Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2005-083
AIM-2005-37

December 21, 2005

Visual Tool Tip Detection and Position Estimation for Robotic Manipulation of Unknown Human Tools

Charles C. Kemp, Aaron Edsinger

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Visual Tool Tip Detection and Position Estimation for Robotic Manipulation of Unknown Human Tools

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

20th December 2005

Abstract

Robots that 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 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. The robot rotates the tool while using optical flow to detect the most rapidly moving image points, and then finds the 3D position with respect to its hand that best explains these noisy 2D detections. The resulting 3D position estimate allows the robot to control the position of the tool endpoint and predict its visual location. We show successful results for this method using a humanoid robot with a variety of traditional tools, including a pen, a hammer, and pliers, as well as more general tools such as a bottle and the robot's own finger.[1]

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, 19]. Robots that use the 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 autonomously learn how to use a new tool, since the set of human tools is large, varies widely in appearance, and continues to grow.

[1] This work was sponsored by the NASA Systems Mission Directorate, Technical Development Program under contract 012461-001.

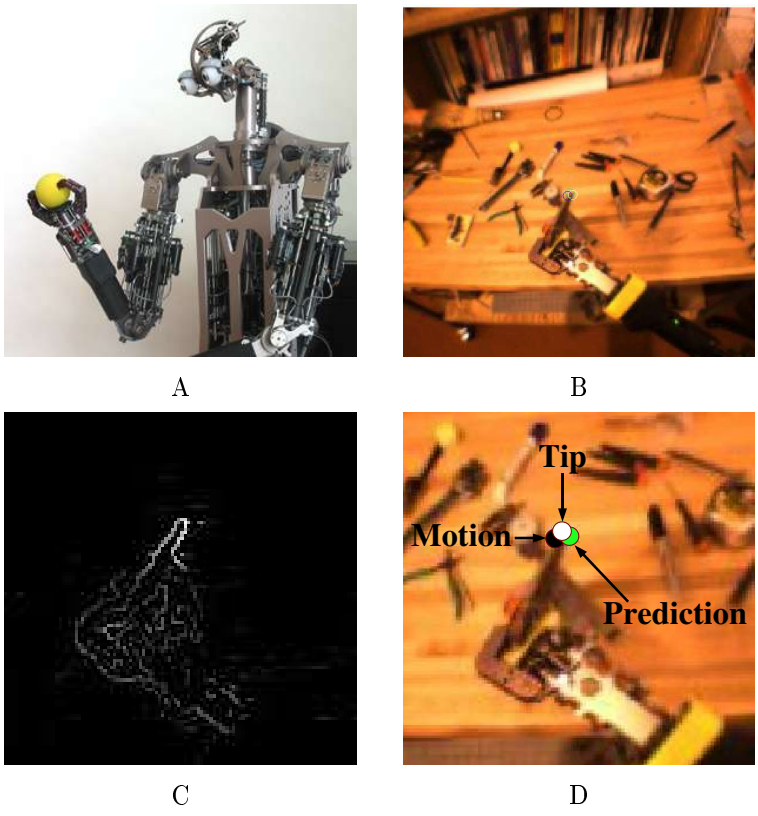

Figure 1: [A] The humanoid robot, Domo, used in this work. [B] A typical view from the robot's camera of the hand holding a pair of pliers. A naturally lit, cluttered background ensures a non-trivial unstructured environment for perception. [C] The tool-tip is detected as the maximum of the motion estimator's weighted edge map. [D] The raw motion-based detection (black), the hand-labeled tool tip (white), and a prediction based on the estimated tool position (grey) 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.

