To The Point: Autonomous Tool Tip Detection and Pose Estimation for Robotic Manipulation of Unknown Human Tools

Charles C. Kemp and Aaron Edsinger
MIT CSAIL
cckemp@csail.mit.edu, edsinger@csail.mit.edu

27th October 2005

Abstract

Robots that are able to use human tools could more easily work with people, perform tasks that are important to people, and benefit from human strategies for accomplishing these tasks. For a wide variety of tools and tasks, control of only the tool's endpoint is sufficient for its use. In this paper, we present a straight-forward method for rapidly detecting the endpoint of an unmodeled tool and estimating its position with respect to the robot's hand. This allows the robot to control the position of the tool endpoint and predict its visual location. We evaluate the approach on a humanoid robot across a variety of human tools.

1 Introduction

Consumer robots are now successfully performing specialized tasks in everyday human environments. Research is gradually leading towards general purpose robots that perform well in human environments and take advantage of the common objects found within them [7]. Robots that are able to use human tools found within these environments could more easily work with people, perform tasks that are important to people, and benefit from human strategies for accomplishing these tasks. Ideally, a robot would quickly learn how to use a new tool autonomously, since the set of human tools is large, varies widely in appearance, and continues to grow. For a wide variety of tools and tasks, control of only the tool's endpoint is sufficient for its use. These basic skills not only enable rudimentary use of the tool, but also assist further learning by allowing the robot to actively test and observe the endpoint.

In this paper, we present a straight-forward method for rapidly detecting the endpoint of an unmodeled tool and estimating its position with respect to the robot's hand. This allows the robot to control the position of the tool endpoint and predict its visual location. We show successful results for this method using

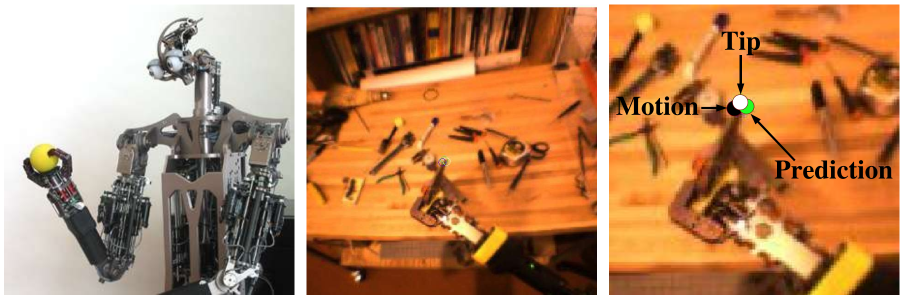

Figure 1: [Left] The humanoid robot, Domo, used in this work. [Middle] A typical view from the robot's camera of the hand holding a pair of pliers. A naturally lit, cluttered background ensures a non-trivial unstructured environment for perception. [Right] A magnified view of the hand and tool. The motion interest point (black), hand labeled tool tip (white), and the estimation results (green) are annotated.

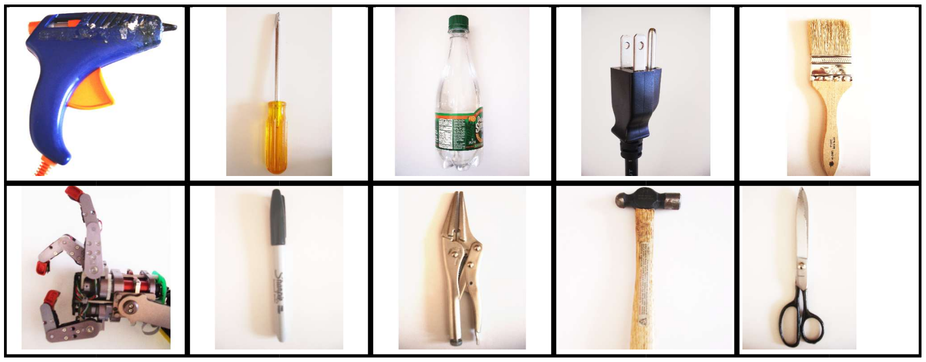

Figure 2: The approach was evaluated on a hot-glue gun, screwdriver, bottle, electrical plug, paint brush, robot finger, pen, pliers, hammer, and scissors.

the humanoid robot pictured in Figure 1 with a variety of traditional tools shown in Figure 2, including a pen, a hammer, and pliers, as well as more general tools such as a bottle and the robot's own finger.

To find the tip of the tool, the robot rotates the tool while detecting the most rapidly moving point between pairs of consecutive images. An estimation process then finds the 3D point in the hand's coordinate system that best explains these noisy detections. Given this protocol, motion serves as a powerful cue for detecting the endpoint of a wide assortment of human tools. The method makes few assumptions about the size and shape of the tool, its pose in the robot's hand, or the environment in which the tool is being manipulated. The method requires a calibrated camera and a model of the kinematic chain from the robot's camera to its hand.

We start by discussing related work in Section 2. We then describe the tool tip detection method in Section 3. Next, in Section 4, we present our experimental results with the robot and 10 different human tools. We conclude with a discussion of the approach in Section 5.

2 Related Work

Research involving robot tool use often assumes a prior model of the tool or constructs a model using complex perceptual processing. Industrial robot arms use specialized, well-modeled tools that are rigidly mounted using exchangeable end-effectors [17]. For example, Ruf [1] has demonstrated a real-time system that can visually localize the tool end of a manipulator using a polyhedral model of the tool. A recent review of robot tool use by Amant [24] fails to find significant examples of robots using human tools outside of work at NASA. NASA has explored the use of human tools by robots with the Robonaut platform[5]. They used a detailed set of tool templates combined with stereo depth information to successfully guide a standard power drill to fasten a series of lugnuts [14]. These approaches are not likely to scale to the wide variety of human tools since they depend on detailed models.

