Maximum a Posteriori (MAP) solution. Max-product gives the MAP estimate for graphs without loops and it finds a local maximum of the posterior even for non-Gaussian distributions [4]. The message $m_{i \to j}(x_i, x_j)$ passing from node i to j at each iteration is given by

$$m_{i \to j}(x_j) = \max_{x_i} \phi_i(x_i, z_i) \psi_{ij}(x_i, x_j) \prod_{s \in N(i) \setminus j} m_{s \to i}(x_i) \quad (5)$$

The MAP estimate of belief over the possible states of x_i is determined by

$$x_i = arg_{x_i} \max b_i(x_i) \quad (6)$$

3. Mean-shift belief propagation - MSBP

Instead of evaluating all the possible states of our hidden variable space, MSBP works within a local regular grid of samples centered at the predicted state. We first resample the continuous hidden variable space into a regular grid of samples for each node. This is a standard method in the context of density estimation [1]. This grid of samples becomes a hidden variable space within which BP message passing is performed to compute a weight (belief) for each sample. Once weights are computed, mean-shift on the samples at each node performs hill-climbing to reach a new predicted state for each node. A new discrete grid of samples is then generated centered on this predicted state, and the process repeats. The details of the new MSBP approach are given below; for concreteness we describe the messages, compatibility functions and joint compatibility functions in terms of our multi-target tracking application.

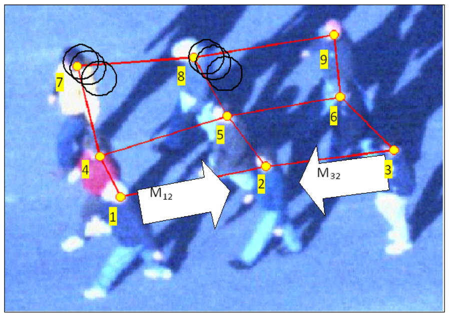


Figure 3. Lattice graph with 9 nodes. Several sample locations of the kernel are circled. Given a set of sample regions, observation and compatibility terms are measured by intensity correlation and distance between nodes.

3.1. Samples

Consider a lattice graph with 9 nodes as shown in Figure 3. The 2D hidden variable space $\mathbf{x_i}$ is the (x, y) location of the person represented by node i. Let $\dot{\mathbf{x_i}}, \{\dot{\mathbf{x}} = (x, y) \mid x \in x_1^i, ...x_n^i, y \in y_1^i, ...y_m^i\}$ be a set of local samples centered at the current predicted state for node i. The observation(joint compatibility) function is computed as normalized correlation between an appearance model of the person and the image patch at sample location (x, y) (Figure 3). Compatibility is a function measuring the pixel distance between nodes i and j. If the distance between adjacent people is close to a reference model distance, it gives a high value, while allowing some tolerance. A message from node 1 to 2 is simply interpreted as node 1's idea about where node 2 should be, based on the appearance observation made in node 1 and the compatibility between nodes 1 and 2. The observations and belief arrays at each node have size equal to the number of local samples we use to approximate the continuous hidden variable space.

3.2. Weight

[1] To generate the sample weights for mean-shift BP, a message is built with each weight computed by equation (7) below. Message passing from node i to node j according to the sum-product rule is given by

$$m_{i\to j}^{(n+1)}(\dot{\mathbf{x_j}}) = \sum_{\dot{\mathbf{x_i}}} \phi_i(\mathbf{x_i}, z_i)\psi_{ij}(\mathbf{x_i}, \mathbf{x_j}) \prod_{s\in N(i)\backslash j} m_{s\to i}^{(n)}(\mathbf{x_i}) \tag{7}$$

where $N(i) \setminus j$ means all neighbors of node i except j, ϕ_i is the joint compatibility function, and ψ_{ij} is the compatibility function. Note that sample points and summation are restricted to a discrete grid of points within the mean-shift.

After passing of messages according to the message update scheme, the corresponding observation is built with weights computed by the image likelihood function evaluated on each sample. The weight of each sample is computed by equation (8) and the belief of each node is constructed. For a graphical intuition of computing weights, please see Figures 4 and 2. Belief about the state of node j (probability of state of node j based on the evidence about j gathered from its neighborhood plus the image observation at node j) is given by

$$b_i(\dot{\mathbf{x_i}}) = k\phi_i(\dot{\mathbf{x_i}}, \dot{z}_i) \prod_{s\in N(i)} m_{s\to i}(\dot{\mathbf{x_i}}) \tag{8}$$

where k is a normalization constant.