For a wide variety of tools and tasks, control of the tool's endpoint is sufficient for its use. For example, use of a screwdriver requires precise control of the tool blade relative to a screw head but depends little on the details of the tool handle and shaft. Radwin [26] describes 19 categories of common power and hand tools. Approximately 13 of these tool types feature a distal point on the tool which can be considered the primary interface between the tool and the world. 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. These basic skills can enable rudimentary use of the tool and assist further learning by helping the robot to actively test and observe the endpoint. We show successful results for this estimation method using the humanoid robot pictured in Figure 1A 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 a tool held in the hand, 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 position 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 typically use specialized, well-modeled tools that are rigidly mounted using exchangeable end-effectors [21]. For example, Ruf [27] 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 [29] 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 [16]. These approaches are not likely to scale to the wide variety of human tools since they depend on detailed models.

In the work of Brooks [8], perception is directly coupled to action in the form of modular behaviours that eschew complex intermediate representations. Our method relates to this work in three ways. First, the robot's action is used to simplify the perceptual problem. Second, the method directly detects the tip of the tool without requiring an initial segmentation of the tool or reconstruction of its shape. Third, our approach is suitable for implementation as a real-time modular behaviour.

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 [12, 22] 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 [24] applied image differencing for detection of 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 [23]. Kemp [18] 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 [34]. For example, Duric [10] looked at associating a tool's function with its prototypical motion. Robots that can actively learn about tool use have 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 [31] 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 extracted from the image or that the tool features are known in advance.

Our tool tip detection method makes use of optic flow to build 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 [15, 33, 32, 25, 30] and modeling measurement error from block matching [1, 28]. We are currently unaware of previous work outside of [18] that estimates the significance of observed motion in real-time using our specific method.

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 [3]. These methods, however, typically rely on a well modeled background to cleanly segment the object, simple platform motion, and occlusion free views of the object. More generally, our estimation technique relates to the well-studied area of 3D estimation from multiple views [13].

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 while the hand is moving. This ignores 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 tends to increase the speed of the tool tip in the image relative to background motion. 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 From 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,

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

Figure 3: This figure shows a visualization of block matching with a Gaussian error model. The parameter values are identical to those used for this paper.

This 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.

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 [9] in order to achieve real-time rates and reduce the use of uninformative points.

We compare each of the edge points in the first image to each location within an $11x11$ search window in the second image. As visualized in Figure 3, 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 error value, s_b, and find the covariance matrix, C, for a Gaussian model of the matching error around this best match using

$$C = \alpha I + \frac{1}{\sum_j w_j} \sum_j w_j (p_j - p_b)(p_j - p_b)^T \text{ where } w_j = \begin{array}{l} 1 \text{ if } s_j < s_b + \tau \\ 0 \text{ if } s_j \geq s_b + \tau \end{array}, \quad (2)$$

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. We set $\tau = 200$. This hand-tuned value 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 avoid over-confidence in the estimated error distribution, we also add αI to C, which is equivalent 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

$$X_1 = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n & v_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_n & v_n & 1 \end{bmatrix}, \; a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}, \; x_2 = \begin{bmatrix} u_1 + t_{u1} \\ v_1 + t_{v1} \\ u_2 + t_{u2} \\ v_2 + t_{v2} \\ \vdots \\ u_n + t_{un} \\ v_n + t_{vn} \end{bmatrix}, \quad (3)$$

where 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 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,

$$\Sigma^{-1} = \begin{bmatrix} C_1^{-1} & 0 & 0 & \cdots & 0 \\ 0 & C_2^{-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & C_n^{-1} \end{bmatrix}. \quad (4)$$

Due to the sparse block form of the matrices, these equations can be significantly simplified and solved in real-time as shown 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,

$$\mathcal{N}(\Sigma, X_1 a)(x_2) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(X_1 a - x_2)^T \Sigma^{-1} (X_1 a - x_2)}. \quad (5)$$

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 the motion of each edge point fits the model by calculating the Mahalanobis distance, h_i, between the best match translation vector, t_i, and the translation predicted by the model, $A \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$, with

$$h_i = \left(\left(A \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} - \begin{bmatrix} u_i + t_{ui} \\ v_i + t_{vi} \end{bmatrix} \right)^T C_i^{-1} \left(A \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} - \begin{bmatrix} u_i + t_{ui} \\ v_i + t_{vi} \end{bmatrix} \right) \right)^{\frac{1}{2}}. \quad (6)$$

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

These distances give 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. If this weight is below a conservative threshold, the detection is ignored. 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 will tend to be more certain with corners.