The robot hand can be considered as a specialized type of tool, and many researchers have created autonomous methods of visual hand detection through motion. Fitzpatrick and Metta [10, 19] used image differencing to detect ballistic motion and optic-flow to detect periodic motion of the robot hand. For the case of image differencing they also detected the tip of the hand by selecting the motion pixel closest to the top of the image. Natale [21] used image differencing to detect periodic hand motion with a known frequency, while Arsenio [4] used the periodic motion of tracked points. Michel et. al. used image differencing to find motion that is coincident with the robot's body motion [20]. Kemp [15] combined the motion model described in Section 3.1 with a wearable system to detect the hand of the wearer and learn a kinematic model. These methods localize the hand or arm, but do not select the endpoint of the manipulator in a robust way.

A long history of work in AI and computer vision has focused on learning tool function [29]. For example, Duric [8] looked at associating a tool's function with its prototypical motion. Robots that can actively learn about tool use has been the subject of more recent work. Bogoni [6] investigated relating the physical properties of the tool to the perceptual outcomes of its use when tested by a robot. Stoytchev [26] has explored learning a tool's function through its interactions with objects. This body of work typically assumes that a clean segmentation of the tool can be readily extracted from the image or that the tool features are known in advance.

In our work, we use our knowledge of how the robot's hand rotates while holding the tool to make 3D estimations about the location of the tool tip. This relates to methods for 3D scanning in which objects are placed on a rotating platform in front of a single camera [18]. However, these methods typically use a known or well-modeled background to cleanly segment the object, simple platform motion, and occlusion free views of the object. Instead of attempting to create a 3D reconstruction of the object for all views, we skip directly to analyzing the feature of interest under a short period of random hand motion. More generally, our 3D estimation technique relates to the well-studied area of 3D estimation from multiple views [11].

Our tool tip detection method makes use of optic flow with an affine global motion model and a per-pixel Gaussian measurement error model. The method bears many similarities to optic flow algorithms in the literature, such as global motion modeling with 2D affine models [13, 28, 27, 22, 25], and modeling measurement error from block matching [2, 23]. However, we are unaware of previous work outside of [15] that estimates the significance of observed motion in real-time using our specific method.

3 Detecting the Tool Tip

We wish to detect the end point of a tool in a general way. The detection process combines two types of information. First, the detection process looks for points that are moving rapidly when the hand is moving. This throws out points that are not controlled by the hand and highlights points under the hand's control that are far from the hand's center of rotation. Typically tool tips are the most distal component of the tool relative to the hand's center of rotation, and consequently have higher velocity. The hand is also held close to the camera, so projection increases the speed of the tool tip in the image. Second, the detection process makes use of 3D information provided by a kinematic model of the robot in order to filter out noise and combine detections from multiple 2D views of the object.

3.1 2D Tool Tip Detection Using Motion

In order to find the tool tip, we find points that are moving significantly with respect to the background. We model global image motion using a 2D affine model and then weight edges by how much their motion differs from this model.

This difference is measured using the Mahalanobis distance between each edge's Gaussian error measurement model and the motion predicted by the 2D affine motion model. We then select the edge point with the largest weight as the most likely location for the tool tip. This weight reflects both the estimated speed of the edge point and the certainty of the estimate.

The global motion model, A, can be represented as a $2x3$ affine matrix that transforms an image position $p_1 = (u_1, v_1)$ at time step 1 into an image position $p_2 = (u_2, v_2)$ at time step 2,

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$$

The model can account for global changes in translation, scale, rotation, and shearing. We use weighted linear least squares to fit the model A to a set of estimated translations, t_i, each of which has an associated covariance matrix, C_i, that represents the estimate's error. After we have our model, A, we compute the Mahalanobis distance between estimates t_i and model A in order to estimate how likely it is that each translation was generated by the model.

We use the standard technique of block matching to estimate the motion of points between consecutive images. This technique searches for point correspondences between consecutive images using image blocks as point descriptors. We compare two locations, p_1 and p_2, in the consecutive images, I_1 and I_2, by computing the sum of absolute differences, s_{12}, between the $5x5$ blocks of pixels surrounding the two locations, $s_{12} = \sum_x |I_1(x - p_1) - I_2(x - p_2)|$. Block matching is only performed at edge points detected in I_1 with a Canny edge detector in order to achieve real-time rates and reduce the use of uninformative points. We compare each of these edge points in the first image to each location within an $11x11$ search window in the second image. This results in an $11x11$ array of error values, s_j, describing the similarity between the location in the first image and the locations in the second image. We then select the best matching location, p_b, which has the lowest value, s_b, and find the covariance matrix, C, for a Gaussian model of the matching error around this best match using

$$C = \alpha I + \frac{1}{\sum_j w_j} \sum_j w_j (p_j - p_b)(p_j - p_b)^T \text{ where } w_j = \begin{matrix} 1 \text{ if } s_j < s_b + \tau \\ 0 \text{ if } s_j \geq s_b + \tau \end{matrix}$$

which thresholds the errors s_j with $s_b + \tau$ to create a binary error map w_j from which this Gaussian error model is computed [1]. To avoid over-confidence in the estimated error distribution, we also add αI to C, which is equivalent

[1]For the blocks of size 5x5 that we use we set $\tau = 200$ which assumes that we should see no more than 4 units of additional error per pixel in a block that truly matches, since 5x5x4=200 . We set τ to this constant value by hand. It works sufficiently well for our purposes, but estimating τ from measured pixel errors, especially as a function of brightness and contrast, might improve the algorithm's performance.

to convolving the error Gaussian with a circular Gaussian with variance α. We use $\alpha = \frac{1}{4}$.

