Visually Discovering Objects' Kinematic Structures

Ian Lenz
Department of Computer Science
Cornell University
Ithaca, NY 14853
ianlenz@cs.cornell.edu

Akram Helou
Department of Computer Science
Cornell University
Ithaca, NY 14853
helou@cs.cornell.edu

1 Introduction

1.1 Problem

In order for a robot to carry out meaningful tasks in diverse and complex environments, it needs the ability to autonomously discover salient parts of its environment that are relevant for successfully carrying out its tasks. One subset of such tasks includes those that involve manipulation of everyday objects. Autonomously acquiring a task relevant model of a previously unseen object would expand the set objects that a robot can effectively (correctly and efficiently) manipulate. Such a model needs to be as simple as possible to decrease the overhead of acquiring it and to increase the likelihood of it transferring to similar objects. A hypothesis is that the kinematics model of an object is enough to allow the robot to effectively use this object. For this project, we will assume that we have some robot that can effectively manipulate a previously unseen object to obtain sufficient information from its sensors to infer a complete kinematics model of the object. Furthermore, the only sensor that we will use is a stereo camera. Thus, our goal becomes to discover the kinematics model of previously unseen objects using only vision.

1.2 Related Work

[3] [2] propose a non-probabilistic framework for discovering the kinematics model of a previously unseen planar object using very little assumptions about the object or the environment it is embedded in. The method can deal with both revolute and prismatic joints, is robust against different object shapes and textures variation as well as environment conditions. There are several limitations in this work. First, the method can only deal with planar objects. Second, it does not attempt to model the uncertainly in the physical dimensions of the links or the relative locations of the joints.

On the other hand, [1] proposes a probabilistic framework for discovering the kinematics model of arbitrary objects in motion from 2D or 3D time series of features. Their kinematics model cannot account for prismatic joints however. Furthermore, an object in motion captured from a camera should roughly remain in the same plane; this is restriction to planar objects again. However, an advantage over the previously cited work is that uncertainty is explicitly modeled.

2 Algorithm

2.1 Previous Algorithm

Our first attempt fit a mixture of Gaussians to each frame independently, then used SIFT keypoints to match them across frames. Joint locations were inferred using the intersection of the Gaussians' principal axes. This algorithm was effective at determining whether an object was jointed or rigid, but had significant error in segmenting objects into links and in determining their joint locations. Results from this algorithm can be seen in fig. 1, and a case where it clearly failed can be seen in fig. 2.

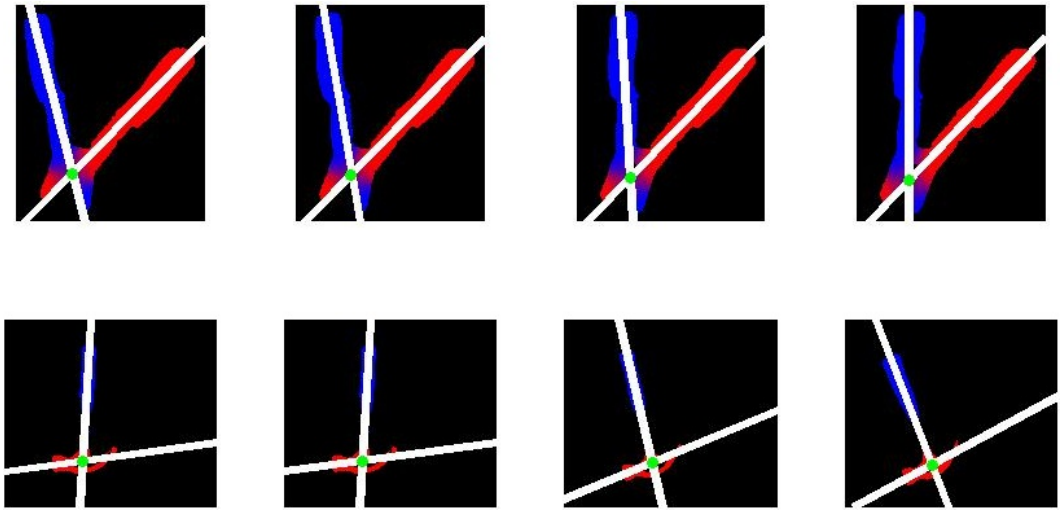

Figure 1: Results from previous algorithm. Links along with the principal axes of the corresponding Gaussians (white) and predicted joint locations (green) for series of frames of a jointed object (top) and rigid object (bottom)