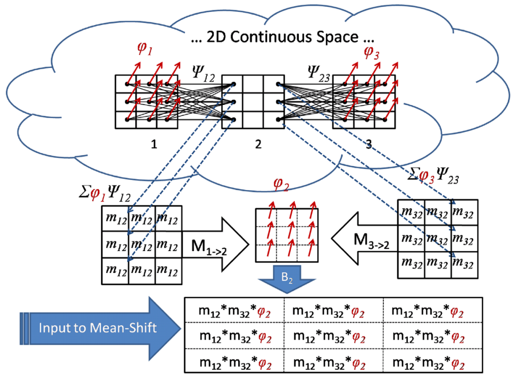


Figure 4. Illustration of the 2D MSBP procedure. Instead of evaluating a continuous hidden variable space, MSBP works with a regular grid of samples centered at the current predicted state at each node.

3.3. Mean-shift on weighted samples

Once the grid of samples and their weights are computed within the BP framework, the next step is to perform mean-shift on the set of weighted samples. Mean-shift is performed on the belief array for each node, in the sense that mean-shift is performed on a weighted grid of samples the same size as the belief array, where the weight of each sample is the entry contained in the belief array. The mean-shift result for each node gives an updated estimate of the mode of the marginal posterior. The procedure in section (3.1,3.2 and 3.3) is then repeated, centered at the new estimate, until its convergence. We compute a mean-shift update as

$$\mathbf{x^{(n+1)}} = \frac{\sum_i K(\mathbf{x_i} - \mathbf{x^{(n)}})b(\mathbf{x_i})\mathbf{x_i}}{\sum_i K(\mathbf{x_i} - \mathbf{x^{(n)}})b(\mathbf{x_i})} \tag{9}$$

where $b(\mathbf{x_i})$ is the weight computed in section 3.2, $\mathbf{x^{(1)}}$ is the initial predicted location, $\mathbf{x^{(n)}}$ is the estimated location after the n^{th} iteration, and $\mathbf{x_i}$ is a sample inside the mean-shift kernel K.

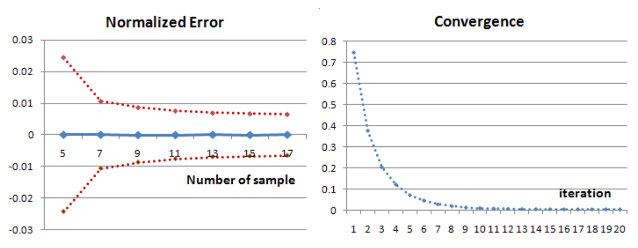
[1]The equations presented in this section are identical to BP except that MSBP works on the grid of local samples representing the hidden variable space at each node.

MSBP is efficient because it maintains a small set of samples, and it only has to compute a convolution value at a relatively small number of points on the path to a local posterior mode. Recall from Figure 2 that instead of looking at the entire marginal density, MSBP only needs to compute a local window of sample values on the density surface as it proceeds to the mode. Therefore, the density surface can be sampled more densely, as well as efficiently. This more detailed analysis of the surface leads to more accurate and stable solutions, with the same number of samples, as compared to DBP (via quantization of the space) or NBP.

3.4. Gaussian Graphical Models

In order to test our MSBP method on a loopy BP problem that has an analytic solution, we adopt the approach of Sudderth et al [13] and measure MSBP's performance on a Gaussian MRF with a 5×5 regular grid and randomly chosen inhomogeneous compatibility potentials. The goal of this test is to verify: (1) the convergence of MSBP; (2) whether MSBP is an unbiased mode estimator; and (3) the stability of the estimator. To assess (1), the number of iterations needed to achieve steady state is measured. For (2) and (3), $E(\tilde{m}_s)$ and $Var(\tilde{m}_s)$ are measured for 1000 trials respectively, where $\tilde{m}_s = (\hat{m}_s - m_s)/\sigma_s$, $\hat{m}_s$ is estimated marginal mean, m_s is the true mean, and σ_s is the true variance.

As can be seen in Figure 5(a), the MSBP algorithm gives unbiased estimates for the mean, and the standard deviation is very small. This shows that the estimator is stable and accurate. The variance of the estimate decreases as the number of samples approximating the local density surface increases. Although the number of iteration varies depending on the initial state of each node, MSBP enters into a steady state after about 20 iterations, as can be seen in the sample convergence plot in Figure 5(b).