We use weighted linear least squares to incorporate the error covariance matrices, C_i, generated by the block matching process into the estimation of the motion model A. We solve $a = (X_1^T \Sigma^{-1} X_1)^{-1} X_1^T \Sigma^{-1} x_2$, which minimizes $(X_1 a - x_2)^T \Sigma^{-1} (X_1 a - x_2)$ where we define the terms as follows:

$$X_1 = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n & v_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_n & v_n & 1 \end{bmatrix}, \quad a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}, \quad x_2 = \begin{bmatrix} u_1 + t_{u1} \\ v_1 + t_{v1} \\ u_2 + t_{u2} \\ v_2 + t_{v2} \\ \vdots \\ u_n + t_{un} \\ v_n + t_{vn} \end{bmatrix}$$

a is the vectorized form of the matrix A, (u_i, v_i) is the location of an edge point i in image I_1, and (t_{ui}, t_{vi}) is the lowest error translation of edge point i into image I_2. We now define Σ to be a sparse block diagonal matrix with $2x2$ matrices C_i along the diagonal, where C_i is the covariance matrix describing the match error for edge point i.

$$\Sigma^{-1} = \begin{bmatrix} C_1^{-1} & 0 & 0 & \cdots & 0 \\ 0 & C_2^{-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & C_n^{-1} \end{bmatrix}$$

Due to the sparse block form of the matrices, these equations can be significantly simplified and solved in real-time as shown in detail in Appendix A. We can consider this weighted linear least squares solution to be the maximum likelihood estimation of our model a where the error is Gaussian distributed, $\mathcal{N}$, with covariance matrix Σ.

$$\mathcal{N}(\Sigma, X_1 a)(x_2) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(X_1 a - x_2)^T \Sigma^{-1} (X_1 a - x_2)}$$

The fitting process is iterated in order to remove the influence of edge points that are not likely to be part of the motion background. On each iteration we remove the worst fitting edge points and reestimate a, which is computationally reasonable since we only need to perform block matching once. We determine how well each edge point matches the model by calculating the Mahalanobis distance, d_i, between the best match translation vector, t_i, for edge point i and the translation predicted by the model, Av_i, relative to the edge point's error model defined by the covariance matrix C_i.

$$d_i = \left(\left(Av_i - \begin{bmatrix} u_i + t_{ui} \\ v_i + t_{vi} \end{bmatrix} \right)^T C_i^{-1} \left(Av_i - \begin{bmatrix} u_i + t_{ui} \\ v_i + t_{vi} \end{bmatrix} \right) \right)^{\frac{1}{2}}$$

The units of Mahalanobis distance, d_i, are image pixels, so working with these distances is intuitive. The Mahalanobis distances also ranks the edge points, which allows us to throw out the points that fit the model poorly.

The motion processing gives us a weighted edge map, where edge points with larger weights are deemed less likely to be moving with the background. We select the edge point with the largest weight as the most likely location for the tool tip. This weight reflects both the estimated speed of the edge point and the certainty of the estimate, so the tendency of the tool tip to be a corner may also help with detection since the estimated motion by block matching will tend to be more certain with corners.

3.2 Using the Kinematic Model to Interpret the Detected Points

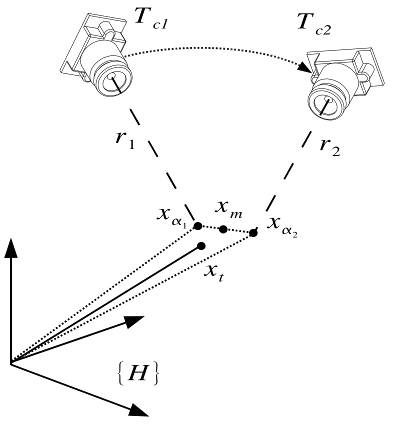

Figure 3: 3D estimation of the tool tip, x_t given two observations. Within the hand's coordinate system, $\{H\}$, the hand remains stationary and the camera moves around the hand. The kinematic model is known such that $T_c^{-1}x_t$ is the tool tip in the camera frame for robot configuration c. At each observation, the rays r_1 and r_2, defined in frame $\{H\}$, pass through the tip detection pixel and the optical center of the camera. The points x_{α_1} and x_{α_2} on these rays defines the minimal distance between them. The estimate of intersection between these rays, x_m, is the midpoint between x_{α_1} and x_{α_2}.

In the previous subsection we presented a method for detecting motion feature points that are likely to correspond with the tip of the tool in the robot's hand. After detecting these 2D motion feature points from a series of frames with distinct views, we use the robot's kinematic model to combine these 2D feature points into a single 3D estimate of the tool tip's position in the hand's coordinate system. This is illustrated in Figure 3.

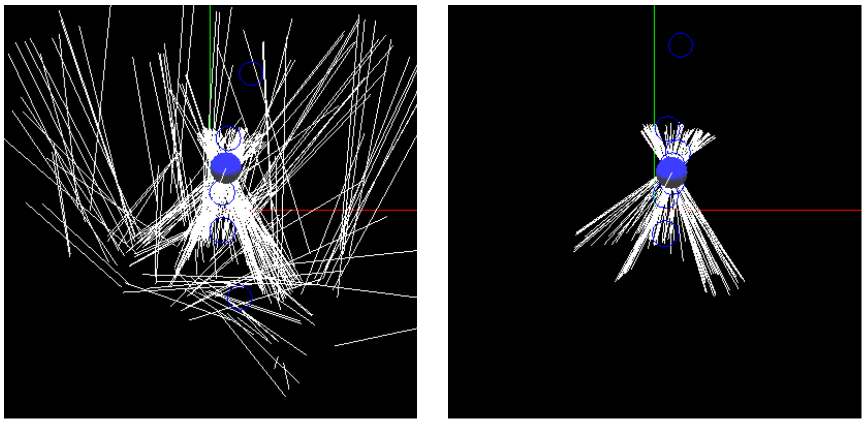

Figure 4: Estimation results for the pliers. The sphere is at the point of the largest cluster and corresponds to the estimated tool tip position in the robot's hand frame of reference. Each ray indicates a data sample and emanates from the camera focal point through the tip estimation pixel. [Left] Clustering results on the training data using motion based tip estimation. [Right] Clustering results on the test data using hand labeled tip estimation. While noise in the motion estimation process is apparent, the approach finds a tool pose close to the hand labeled solution.