3.2 Probabilistic Estimation of the 3D Tool Tip Position

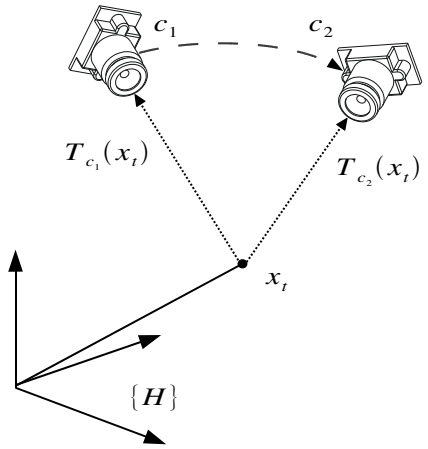

Figure 4: The geometry of the tool tip 3D estimation problem. With respect to the hand's coordinate system, $\{H\}$, the camera moves around the hand. In an ideal situation, only two distinct 2D detections would be necessary to obtain the 3D estimate. Given two observations with kinematic configurations c_1 and c_2, the tool tip, x_t, appears in the image at $T_{c_1}(x_t)$ and $T_{c_2}(x_t)$.

In the previous section 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 points in a series of images with distinct views, we use the robot's kinematic model to combine these 2D points into a single 3D estimate of the tool tip's position in the hand's coordinate system. With respect to the hand's coordinate system, $\{H\}$, the camera moves around the hand while the hand and tool tip remain stationary. This is equivalent to a multiple view 3D estimation problem where we wish to estimate the constant 3D position of the tool tip, x_t, with respect to $\{H\}$. In an ideal situation, only two distinct 2D detections would be necessary to obtain the 3D estimate, as illustrated in Figure 4. However, we have several sources of error, including noise in the detection process and an imperfect kinematic model.

A variety of approaches would be appropriate for this estimation, since only three parameters need to be estimated and we have plenty of data from a mod-

erately noisy source. In this paper, we estimate x_t by performing maximum likelihood estimation with respect to a generative probabilistic model.

We first model the conditional probability distribution, $p(d_i|x_t, c_i)$, which gives the probability of a detection at a location in the image, d_i, given the true position of the tool tip, x_t, and the robot's configuration during the detection, c_i. We model the detection error that is dependent on x_t with a 2D circular Gaussian, $\mathcal{N}_t$, centered on the true projected location of the tool tip in the image, $T_{c_i}(x_t)$, with variance σ_t. T_c is the transformation that projects the position of the tool tip, x_t, onto the image plane given the configuration of the robot, c_i. T_{c_i} is defined by the robot's kinematic model and the pin hole camera model for the robot's calibrated camera. This 2D Gaussian error model on the image plane can coarsely represent a number of error sources, including the selection of motion edges around the ideal location, and inaccuracies in the kinematic model. We mix this Gaussian with another 2D Gaussian, $\mathcal{N}_f$, centered on the image with mean 0 and a large variance σ_f. This Gaussian accounts for false detections across the image that are independent of the location of the tool tip. In summary,

$$p(d_i|x_t, c_i) = (1-m)\mathcal{N}_t(T_{c_i}(x_t), \sigma_t^2 I)(d_i) + m\mathcal{N}_f(0, \sigma_f^2 I)(d_i), \tag{7}$$

where m is the mixing parameter for these two Gaussians.