(a) (b)

Figure 5. (a)MSBP performance on a GMRF with a 5×5 grid. The solid line is the mean estimate and the dashed lines are the standard deviation of the normalized error computed by $E(\tilde{m}_s)$ and $Var(\tilde{m}_s)$ (b) Sample convergence rate of the MSBP

4. Applications

4.1. Multi-target tracking

In [9], a lattice graph model is used to track multiple people marching in formation. Good results were achieved under lighting changes and occlusion. It was also demonstrated that tracking the formation using a netted collaborative tracker [14] exhibited higher error, which indicates that the use of graphical models can improve the performance of multi-target tracking. We adopt the method of [9] to test the robustness and efficiency of our proposed MSBP algorithm. For brevity, all the parameters used by our applications are explained in Section 4.3.

4.1.1 Compatibility (Graph model)

The compatibility function $\psi(\mathbf{x_i}, \mathbf{x_j})$ is a pairwise constraint function defined over the structure of the local neighborhood of $\mathbf{x_i}$, and is given by

$$\psi(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\beta \mid d(\mathbf{x_{ci}}, \mathbf{x_{cj}}) - d(\mathbf{x_{mi}}, \mathbf{x_{mj}}) \mid)^2 \quad (10)$$

where $\mathbf{x_{ci}}, \mathbf{x_{cj}}$ are random variables indicating the current position of two neighboring targets, $\mathbf{x_{mi}}, \mathbf{x_{mj}}$ are constants, indicating the initial geometric configuration of the targets, and $d(a, b)$ is the distance between a and b. This compatibility function allows some amount of change in distance between nodes by treating every link between nodes as an elastic band. Parameter β adjusts elasticity of the band in the graph model; if β is higher, the elasticity is lower, and vice versa.

4.1.2 Joint Compatibility (Observation model)

The observation image likelihood is given by a function of appearance similarity

$$\phi(\mathbf{x_i}, z_i) = \exp(-\alpha(1 - z_i)), z_i = NCC(T_i, I(\mathbf{x_i})) \quad (11)$$

where z_i is normalized correlation between a template T_i and $I(\mathbf{x_i})$, a patch of the same size centered at location $\mathbf{x_i}$ of the incoming image, and α is a parameter governing the influence of the observation z_i over the lattice model.

4.1.3 Experiment Results

We present experimental results on two sequences where multiple targets move in a lattice formation. "Crowd Sequence" is a medium resolution video taken from an airborne camera circling around a set of 9 marchers. The "Marching Band Sequence" is a low resolution video of a marching band, shot from a handheld camcorder during the halftime show of a football game. We have used 30x30 search regions, 100 particles and a 10x10 discrete kernel (circular) for DBP, NBP and MSBP, respectively. Because the “Crowd sequence” was taken by an airborne camera, it has large inter-frame movement as well as changes in viewpoint and lighting. Our choice of 30x30 search region for DBP was deemed to be the minimum size capable of handling the amount of pixel displacement between frames in the “Crowd Sequence” video.

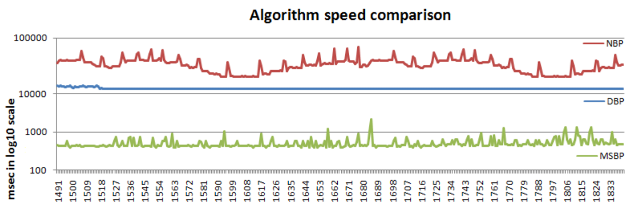


Figure 6. Timing results of DBP, NBP and MSBP on the Crowd sequence (note the log scale). Timings are based on 30x30 local regions, 100 particles and 69 samples used for DBP, NBP and MSBP, respectively.

The experimental timings in Figure 6 show that MSBP is more efficient than DBP and NBP. According to the theoretical computation time with respect to the size of the hidden variable space, NBP should generally be faster than DBP. However, in this example, the computation needed for Gibbs sampling offsets the efficiency of the NBP for the medium size of the hidden variable space. As can be seen in Figure 7, results of MSBP are better than DBP and NBP in terms of accuracy as well. In Figure 7, the accuracy of NBP is the worst. Although formal verification is difficult, we hypothesize that NBP’s approximate message updates and message products lead to accumulation of error in the estimates of the template position.