Each feature detection specifies a pixel on an image. Given the kinematic model, a pin-hole camera model for the calibrated camera, and the robot's configuration when the image was captured, each of these pixels can be converted into a ray with a known relationship to the robot's hand in world coordinates. Each ray starts at the corresponding pixel on the image plane of the pin-hole camera model and points into the world through the optical center of the pin-hole camera[2]. We can then use the kinematic model to transform this ray into the coordinate system of the robot's hand. Within the hand's coordinate system, the hand remains stationary and the camera moves around the hand acquiring distinct views. Since the tool is rigidly grasped by the hand, the tool and its tip remain stationary with respect to the hand's coordinate system.

As with traditional multi-view estimation, pixels that correspond with the same 3D point will have associated rays that intersect one another at that 3D point. If the detection of the tool's endpoint, the camera model, and the kinematic model were all perfect, only two views of the endpoint (that produce non-parallel rays) would be needed in order to know its 3D location. The detection method, the camera model, and the kinematic model are all imperfect, so we estimate the relevant 3D location from many samples.

A variety of approaches would be appropriate for this estimation, since only

[2]Technically, the ray should start after the outer lens of the camera, since the detected point could not be closer than that.

three parameters need to be estimated and we have plenty of data from a moderately noisy source. In this work we perform K-means clustering on a set of 3D points in the hand's coordinate system, where each 3D point corresponds to two rays that intersect or nearly intersect with one another, see Figure 4. This method is computationally efficient, easy to visualize, and produces satisfactory results.

For each image we have an associated kinematic configuration, c, and a transform, T_c, that transforms a point in the camera's coordinate system to a point in the hand's coordinate system. Using a pin-hole camera model with optical center (u_p, v_p) and focal length, f, we can transform a pixel location of a detection, (u, v), with an arbitrary depth, z, into the camera's coordinate system. We combine these two transformations to find a point in the hand's coordinate system on the ray, $r_{u,v}$, associated with pixel (u, v).

$$r_{u,v}(z) = T_c \begin{bmatrix} \frac{(u-u_p)z}{f} \\ \frac{(v-v_p)z}{f} \\ z \end{bmatrix}$$

Given this function of z, we can define the ray with two points $s = r_{u,v}(z_{start})$ and $g = r_{u,v}(z_{>start})$, where z_{start} is the depth of the start of the ray, which corresponds with the outer surface of the lens of the camera.

Given two rays, r_1 and r_2, we can find the points on the rays that are closest to one another using the following equation in the standard linear least squares form $Ax = b$,

$$\begin{bmatrix} g_1 - s_1 & g_2 - s_2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = s_2 - s_1$$

where the set of points on the ray are defined by $x_\alpha = (g-s)\alpha + s$ such that $x_\alpha \cdot (g-s) > 0$. If the points associated with α_1 and α_2 are valid points on the rays, then $x_m = \frac{x_{\alpha_2} + x_{\alpha_1}}{2}$ will minimize the squared distance between the two rays. We then use x_m as their most likely point of intersection. We collect all of the hypothesized intersections, x_m, from pairs of rays that come within 2.54 cm of one another. This computation is $O(n^2)$, where n is the number of detections. We further filter this set of hypothesized intersections by throwing out points that are greater than 3 meters from the center of the hand, and points that are inside the robot's hand or wrist, which we model with two small spheres. After collecting and filtering this set of probable intersections, we perform K-means in order to find 3D positions that are close to high density areas of probable intersections, and hence have a relatively high likelihood of being the tool tip. We then select the mean that has the most members as the most likely tool tip location.

There are many sources of error that we ignore in our model including error sensitivity as a function of distance from the camera due to projection, uncertainty about the hand's rotation which will have a larger impact on long objects, and the higher likelihood of intersections at points that are close to the camera.

In appendix B we describe an alternative approach for this estimation using a generative probabilistic model. Many elements of the estimation we use can be interpreted probabilistically within this model. For example, x_m is the maximum likelihood location for the intersection of two rays, and we filter points for which $p(x_t)$ has zero probability.

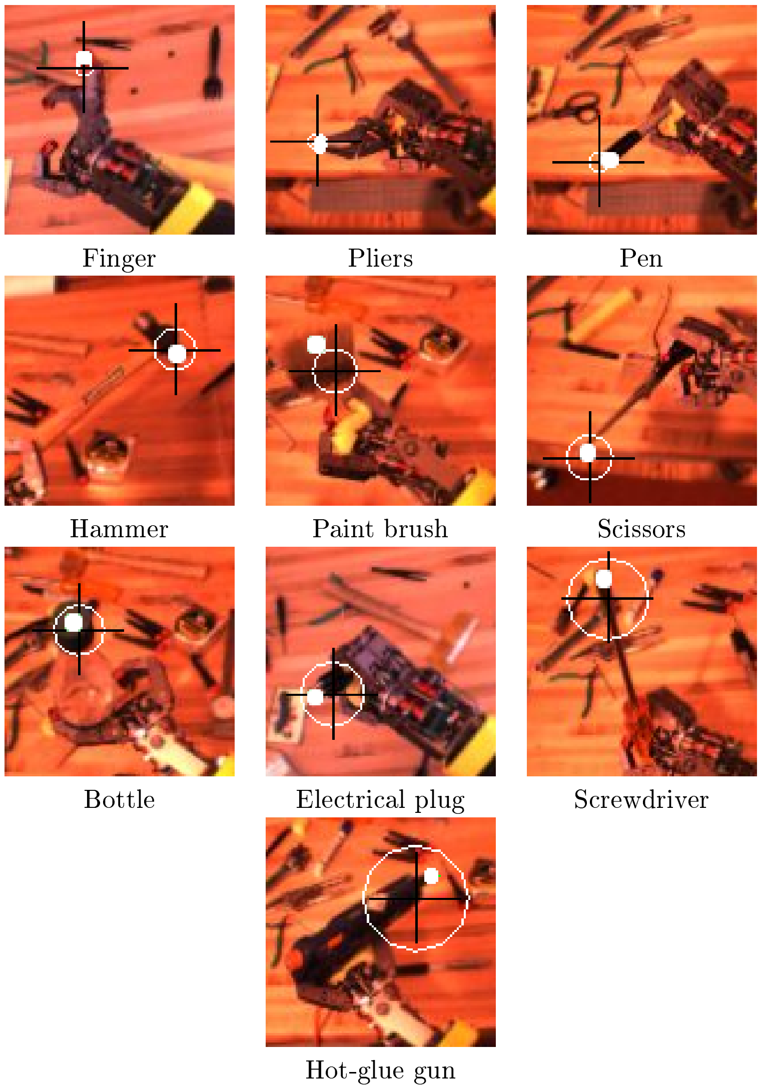

Figure 5: An example of the tip prediction for each tool. The cross is centered at the prediction point and measures 40 pixels across for scale. The radius of the white circle indicates the tool's mean pixel error. The white dot indicates the hand labeled tool tip. The error tends to scale with the size of the tool tip as expected.