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

$$p(d_1 \ldots d_n|x_t, c_1 \ldots c_n) = \prod_i p(d_i|x_t, c_i). \tag{8}$$

Using Bayes rule we have

$$p(x_t|d_1 \ldots d_n, c_1 \ldots c_n) = \frac{p(d_1 \ldots d_n|x_t, c_1 \ldots c_n)p(x_t, c_1 \ldots c_n)}{p(d_1 \ldots d_n, c_1 \ldots c_n)}. \tag{9}$$

Since we are only looking for relative maxima, we can maximize

$$p(d_1 \ldots d_n|x_t, c_1 \ldots c_n)p(x_t, c_1 \ldots c_n). \tag{10}$$

We also assume 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, so that $p(x_t, c_1 \ldots c_n) = p(x_t)p(c_1 \ldots c_n)$. 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

$$\begin{aligned} \widehat{x_t} &= \text{Argmax}_{x_t} \left(p(d_1 \ldots d_n|x_t, c_1 \ldots c_n)p(x_t) \right) \\ &= \text{Argmax}_{x_t} \left(\log\left(p(x_t)\right) + \textstyle\sum_i \log\left(p(d_i|x_t, c_i)\right) \right). \end{aligned} \tag{11}$$

We define $p(x_t)$ as being approximately zero for positions that are inside the robot's hand or wrist, or are greater than 1 meter from the center of the hand. $p(x_t)$ is uniform everywhere else.

A variety of methods could be used to find our estimate $\widehat{x_t}$ given expression 11, including gradient ascent and brute force sampling. We use the Nelder-Mead Simplex algorithm implemented in the open source SciPy scientific library [17] to optimize this cost function. More efficient optimization algorithms are applicable, but this algorithm is easy to use since it only requires function evaluations. Even though each evaluation of the cost function requires $O(n)$ computation, where n is the number of detections, we found it to converge quickly given our small set of moderately noisy observations.

There are many sources of error that we ignore in our model, including uncertainty about the hand's rotation (which will have a larger impact on long objects), the projection dependent aspects of the kinematic uncertainty, and uncertainty in the temporal alignment of the kinematic configuration measurements and the motion-based detections.

4 Experimental Results

The described approach was evaluated on a 29 DOF humanoid robot at the MIT CSAIL Humanoid Robotics Group [11]. 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) for a wide field-of-view and high resolution imaging.

4.1 Setup

Experiments were conducted on a variety of tools with differing lengths and endpoints, see Figure 2. For each experiment, the 11 DOF kinematic chain from the camera to the robot wrist was servoed to maintain a fixed pose that ensured tool visibility in the wide-angle camera. The tool was placed in the robot's hand and the 2 DOF (pitch,roll) of the wrist were ramped smoothly to random positions in the range of ± 60 degrees for a short duration. The robot's joint angles and camera images were sampled at $30hz$. Approximately 500 samples (15 seconds of motion) were captured for each tool and randomly distributed into a training set of 400 samples and a test set of 100 samples. We then hand labeled the tool tip location for each frame of the test set.