Each algorithm is also compared in terms of tracking stability. Liu et al [10] define a pair of regularity measurements for appearance and geometry. We adopt their appearance regularity measurement, A-score, to compare the stability of each tracking result. A-score is given by

$$A = \frac{1}{m}\sum_{i=1}^{m} std([T_1(i), \cdots, T_n(i)]) \qquad (12)$$

where T_n is a template taken from the estimated target location in frame n, and m is the number of pixels within each template T_n. A lower A-score means more appearance similarity among corresponding pixels in the region around the estimated target location. In our multi-target tracking example, a series of templates extracted by each algorithm over n frames forms the set of templates, $T_1, T_2, \cdots T_n$. As can be seen in Figure 8, stability of MSBP is better than NBP (see Figure 5 and [13]) and comparable to that of DBP.

Sample tracking results are shown in Figures 9 and 10. MSBP is used to track members of a marching band and gives smooth tracking result even in challenging situations

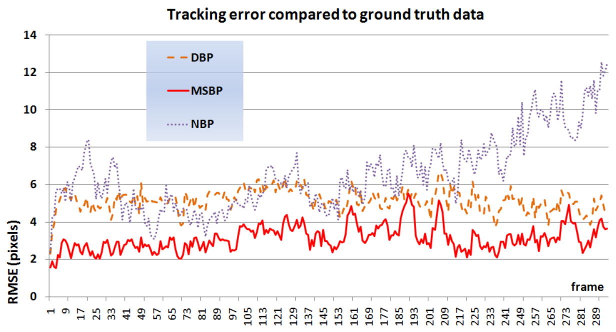


Figure 7. Tracking error compared to ground truth. MSBP is the most accurate and NBP is the worst. Due to the approximation error built-up during the tracking procedure, error is cumulative and tends to increase over time. MSBP has lower RMSE error than DBP and NBP.

A-score	seq1	seq2	seq3
NBP	31.76	36.01	-
DBP	16.11	19.98	17.57
MSBP	16.71	17.35	17.35

Figure 8. Quantitative A-score computed for 3 different sample sequences with over 300 frames length. Smaller scores are better.

without using any motion prediction model. For more information, please refer to our website.

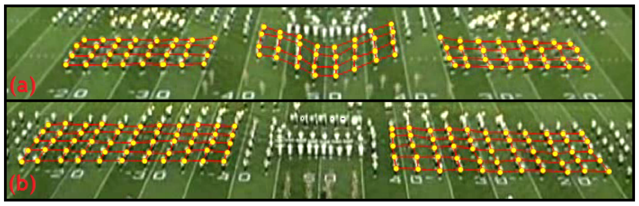


Figure 9. (a) Sample frame from tracking 128 marching band members using MSBP for over 600 frames (b) Sample tracking result of 110 marching band members using MSBP for over 300 frames. Smaller scores are better.

4.2. 2D articulated body tracking

We also tested our algorithm on a second tracking application where the goal is to track human body parts. Given an approximately known pose, a 2D view of the human body can be represented by a graphical model as shown in Figure 11. The model is similar to a loose-limbed body model where body parts are not rigidly connected but attracted to each other [6, 12]. Each body part is represented by a node in a tree graph and the hidden variables we want to infer are 3 dimensional $\mathbf{x_i} = (x, y, \theta)$, representing image translation (x, y) and in-plane rotation θ. The attraction between body parts is modeled by a compatibility function $\psi(\mathbf{x_i}, \mathbf{x_j})$ that tolerates small changes of distance

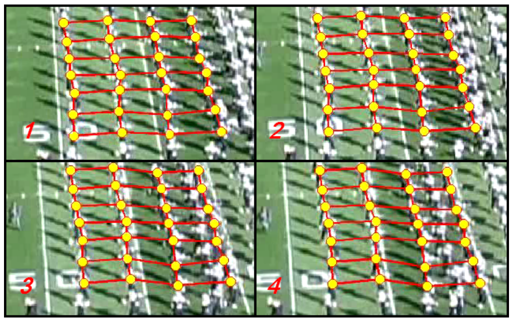


Figure 10. Tracking using MSBP is robust to occlusion and clutter. Without using any motion prediction model, the tracker can track multiple targets successfully under challenging situations such as this sequence where several rows of marchers pass each other.

between body parts and penalizes large changes. The observation image likelihood is given by a function of appearance similarity based on normalized correlation between a body part's intensity template model and the tracked body part's hypothesized location and orientation in the current image. The messages are propagated inward to node X_0 (Torso), then propagated outwards again, as can be seen in Figure 11(a).