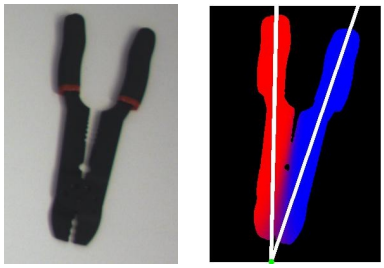

Figure 2: Object for which the previous algorithm did not perform compute link membership correctly, leading to significant error in predicting joint location

2.2 Overview of New Algorithm

Our new algorithms is derived from constructing a maximum likelihood solution. Our algorithm makes two assumptions. First, each link of an object can be represented by multiple Gaussians. Second, a link in a rigid object can only undergo a translation and rotation transformation from frame to frame. These assumptions naturally lead to defining a joint likelihood function across all frames consisting of a mixture of Gaussians for each frame with the following additional constraints: the covariance matrix and mean for each Gaussian in each frame are rotated and translated versions

The algorithm, as a whole, progresses as follows:
1. Compute SIFT matches between all pairs of distinct frames.
2. Take the base frame as the frame with the largest total number of SIFT matches with other frames.
3. Fit a mixture of K Gaussians to the base frame using E-M
4. Make an initial estimate of R_j^t and T_j^t for all frames t and Gaussians j by using the SIFT keypoints in frame t which match the SIFT keypoints for Gaussian j in the base frame.
5. Until convergence:
5.1. Update μ, Σ, and ϕ simultaneously as per the maximization above
5.2. Update R and T simultaneously as per the maximization above
6. Infer joint presence and locations

Figure 3: Overview of the algorithm

of the covariance matrix and mean of some corresponding Gaussian in some base frame. The base frame is defined as the frame with the maximum number of SIFT keypoint matchings with the remaining frames. Since it is difficult to simultaneously maximize the likelihood function for all parameters, we iteratively maximize over the covariances and means of the Gaussians, and then maximize over the rotation and translation between the base frame and the remaining framesframes while holding all other parameters fixed in both cases. Once we have converged to a satisfactory model of the object's links, we compute the joint locations of the object and determing whether it is a rotational or fixed joint. The details of the algorithm are specified in figure 3.

Overall, our new algorithm improves over our previous one in two areas: First, our previous algorithm assumed that any link can be approximated using one Gaussian. In contrast, our new algorithm assumes that any link can be fitted by multiple Gaussians which is a intuitively plausible. Second, our previous algorithm models the links in each frame independently of the other frames. Our new algorithm uses all the frames simultaneously to determine the kinematics structure of the object while adding the constraint that across frames links can only undergo translations and rotations transformations.

2.3 Likelihood Function and Paremeter Updates

Assume we have N frames and K Gaussians per frame. We also have data points $x_i^t, i \in 1, ..., n_t, t \in 1, ..., N$ (where n_t is the total number of data points in frame t). These data points correspond to foreground points in the image. The data itself is of dimension m. Finally, let A_l^u denote a variable A which belongs to the lth Gaussian in frame u.

Here are the parameters of the likelihood function:

Means of each of the K Gaussians: $\mu = (\mu_1, \mu_2, ..., \mu_K)$
covariance matrices of each of the K Gaussians: $\Sigma = (\Sigma_1, \Sigma_2, ..., \Sigma_K)$
Membership probabilities in each of the K Gaussians:

$\phi^1 = (\phi_1^1, \phi_2^1, ..., \phi_K^1)$
$\phi^2 = (\phi_1^2, \phi_2^2, ..., \phi_K^2)$
...
$\phi^N = (\phi_1^N, \phi_2^N, ..., \phi_K^N)$
Rotations between each of the K Gaussians in the base the frame and the coresponding Gaussians in the remaining frames:

$R^1 = (R_1^1, R_2^1, ..., R_K^1)$
$R^2 = (R_1^2, R_2^2, ..., R_K^2)$
...
$R^N = (R_1^N, R_2^N, ..., R_K^N)$
Translations between each of the K Gaussians in the base the frame and the coresponding Gaussians in the remaining frames:

$T^1 = (T_1^1, T_2^1, ..., T_K^1)$
$T^2 = (T_1^2, T_2^2, ..., T_K^2)$
...
$T^N = (T_1^N, T_2^N, ..., T_K^N)$

The likelihood function:

$$
\begin{aligned}
L(\mu, \Sigma, \phi, R, T) &= \prod_{t=1}^{N} \prod_{i=1}^{m_t} p(x_i^t, y_i^t; \mu, \Sigma, \phi, R, T) \\
&= \prod_{t=1}^{N} \prod_{i=1}^{m_t} p(x_i^t | y_i^t; \mu, \Sigma, \phi, R, T) p(y_i^t; \mu, \Sigma, \phi, R, T)
\end{aligned}
$$

The log-likelihood function:

$$l(\mu, \Sigma, \phi, R, T) = \sum_{t=1}^{N} \sum_{i=1}^{m_t} \log p(x_i^t | y_i^t; \mu, \Sigma, \phi, R, T) + \log p(y_i^t; \mu, \Sigma, \phi, R, T)$$

With

$$p(x_i^t | y_i^t; \mu, \Sigma, \phi, R, T) = \frac{1}{(2\pi^{m/2})|R_{y_i^t}^t)^T \Sigma_{y_i^t} R_{y_i^t}^t|^{1/2}} \ldots$$
$$\ldots \exp\left(-\frac{1}{2} [x_i^t - (\mu_{y_i^t} + T_{y_i^t}^t)]^T [(R_{y_i^t}^t)^T \Sigma_{y_i^t} R_{y_i^t}^t]^{-1} [x_i^t - (\mu_{y_i^t} + T_{y_i^t}^t)] \right)$$
$$p(y_i^t; \mu, \Sigma, \phi, R, T) = \phi_{y_i^t}^t$$

The likelihood function is maximized over μ, Σ, and ϕ by:

$$\mu_j = \frac{\sum_{t=1}^{N} \sum_{i=1}^{m_t} 1\{y_i^t = j\}(x_i^t - T_j^t)}{\sum_{t=1}^{N} \sum_{i=1}^{m_t} 1\{y_i^t = j\}}$$

$$\Sigma_j = \frac{\sum_{t=1}^{N} \sum_{i=1}^{m_t} 1\{y_i^t = j\} R_{y_i^t}^t [x_i^t - (\mu_{y_i^t} + T_{y_i^t}^t)][x_i^t - (\mu_{y_i^t} + T_{y_i^t}^t)]^T (R_{y_i^t}^t)^T}{\sum_{t=1}^{N} \sum_{i=1}^{m_t} 1\{y_i^t = j\}}$$

$$\phi_j^t = \frac{1}{m_t} \sum_{i=1}^{m_t} 1\{y_i^t = j\}$$

The likelihood function is maximized over T is done by:

$$T_j^t = \frac{\sum_{i=1}^{m_t} 1\{y_i^t = j\}(x_i^t - \mu_j)}{\sum_{i=1}^{m_t} 1\{y_i^t = j\}}$$

Maximizing over R is more difficult, and we have not been able to find a closed-form solution. This difficulty is exacerbated by the fact that R, as a rotation matrix, has a few additional constraints on its form. Currently, we maximize over R_j^t by finding the covariance of the points in frame t assigned to Gaussian j, then finding the rotation between the principal axis of Σ_j and this new covariance. While this is not a direct maximization from the equations, it has proven highly effective in practice.

2.4 Initialization

To obtain an intial estimate of Gaussian parameters, we fit Gaussians to the foreground points in the base frame i using standard E-M. Then, for each Gaussian k, we obtain a vector of SIFT keypoints P_k. Then, for each other frame j, we compute the SIFT matches from these keypoints, giving us two vectors, $P_k^{i,j}$ and $P_k^{j,i}$, corresponding to the points in frame i whcih match the points in frame j and vice-versa, ordered so that the point at each position in $P_k^{i,j}$ is a SIFT match for the point at the corresponding position in $P_k^{j,i}$. We want to initialize T_k^j and R_k^j such that:

$$P_k^{j,i} = R_k^j (P_k^{i,j} - \mu_k) + T_k^j$$

However, since we will typically have more than two keypoint matches per Gaussian, this is an overconstrained system and we must resort to least-squares.Taking N as the length of $P_k^{i,j}$, T_k^j is simply computed in this manner by:

$$T_k^j = \frac{1}{N}\sum_{n=1}^{N} P_k^{j,i(n)} - P_k^{i,j(n)}$$

R_k^j is more difficult to compute. An estimate can be made by:

$$\hat{R}_k^j = ((P_k^{i,j})^T P_k^{i,j})^{-1}(P_k^{i,j})^T P_k^{j,i}$$

However, we have no guarantee that $\hat{R}_j^k$ will follow the constraints of a rotation matrix and thus make a least-squares estimate to the closest rotation matrix by:

$$R_k^j = \hat{R}_k^j((\hat{R}_k^j)^T \hat{R}_k^j)^{-1/2}$$

This guarantees R_k^j to be orthogonal, though we must also ensure that it is a special orthogonal matrix, ie that its determinant is 1. This is simply done by multiplying each value in the first column by -1 if the determinant of R_k^j is -1.