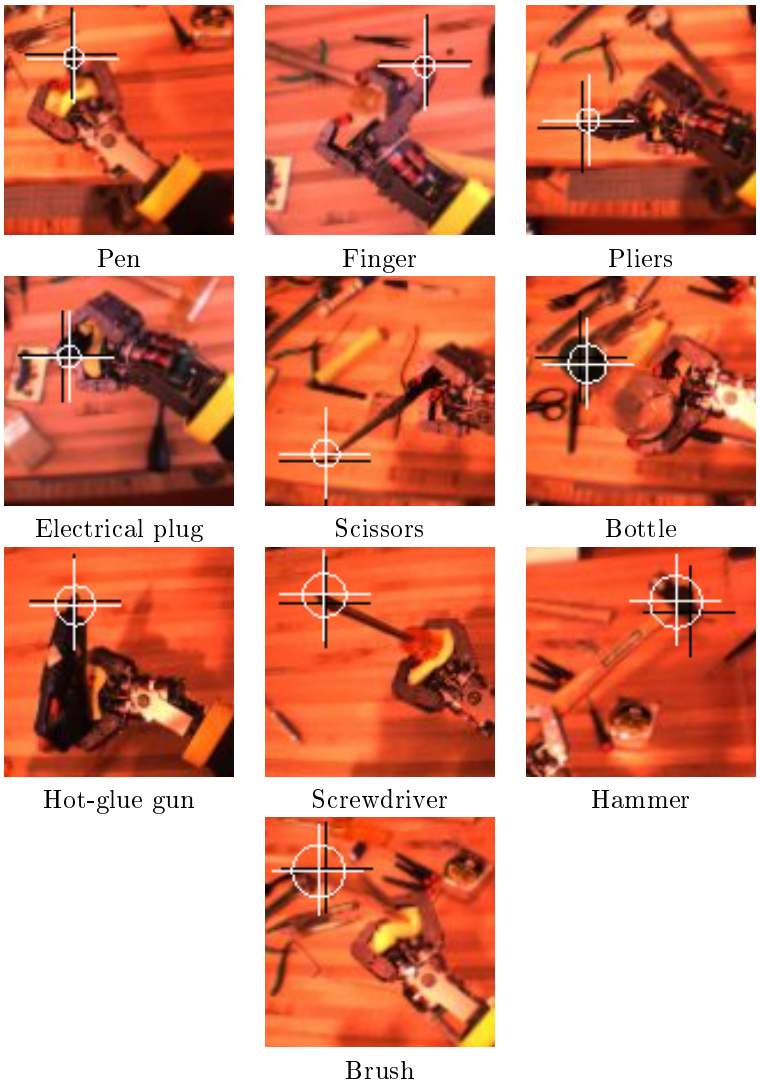

Figure 5: An example of the tip prediction for each tool. The white cross is centered at the prediction point and measures 40 pixels across for scale. The radius of the white circle indicates the tool's mean pixel error. The black cross indicates the hand labeled tool tip.

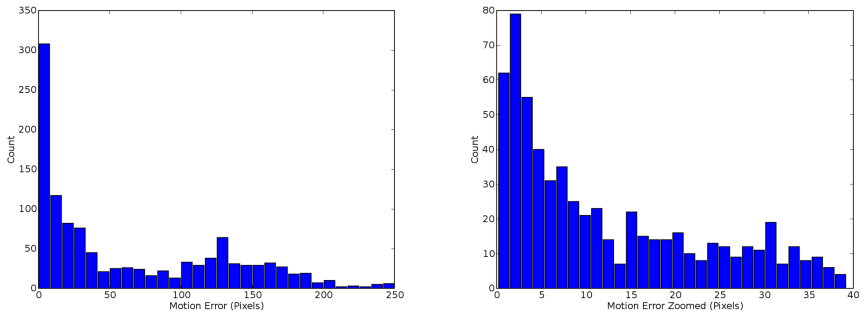

Figure 6: [Left] Error histogram, in pixels, for raw motion-based detection of the tool tip with respect to the hand-labeled tool tip for all tools. [Right] Detailed view of the left graph.

4.2 Tool Tip Detection

Visual detection of the tool tip was computed using the motion model from Section 3.1. In our experiments, the localization was computed offline for each pair of sequential images, though real-time rates are achievable with little adaptation. As shown in Figure 1B, a naturally lit, cluttered background was used to ensure a non-trivial unstructured environment for perception. The detection method is noisy, but as shown in Figure 6, 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 was estimated as close to identity. Without modification the model could be used in situations with a non-stationary camera and other causes of global affine motion.

4.3 Tool Position Estimation

The position estimation accuracy was evaluated by first estimating the 3D tool position in the hand on the training data set as described in Section 3.2. We used the parameter values $\sigma_t = 5.0$, $\sigma_f = 150.0$, and $m = 0.5$. The 3D position 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 is indicative of inaccuracies in the kinematic model and the camera model. The algorithm performs favorably with respect to this baseline error. Figure 7 illustrates the mean prediction error, in pixels, across the set of tools. Figure 5 illustrates the typical tip prediction for each tool.