4.2.1 Compatibility function (Graph model)

The compatibility function $\psi(\mathbf{x_i}, \mathbf{x_j})$ between adjacent body parts is a pairwise constraint function defined over the structure of the local neighborhood of $\mathbf{x_i}$, and is given by the same equation (10) used in the multi-target tracking application. This compatibility function allows some amount of change in distance between joints by treating every link between nodes as a spring. This is natural for any loose-limbed model because body parts cannot be separated.

4.2.2 Joint Compatibility (Observation model)

The observation image likelihood is given by the same function of appearance similarity as equation (11), where z_i is normalized correlation of a model template with a patch centered and rotated in the image according to $\mathbf{x_i}(x, y, \theta)$, and α is a parameter governing the influence of the observation similarity z_i over the body model.

4.2.3 Experimental Results

A C++ implementation of MSBP and DBP were compared using the CMU motion of body database [5]. The first frame has body parts labeled by hand according to the graphical model from Figure 11. Both algorithms are run on each subsequent frame until convergence. The current pose is used as the predicted initial pose for the next frame. Results

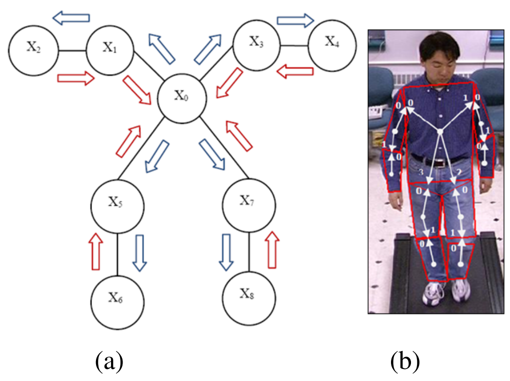


Figure 11. Graphical model representing a loose-limbed body model. Each body part is a node in the graph. The observation likelihood function is based on intensity patch correlation, and the compatibility function penalizes large distances between joints. (a) Messages are first propagated inward to node X_0 (Torso), then propagated outwards. (b) Joint vectors drawn from the center of each body part to its joints. For example, each forearm has one joint vector, each upper arm has two joints (shoulder and elbow) and the torso has 4 joints.

are shown for two sequences in Figure 13. MSBP outperforms DBP in terms of speed and gives equivalent results. Due to large 3D motion and weak bottom-up cues (only the template matching), MSBP exhibits some error. As can be seen in Figure 12, MSBP is about 300 times faster than our implementation of standard DBP on a hidden variable space of size 20^3 for a 486 by 640 image sequence.

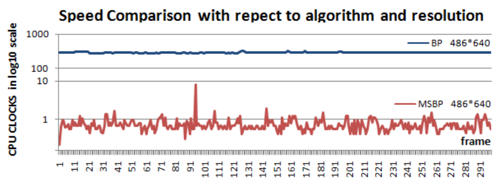


Figure 12. Speed comparison by algorithm and resolution, plotted on a log-CPU time scale. MSBP is approximately 300 times faster than standard BP with hidden variable space size of 20^3.

4.3. Parameter Selection

In the context of the mean-shift algorithm, the major parameter to be selected is the kernel size. There are automatic methods to select this, but in our application the kernel size was chosen as half the size of the initial template used for tracking. For the binsize of the kernel, we choose a size equal to the image pixel size. The binsize could be bigger than one pixel unit for efficiency, but we do not need a binsize smaller than one pixel because image measurements exist only at each pixel in our application of interest.

Application-specific parameters for the compatibility functions are chosen depending on the spacing between targets or maximum tolerance of change between nodes. The α parameter governing the tradeoff between model and data constraints is a parameter seen frequently in vision applications. A higher value means more sensitivity to the measurement data. We used $\alpha = 5$ for the first application and $\alpha = 50$ for the second. Parameter β governs the expected elasticity of the geometric placement of nodes; please refer to section 4.1.1 and 4.2.1. In the "Marching Band Sequence", we select β to be 0.1, because it is a low resolution sequence and target spacing is at most $5 \sim 6$ pixels. However the crowd sequence is a medium resolution sequence with larger spacing between each target, thus $\beta = 0.001$ is chosen. In the 2D articulated body tracking case, $\beta = 0.005$ is chosen.