4 Experimental Results

Validation of the described approach was conducted on a 29 DOF humanoid robot at the MIT CSAIL Humanoid Robotics Group [9]. The robot, named Domo, has 4 DOF in each arm, 2 DOF in each wrist, 4 DOF in each hand, and 9 DOF in the active vision head. Domo is equipped with compliant and force sensing actuators throughout most of the body. It has two Point Grey Firewire cameras with differing focal lengths (2.8mm and 3.6mm) to allow both a wide field-of-view and higher resolution for inspection.

4.1 Setup

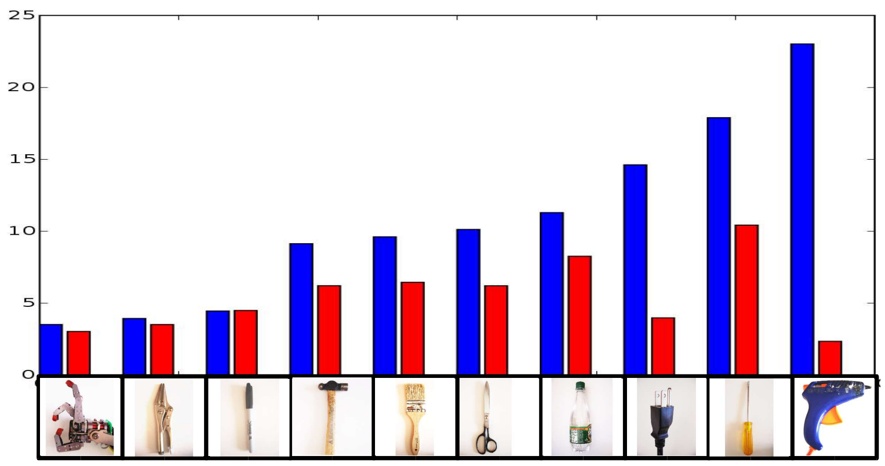

Figure 6: The mean prediction error, in pixels, for each tool. The 3D tool pose in the hand is estimated using two data sets: detector tool tips and the hand labelled tool tips. The 3D pose for both estimates is then projected onto the image plane at each sample in the test set and compared to the hand labelled location. The left (blue) bar indicates the detector error and the right (red) bar indicates a baseline error. The baseline errors are an indication of inaccuracies in the kinematic and camera calibrations.

Experiments were conducted on a variety of tools with differing lengths and endpoints. The set of tools are listed in Figure 6.

For each experiment, the 11 DOF kinematic chain from the camera up to the robot wrist was servoed to maintain a fixed pose that ensured tool visibility. The wide FOV lens, which can image a larger range of tool sizes, was used. The tool was placed in the robot hand at an arbitrary orientation and the tool pose measured manually for reference. The 2 DOF (pitch,roll) of the wrist were ramped smoothly to random positions in the range of ±60 degrees for a short duration.

Samples of the robot joint angles and the camera image were collected at $30hz$. Approximately 500 samples (15 seconds of motion) were captured for each tool. The most significant motion point was then computed for each image. The data was randomly distributed into a training set of 400 samples and a test set of 100 samples. Finally, the tool endpoint was hand labeled for each frame of the test set.

4.2 Visual Endpoint Detection

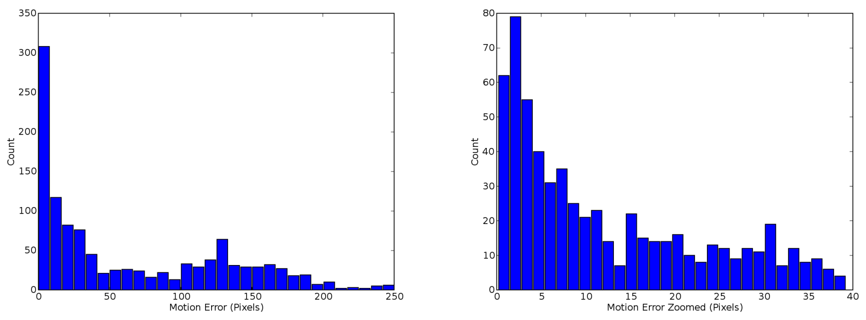

Figure 7: [Left] Error histogram, in pixels, for visual detection of the tool-tip across all tools. [Right] Detailed view of left graph. For each tool, the robot randomly rotated its wrist for 15 seconds. The detector point was located in each image. A random set of 100 samples was selected and the actual pixel location was hand labeled for comparison.

Visual detection of the tool endpoint was computed with the motion model from Section 3.1 . The model incorporates both the magnitude of pixel motion and its uncertainty. This helped to reduce erroneous detection points. In our experiments, the localization was computed offline for each pair of sequential images, though real-time rates are achievable. As shown in Figure 1, a naturally lit, cluttered background was used to ensure a non-trivial unstructured environment for perception. The motion based detection method is noisy, but as shown in Figure 7, the detections tended to match the hand labeled tool tip locations. In the experiments we present, the camera and environment were nearly stationary and the affine model of background motion usually was estimated as identity. However, the model can be used in situations with a non-stationary camera and other causes of global affine motion.