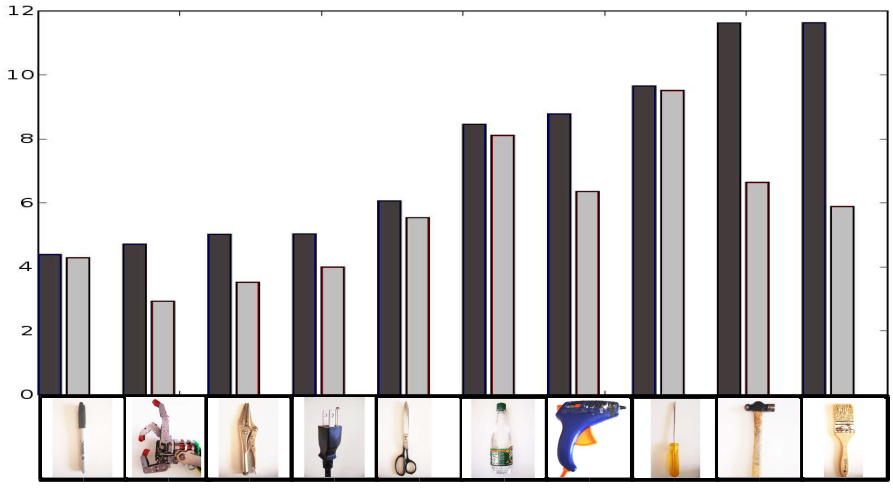

Figure 7: The mean prediction error, in pixels, for each tool. The 3D tool tip position in the hand is estimated using two data sets: 2D motion-based tool tip detection and hand-labelled tool tips. The 3D positions for both estimates are then projected onto the image plane for each sample in the test set and compared to the hand labelled location. The left (dark) bar indicates the detector error and the right (light) bar indicates the hand labelled, baseline error. The baseline errors are an indication of inaccuracies in the kinematic model and the camera model.

4.4 Discussion of the Results

As Figure 5 illustrates, the prediction performed well. The wide angle 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 7 demonstrate. On the Domo robot, the transform T_c was computed from a hand tuned model. The electro-mechanical details of the robot make precise calibration difficult. An autonomous method for the hand-eye calibration problem [14, 2] could potentially reduce this error component. We analyzed the estimated 3D position 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 where the tool becomes a natural extension of the robot's body. We trained each estimator on a data set of 400 samples which is 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 position 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 position would be indeterminate. Finally, it is useful to limit the rotational velocity of the wrist to reduce significant motion blur at the tool tip and to avoid moving outside the search window used in block-matching.

5 Discussion

We have presented a straight-forward approach to detecting the end point of a tool in a robot hand and estimating its 3D position. The strength of our approach is that it assumes little prior knowledge about the tool or its position 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 many human tools depends on the tool's endpoint. 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.

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

Our method requires that the robot already have the tool firmly in its grasp. It is appropriate for applications where the tool is placed within the hand of the robot, such as by an instructor who is teaching the robot. It may also be useful in situations where the robot has already perceived the tool in order to grasp it. The details of the tool's position in the hand after grasping may be uncertain. Also, many plausible grasping methods could be successful without identifying the endpoint of the tool. If the grasp happens to occlude the tool tip our method will not work, and may instead detect the end of the tool's handle. Although the dependency of the method on the details of the grasp merits further investigation, we believe that the method will tend to work with stable grasps on human tools.

Other perceptual cues could be beneficially integrated with this method, such as stereo vision and extra visual features. Motion does, however, have some especially beneficial properties for this type of detection. First, motion helps us to find elements of the world that are controlled by 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, stereo could not detect which points are under the hand's control. Second, by moving the hand and tool we are able to observe them from several distinct views, which reduces sensitivity to the particular position of the hand and 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 reduce the method's dependence on the kinematic model. Likewise, motion could be further exploited in the estimation

process by tracking features over multiple frames or otherwise incorporating a model of the expected dynamics of the tool tip.