2.5 Inferring Joint Locations

Once the algorithm has converged, we can compute the location of a joint between two links, denoted as links j and k using frames i and $i+1$ by observing the translation and rotation undergone by link k relative to link j between these frames.

First, we compute R1 and R2, which are the rotations made by link k relative to link j in frames i and $i+1$, respectively. We also compute T1 and T2, which are the translations made by link k relative to link j in the coordinate frame defined by link j's rotation, for frames i and $i+1$, respectively.

$$\begin{aligned} R1 &= (R_j^i)^T * R_k^i \\ R2 &= (R_j^{i+1})^T * R_k^{i+1} \\ T1 &= (R_j^i)^T * (T_k^i - T_j^i) \\ T2 &= (R_j^{i+1})^T * (T_k^{i+1} - T_j^{i+1}) \end{aligned}$$

Once these have been computed, we can compute the relative rotation and translation made by link k with respect to link j between frames i and $i+1$ simply, as follows:

$$\begin{aligned} R_{\text{rel}} &= R1^T * R2 \\ T_{\text{rel}} &= T2 - T1 \end{aligned}$$

We can then apply these to compute the joint location P, relative to the mean for link k, as:

$$\hat{J} = R_j^i * (R_k^i)^T * (I - R_{\text{rel}})^{-1} * T_{\text{rel}}$$

We compute $\hat{J}$ for each pair of sequential frames. We discard any values with very large L2 norms, since these correspond to motions for which links j and k appear to be rigid with respect to each other. If all values are discarded in this way, we say that links j and k are components of a rigid body.

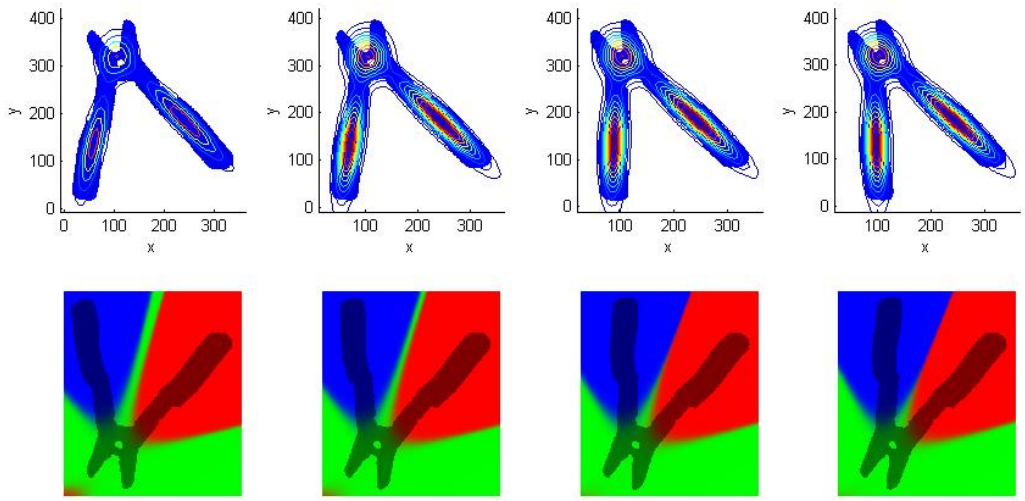

Figure 4: Mixtures of Gaussians fit to a series of images of a pliers by our new algorithm. Top row shows the distributions themselves as contours, bottom shows the assignment of points to links.

If there are some frames in which the relative motion of links j and k cannot be explained as a rigid body motion, we know that either there exists a joint between them or they are independent. If all computed joint locations are relatively close to each other (all pairwise L2 distances are within a certain threshold), we say that there exists a joint between links j and k, at a location equal to the average value of $\hat{J}$ for these frames. If there is some pair of computed joint locations whose L2 distance exceeds this threshold, we say that links j and k are independent.