4.3 Tool Pose Estimation

The pose estimation accuracy was evaluated by first estimating the 3D tool pose in the hand on the training data set as described in Section 3.2, with $k = 8$. Figure 4 visualizes the estimation cluster for the pliers. The 3D pose was projected onto the image plane for each sample in the test set. The predicted appearance of the tool tip was then compared to the hand labelled location to compute the mean pixel error. A baseline comparison can be made by performing the estimation process on the hand labeled data set. The resulting error indicates inaccuracies in the kinematic calibration and camera model. The algorithm performs favorably with respect to this baseline error. Figure 6 illustrates the mean prediction error, in pixels, across the set of tools. Figure 5 illustrates the typical tip predictions for each tool.

4.4 Discussion of Results

As Figure 5 illustrates, the prediction performed well considering the perceptually difficult environment that the experiments use. The wide FOV camera from which the images were captured allows a larger variety of tool sizes to be explored, but the resolution of the tip was often low, on the order of 10 pixels. Errors can originate from the kinematic and camera model, as the baseline errors in Figure 6 demonstrate. On the Domo robot, the transform T_c was computed from a hand tuned model. The electro-mechanical details of the Domo robot make precise calibration difficult. An autonomous method for the hand-eye calibration problem[12, 3] could potentially reduce this error component. We analyzed the estimated 3D pose of each tool by its prediction in the image. The accuracy of these predictions indicate that the 3D estimate could be incorporated into a Cartesian space controller such that the tool becomes a natural extension of the robot's body. We trained each estimator on a data set of 400 samples which may be conservatively high given the effectiveness of the motion based detector and the ideal requirement of only two distinct views. An online, iterative extension of the estimation process could be implemented with little modification in order to provide real-time pose estimation. It is important that the wrist sample a large space of poses. In the extreme case of hand rotation occurring only in the image plane, the depth of the tool pose would be indeterminate. It is also important that the rotational velocity be limited to reduce the possibility of significant motion blur at the tool tip.

5 Discussion

We have presented a straight forward approach to detecting the end point of a tool in a robot hand and estimating it's 3D position. The strength of the approach is that it assumes little prior knowledge about the tool or its pose in the hand and avoids complex perceptual processing. Rather than segmenting the tool, estimating the 3D shape of the tool, or otherwise representing the details of the tool prior to detecting the tip, this method jumps directly to detecting the tip of the tool. The success of the method relies on two main observations. First, the natural utility of a large variety of tools is focused at the tool's endpoint, so detecting this endpoint may be sufficient for control or at least a good first step when learning about a tool. Second, for many of these tools the endpoint can be detected by its rapid motion in the image when the robot moves its hand while holding the tool.

This method requires that the robot already have the tool firmly in its grasp. One might question whether or not this would be useful in practice, as grasping may have already required perception of the object. However, this does not obviate the usefulness of the method. First, for many applications the tool may be placed within the hand of the robot by other means, such as by an instructor who is teaching the robot about the tool. Second, even if the robot has in some way perceived the tool in order to grasp it, the details of the tool's pose in the hand after grasping may be uncertain. Third, many plausible grasping methods may be successful without identifying the endpoint of tool, such as grasping by a highly compliant hand.

Similarly, one might be concerned about the extent to which the specific grasp on the tool is important for our end-point detection method. We do not have data to directly address this concern, but we believe that our method will work with most natural and secure grasps on human tools. For example, approximately stick shaped tools will support many natural and secure grasps, but many of these grasps will result in the tool tip being the farthest point from the hand's center of rotation. In the event that the end point is itself grasped, then the tip of the handle may be detected. If this happens, the system might recognize that the detected tip does not meet some desired properties, and then try another grasp.

Other modalities could be beneficially integrated with this method, such as stereo vision, additional visual features, and tracking points over multiple frames. Motion does, however, have some especially beneficial properties for this type of detection that make its use well-justified. First, motion allows us to find elements of the world that are likely to be controlled by moving the robot's hand. Stereo analysis of a static scene could be used to select elements of the scene that are close to the hand, but without motion it would not be able to detect which points are under the hand's control. Others have used this technique to detect the robot's body and objects with which it makes contact[20, 19]. Second, by moving the hand and tool we are able to observe them from several distinct views. We might instead attempt to directly find the point that is farthest from the hand's center of rotation using a monocular camera or stereo rig, but they would be sensitive to the particular pose of the hand, which might occlude the tool tip or make estimation noisy. Performing the estimation over multiple views by moving the hand increases overall robustness. Finally, although not used in this paper, the motion of the hand could be used to visually estimate the center of the hand's rotation and better select points controlled by the hand. Motion could allow us to limit our reliance on the kinematic model by visually estimating the hand's center of rotation, if not the 3D rotation of the hand. Likewise, with motion we could use our understanding of the hand's rotation to filter out elements of the scene that are moving in ways that do not correspond with the hand's rotation.

The tool tip detection is robust to static elements of the environment that are cluttered and unstructured, as long as there is some contrast between the tool and the background in the image. Additionally, motion in the environment that is far from the camera will be discounted by the projection of the tool's motion, since the tool is held close to the camera. Although our tests used a stationary camera, the motion processing algorithm does compensate for global affine motion in the image, which reduces the effect of camera motion.

For the results we present, the robot's hand is roughly human in size and shape, which is well-matched to human tools. This detection method might not perform as well with robot end-effectors that differ significantly from a human hand, since, for example, their shape might be large with respect to the tool.

The technique affords many avenues for further exploration. A reliable prediction of the tool tip in the visual scene could allow us to construct a model of the tool's visual features. This, in turn, could be used to more precisely detect and control the tool. A large literature exists for visually servoing of a robot hand to an object [16], and these approaches can be naturally extended to include control of the tool. In addition, the estimate of the tool's 3D pose in the hand can also be applied to a traditional Cartesian space control scheme.

Localization of the tool endpoint during manipulation can also be applied to learning about the functional relationship betwen the endpoint and objects. The approach described in this paper can allow the robot to actively test and observe the endpoint during interactions with the world. It is a first step towards robots that can autonomously learn about and use novel, unmodeled tools in natural environments.