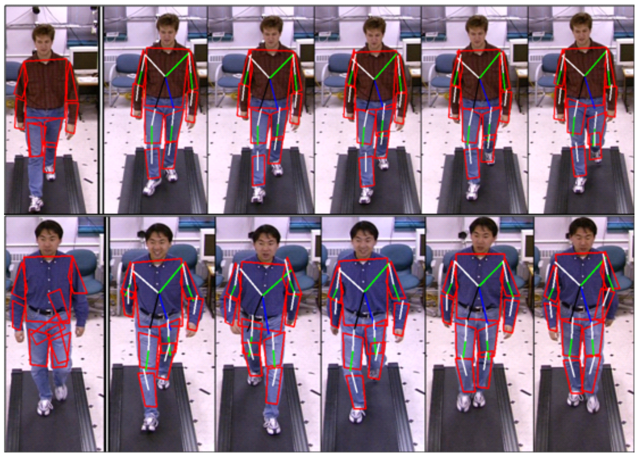


Figure 13. Leftmost column shows results of tracking individual body parts without a graphical body configuration model. All other frames show results of the MSBP algorithm. The sequence shown in the second row exhibits large 3D motion. In all cases, MSBP gives results that are visually equivalent to the standard BP algorithm (not shown here), while being 300 times faster.

5. Conclusion

We have developed an efficient belief propagation framework(MBSP) for vision tracking. Our method is compared to discrete belief propagation and non-parametric belief propagation in terms of tracking speed, stability and accuracy. To validate the general applicability of our proposed method, multi-target tracking(2D state space) and 2D articulated body tracking(3D state space) applications are shown. The results show that MSBP is efficient and robust, and outperforms DBP and NBP in terms of accuracy and stability. Computation time of MSBP in our 2D state space is $30 \sim 50$ times faster than DBP, while computation time of MSBP in our 3D state space is about 300 times faster, illustrating the efficiency of MSBP for applications with high dimensional spaces. Our future work will include theoretical validation of our proposed method and extension of the MSBP algorithm to larger and higher dimensional hidden variable spaces through applications such as 3D body tracking.

Acknowledgement This work was partially funded under NSF grant IIS-0535324 on Persistent Tracking, and NIH P50-AG05133.

References

[1] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[2] J. Coughlan and S. Huiying. Shape matching with belief propagation: Using dynamic quantization to accomodate occlusion and clutter. In *Computer Vision and Pattern Recognition Workshop, p180-180*, 2004.

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41, 2006.

[4] W. T. Freeman. Learning low-level vision. *International journal of computer vision*, 40(1):25, 2000.

[5] R. Gross and J. Shi. The CMU motion of body (mobo) database. June 2001.

[6] T. X. Han, N. Huazhong, and T. S. Huang. Efficient nonparametric belief propagation with application to articulated body tracking. In *Computer Vision and Pattern Recognition Vol. 1, p214-221*, 2006.

[7] M. Isard. PAMPAS: real-valued graphical models for computer vision. In *Computer Vision and Pattern Recognition, Vol. 1, p613-620*, 2003.

[8] S. Jian, Z. Nan-Ning, and S. Heung-Yeung. Stereo matching using belief propagation. *IEEE Trans Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[9] W. Lin and Y. Liu. A lattice-based MRF model for dynamic near-regular texture tracking. *IEEE Trans Pattern Analysis and Machine Intelligence*, 29(5):777–791, 2007.

[10] Y. Liu, W. Lin, and J. Hays. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics*, 23(3):368–376, 2004.

[11] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *Computer Vision and Pattern Recognition, Vol. 1, p467-474*, 2003.

[12] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard. Tracking loose-limbed people. In *Computer Vision and Pattern Recognition, Vol. 1, p421-428*, 2004.

[13] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. In *Computer Vision and Pat. Recognition, Vol. 1, p605-612*, 2003.

[14] Y. Ting and W. Ying. Decentralized multiple target tracking using netted collaborative autonomous trackers. In *Computer Vision and Pattern Recognition, Vol. 1, p939-946*, 2005.

[15] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 12(1):1, 2000.

[16] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *International Joint Conference on Artificial Intelligence*, 2001.