Our work affords many avenues for further exploration. The 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 a robot hand to an object [20] and these approaches could be readily extended to include control of the tool tip. In addition, the estimate of the tool's 3D position in the hand can be applied to a traditional Cartesian space control scheme.

The approach described in this paper can help the robot to actively test and observe the endpoint during interactions with the world. As such, it is a first step towards robots that autonomously learn to use novel, unmodeled human tools in human-centric environments.

A Weighted Linear Least Squares Simplification

In this appendix we provide the results of simplifying the weighted linear least squares formulation we use for motion processing. As explained within subsection 3.1 of 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

$$X_1^T \Sigma^{-1} X_1 = \sum_i \begin{bmatrix} u_i^2 c_{1i} & . & . & . & . & . \\ v_i u_i c_{1i} & v_i^2 c_{1i} & . & . & . & . \\ u_i c_{1i} & v_i c_{1i} & c_{1i} & . & . & . \\ u_i^2 c_{3i} & u_i v_i c_{3i} & u_i c_{3i} & u_i^2 c_{4i} & . & . \\ v_i u_i c_{3i} & v_i^2 c_{3i} & v_i c_{3i} & v_i u_i c_{4i} & v_i^2 c_{4i} & . \\ u_i c_{3i} & v_i c_{3i} & c_{3i} & u_i c_{4i} & v_i c_{4i} & c_{4i} \end{bmatrix}, \quad (12)$$

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

$$X_1^T \Sigma^{-1} X_1 = \sum_i \begin{bmatrix} c_{1i} p_i p_i^T & c_{3i} p_i p_i^T \\ c_{3i} p_i p_i^T & c_{4i} p_i p_i^T \end{bmatrix}. \quad (13)$$

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

$$X_1^T \Sigma^{-1} x_2 = \sum_i \begin{bmatrix} (u_i + t_{xi})u_i c_{1i} + (v_i + t_{yi})u_i c_{3i} \\ (u_i + t_{xi})v_i c_{1i} + (v_i + t_{yi})v_i c_{3i} \\ (u_i + t_{xi})c_{1i} + (v_i + t_{yi})c_{3i} \\ (u_i + t_{xi})u_i c_{3i} + (v_i + t_{yi})u_i c_{4i} \\ (u_i + t_{xi})v_i c_{3i} + (v_i + t_{yi})v_i c_{4i} \\ (u_i + t_{xi})c_{3i} + (v_i + t_{yi})c_{4i} \end{bmatrix}, \tag{14}$$

which can also be more clearly written in block form with p_i, so that

$$X_1^T \Sigma^{-1} x_2 = \sum_i \begin{bmatrix} p_i c_{1i} & p_i c_{3i} \\ p_i c_{3i} & p_i c_{4i} \end{bmatrix} \begin{bmatrix} u_i + t_{xi} \\ v_i + t_{yi} \end{bmatrix}. \tag{15}$$

References

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

[2] 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.

[3] Geoff Cross Andrew W. Fitzgibbon and Andrew Zisserman. Automatic 3d model construction for turn-table sequences. In R. Koch and L. VanGool, editors, *Proceedings of SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, volume 1506 of *Lecture Notes in Computer Science*, pages 154–170. Springer Verlag, June 1998.

[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] Rodney A. Brooks. *Cambrian Intelligence*. MIT Press, Cambridge, MA, 1999.

[9] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

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

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001.

[18] 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.

[19] O. Khatib, Brock O. Chang K. Yokoi, K., and A. Casal. Robots in human environments: Basic autonomous capabilities. *International Journal of Robotics Research*, 18(684), 1999.

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

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

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

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

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

[25] 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.

[26] R.G. Radwin and J.T. Haney. An ergonomics guide to hand tools. Technical report, American Institutional Hygiene Association, 1996. http://ergo.engr.wisc.edu/pubs.htm.

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

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

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.