Finally, to compute the actual location, J', of a joint between links j and k at relative position J in frame i:

$$J' = R_k^i * J + \mu_k + T_k^i$$

3 Results

This new algorithm has proven very effective in consistently fitting Gaussians to the same link across frames when the number of Gaussians is no more than three, as can be seen in fig. 4. On the other hand, if the number of Gaussians is greater than three, as can be seen in fig. 8, then a link might be labeled by different Gaussians from frame to frame. This is primarly due to the fact that we do not incorporate SIFT keypoints in the likelihood function therefore, there is no visual notion of link similitude . On the other hand, given accurate fitting of the links, we can consistenly infer the joint locations accurately as can be seen in fig. 6.

3.1 Discussion

It should be noted that we fit at most three Gaussians to any of the objects in fig. 5 when some would be better modeled by more. This is a limiation imposed by our use of SIFT keypoints for initialization. Since these keypoints are sparse, a small area such as the tips of a pair of pliers may not contain any keypoints, making initialization difficult. We are looking into alternative initialization methods which either remove our dependence on SIFT keypoints or augment them. The other reason disallowing us from fitting more than three Gaussians is that we don't incorporate SIFT keypoints into our likelihood function. This limits in a fundamental way because we do not have a visual feature that determines link similitude. This leads to link assignments changing as can be seen in fig. 8.

However, our algorithm is very effective in choosing correct, consistent assignments of foreground points to links for larger links and as long as we don't use more than three Gaussians. This allows it

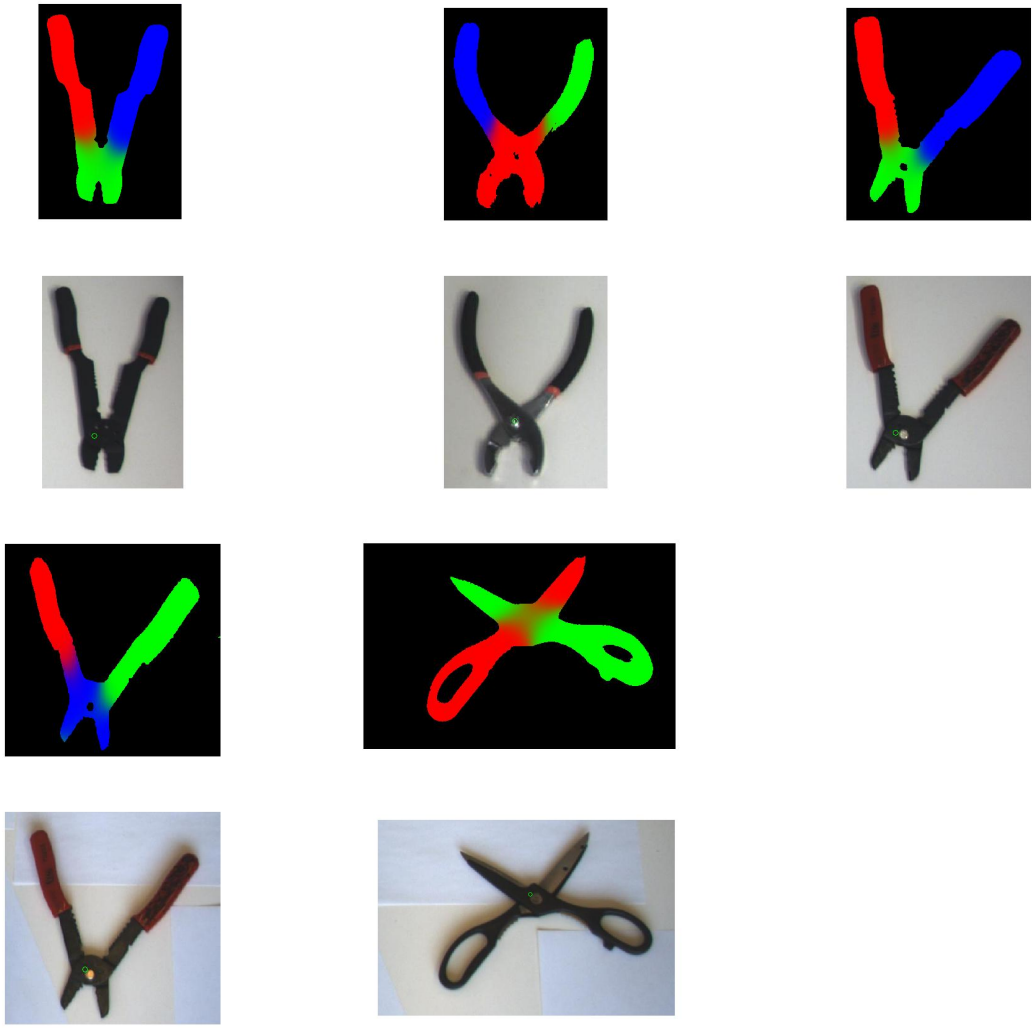

Figure 5: Discovered links (top) and joint locations (bottom) for five objects.