References

[1] R Horaud HH Nagel A Ruf, M Tonko. Visual tracking of an end-effector by adaptive kinematic prediction. In *Intelligent Robots and Systems IROS'97*, 1997.

[2] Padmanashan Anandan. *Measuring Visual Motion From Image Sequences.* PhD thesis, University of Massachusetts, Amherst, Massachusetts, 1987.

[3] Nicolas Andreff, Radu Horaud, and Bernard Espiau. Robot hand-eye calibration using structure from motion. *International Journal of Robotics Research*, 20(3):228–248, Mar 2001.

[4] A. Arsenio and P. Fitzpatrick. Exploiting cross-modal rhythm for robot perception of objects. In *Proceedings of the Second International Conference on Computational Intelligence, Robotics, and Autonomous Systems*, December 2003.

[5] W. Bluethmann, R. Ambrose, A. Fagg, M. Rosenstein, R. Platt, R. Grupen, C. Brezeal, A. Brooks, A. Lockerd, R. Peters, O. Jenkins, M. Mataric,

and M. Bugajska. Building an Autonomous Humanoid Tool User. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robots*, Santa Monica, Los Angeles, CA, USA., 2004. IEEE Press.

[6] Luca Bogoni. Functional features for chopping extracted from observations and interactions. *Image and Vision Computing*, 16:765–783, 1998.

[7] Rodney Brooks, Lijin Aryananda, Aaron Edsinger, Paul Fitzpatrick, Charles Kemp, Una-May O'Reilly, Eduardo Torres-Jara, Paulina Varshavskaya, and Jeff Weber. Sensing and manipulating built-for-human environments. *International Journal of Humanoid Robotics*, 2004.

[8] Zoran Duric, Jeffrey A. Fayman, and Ehud Rivlin. Function from motion. *PAMI*, 18(6):579–591, June 1996.

[9] Aaron Edsinger-Gonzales and Jeff Weber. Domo: A Force Sensing Humanoid Robot for Manipulation Research. In *Proceedings of the 2004 IEEE International Conference on Humanoid Robots*, Santa Monica, Los Angeles, CA, USA., 2004. IEEE Press.

[10] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning About Objects Through Action: Initial Steps Towards Artificial Cognition. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, May 2003.

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

[12] John Hollerbach and Charles Wampler. The calibration index and taxonomy for robot kinematic calibration methods. *International Journal of Robotics Research*, 15(6):573–591, 1996.

[13] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In *Proceedings of ICPR*, pages 743–746, Jerusalem, Israel, 1994.

[14] E. Huber and K. Baker. Using a hybrid of silhouette and range templates for real-time pose estimation. In *Proceedings of ICRA 2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1652–1657, 2004.

[15] Charles C. Kemp. *A Wearable System that Learns a Kinematic Model and Finds Structure in Everyday Manipulation by using Absolute Orientation Sensors and a Camera*. PhD thesis, Massachusetts Institute of Technology, May 2005.

[16] D. Kragic and H. I. Chrisensen. Survey on visual servoing for manipulation. Technical report, Computational Vision and Active Perception Laboratory, 2002.

[17] Thomas Kurfess, editor. *Robotics and Automation Handbook.* CRC Press, Boca Raton, Florida, 2005.

[18] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul A. Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. In *Proceedings of SIGGRAPH 2002*, pages 427–437, 2002.

[19] Giorgio Metta and Paul Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, June 2003.

[20] Michel, Gold, and Scassellati. Motion-based robotic self-recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[21] Lorenzo Natale. *Linking Action to Perception in a Humanoid Robot: A Developmental Approach to Grasping.* PhD thesis, LIRA-Lab, DIST, University of Genoa, 2004.

[22] J.M. Odobez and P. Bouthemy. Robust multiresolution estimation of parametric motion models. *International Journal of Vision Communication and Image Representation*, 6(4):348–365, 1995.

[23] A. Singh and P. Allen. Image-flow computation: an estimation-theoretic framework and a unified perspective. *CVGIP: Image Understanding*, 56:152–177, 1992.

[24] R St. Amant and A.b Wood. Tool use for autonomous agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 184–189, 2005.

[25] C. Stiller and J. Konrad. Estimating motion in image sequences, a tutorial on modeling and computation of 2d motion. *IEEE Signal Process. Mag., vol. 16, pp. 70–91*, 1999.

[26] Alexandar Stoytchev. Behavior-grounded representation of tool affordances. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[27] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. Technical report, M.I.T. Media Laboratory, 1994.

[28] T. Wiegand, E. Steinbach, and B. Girod. Affine multipicture motion-compensated prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):197–209, Feb 2005.

[29] Patrick H. Winston, Boris Katz, Thomas O. Binford, and Michael R. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *National Conference on Artificial Intelligence*, pages 433–439, 1983.

A Weighted Linear Least Squares Simplification

For convenience, in this appendix we provide the results of the simplifying the weighted linear least squares formulation we use for motion processing. As explained within the main article, we wish to solve $a = (X_1^T \Sigma^{-1} X_1)^{-1} X_1^T \Sigma^{-1} x_2$, which minimizes $(X_1 a - x_2)^T \Sigma^{-1} (X_1 a - x_2)$. The computation of $X_1^T \Sigma^{-1} X_1$ only requires that we sum n $6x6$ symmetric matrices, where n is the number of edge points i used in the estimation. The 21 distinct terms have the simple and redundant form shown below, where each . represents the corresponding lower diagonal value and $C_i^{-1} = \begin{bmatrix} c_{1i} & c_{2i} \\ c_{3i} & c_{4i} \end{bmatrix}$ with $c_{2i} = c_{3i}$ due to symmetry.