Object	Joint location error (pixels)
1	16.6110
2	9.3753
3	14.8097
4	12.0953
5	11.4075

Figure 6: Error in inferred joint location for objects in fig. 5

to infer joint locations with a high degree of accuracy, which would be increased for a larger dataset (the results shown used only 5-10 frames each). It is also able to correctly infer whether or not a joint exists between a pair of links, properly reporting that the "links" discovered for the object in 7 represent a single rigid body.

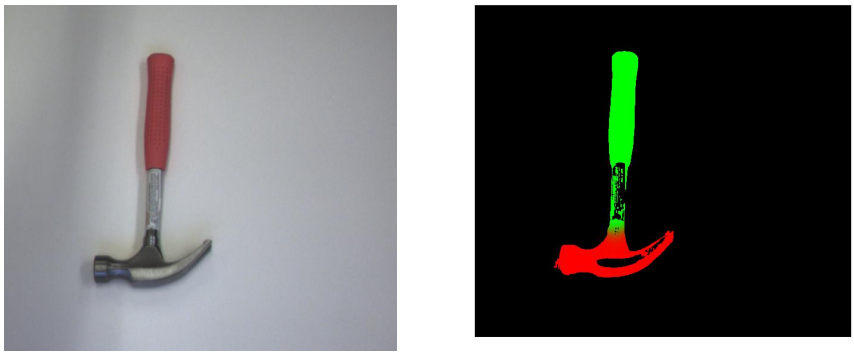

Figure 7: Object (left) and possible link structure (right) for which our algorithm correctly determines a rigid connection between the two inferred links

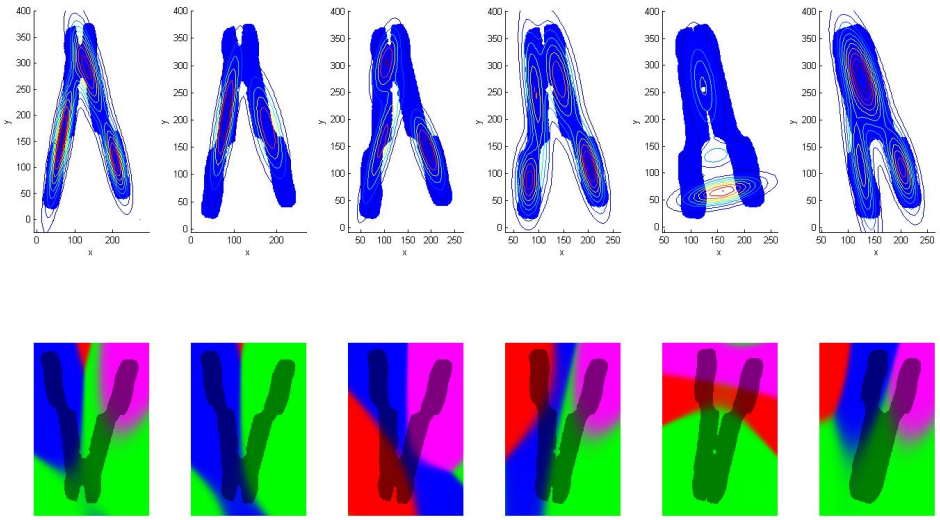

Figure 8: Mixtures of four Gaussians fit to a series of images of a pliers by our new algorithm. Top row shows the distributions themselves as contours, bottom shows the assignment of points to links.

4 Future work

In our future work, we plan on making the likelihood function richer by including SIFT keypoints matching in the likelihood function to bias the algorithm to be consistent with links labeling across frames. Furtheremore, we plan on relying on more than just SIFT keypoints for initializations. Another important milestone is to generalize our work to 3D objects. We plan to do so by incorporating point clouds into our data.

References

[1] Daniel Tarlow David Ross and Richard Zemel. Learning articulated skeletons from motion. *Workshop on Dynamical Vision at ICCV*, 2007.

[2] Yuri Pyuro Dov Katz and Oliver Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. *Proceedings of Robotics: Sci-*

ence and Systems, 2008.

[3] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.