$$X_1^T \Sigma^{-1} X_1 = \sum_i \begin{bmatrix} u_i^2 c_{1i} & . & . & . & . & . \\ v_i u_i c_{1i} & v_i^2 c_{1i} & . & . & . & . \\ u_i c_{1i} & v_i c_{1i} & c_{1i} & . & . & . \\ u_i^2 c_{3i} & u_i v_i c_{3i} & u_i c_{3i} & u_i^2 c_{4i} & . & . \\ v_i u_i c_{3i} & v_i^2 c_{3i} & v_i c_{3i} & v_i u_i c_{4i} & v_i^2 c_{4i} & . \\ u_i c_{3i} & v_i c_{3i} & c_{3i} & u_i c_{4i} & v_i c_{4i} & c_{4i} \end{bmatrix}$$

The structure of the matrix is more clear when written in block form with $p_i^T = \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}$

$$X_1^T U X_1 = \sum_i \begin{bmatrix} c_{1i} p_i p_i^T & c_{3i} p_i p_i^T \\ c_{3i} p_i p_i^T & c_{4i} p_i p_i^T \end{bmatrix}$$

The resulting symmetric $6x6$ matrix, $X_1^T \Sigma^{-1} X_1$, will be positive definite except for extreme circumstances that can be detected by the matrix inversion code, such as when not enough edges are provided due to darkness. We compute $(X_1^T \Sigma^{-1} X_1)^{-1}$ using fast $6x6$ matrix inversion code specialized for symmetric and positive definite matrices.

$X_1^T \Sigma^{-1} x_2$, is also computationally simple with

$$X_1^T \Sigma^{-1} x_2 = \sum_i \begin{bmatrix} (u_i + t_{xi}) u_i c_{1i} + (v_i + t_{yi}) u_i c_{3i} \\ (u_i + t_{xi}) v_i c_{1i} + (v_i + t_{yi}) v_i c_{3i} \\ (u_i + t_{xi}) c_{1i} + (v_i + t_{yi}) c_{3i} \\ (u_i + t_{xi}) u_i c_{3i} + (v_i + t_{yi}) u_i c_{4i} \\ (u_i + t_{xi}) v_i c_{3i} + (v_i + t_{yi}) v_i c_{4i} \\ (u_i + t_{xi}) c_{3i} + (v_i + t_{yi}) c_{4i} \end{bmatrix}$$

which can also be more clearly written in block form with v_i

$$X_1^T \Sigma^{-1} x_2 = \sum_i \begin{bmatrix} p_i c_{1i} & p_i c_{3i} \\ p_i c_{3i} & p_i c_{4i} \end{bmatrix} \begin{bmatrix} u_i + t_{xi} \\ v_i + t_{yi} \end{bmatrix}$$

Now, we only need to multiply these two parts to find our solution for a and the corresponding motion model matrix A.

B A Generative Probabilistic Model for Estimation of the 3D Tool Tip Location

We can interpret the problem of estimating the 3D tool tip position in terms of a probability distribution over possible 3D tool tip locations in the hand's coordinate system, $p(x_t)$. Given a single detection, we would assume that the maximum likelihood solution would be a ray of locations. Likewise, given two detections with non-parallel, yet intersecting rays, we would expect the maximum likelihood solution to be at their intersection point.

We can define a simple generative model by using an orthographic camera and assuming that many of the kinematic errors in the system can be modeled as a spherical 3D Gaussian centered around the tool tip's location in the hand's coordinate system. Due to the orthographic camera, the projection of this Gaussian distribution gives a Gaussian distribution on the image plane. The motion detection error can then be modeled as a 2D circular Gaussian convolved with this projected Gaussian distribution, which results in a 2D circular Gaussian on the image. So, if we ignore the effects of projection, a 2D circular Gaussian distribution on the image centered around the location of the tool tip in the image is a reasonable model for many sources of error.

With this justification we model the conditional probability of a detection at a location, d, in the image using a 2D circular Gaussian centered on the projected location of the tool tip. We also mix this Gaussian with a distribution, b, that models false detections across the image that are independent of the location of the tool tip. In summary,

$$p(d|x_t, c) = (1-u)N(P_c x_t, \sigma^2 I)(d) + ub(d)$$

where $\mathcal{N}$ is a 2D normal distribution, with a diagonal covariance matrix $\sigma^2 I$, and a mean of $P_c x_t$. $P_c x_t$ is the 3D tool tip location in the hand's coordinate system, x_t, projected onto the image given the robot's configuration c. We model a series of detections $d_1 \ldots d_n$ with corresponding configurations of the robot, $c_1 \ldots c_n$, as being independently drawn from this distribution, so that

$$p(d_1 \ldots d_n | x_t, c_1 \ldots c_n) = \prod_i p(d_i | x_t, c_i)$$

Using bayes rule we have

$$p(x_t | d_1 \ldots d_n, c_1 \ldots c_n) = \frac{p(d_1 \ldots d_n | x_t, c_1 \ldots c_n) p(x_t, c_1 \ldots c_n)}{p(d_1 \ldots d_n, c_1 \ldots c_n)}$$

We are only looking for relative maxima, so we can maximize

$$p(d_1 \ldots d_n | x_t, c_1 \ldots c_n) p(x_t, c_1 \ldots c_n)$$

We assume that $p(x_t, c_1 \ldots c_n) = p(x_t) p(c_1 \ldots c_n)$ so that the tool tip position in the hand's coordinate system is independent of the configurations of the system at which the images were captured. Since $c_1 \ldots c_n$are known and constant for the data set, we can drop their distribution from the maximization to end up with

$$\begin{aligned}\widehat{x_t} &= \mathrm{Argmax}_{x_t}\left(p(d_1 \ldots d_n | x_t, c_1 \ldots c_n) p(x_t)\right) \\ &= \mathrm{Argmax}_{x_t}\left(\log\left(p(x_t)\right) + \sum_i \log\left(p(d_i | x_t, c_i)\right)\right)\end{aligned}$$

A variety of methods could be used to find $\widehat{x_t}$ given this expression, including gradient ascent and brute force sampling. Brute force sampling, and many forms of gradient ascent, would require a comparison of each candidate position with the set of detections. The search only needs to proceed over three dimensions, so brute force evaluation is tractable, but it still leads to a $O(v^3 n)$ computation where v is the number of discretizations per dimension and n is the number of detections.