Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

Integrated Tracking and Control Using Condensation-based Critical-Point Matching

Brad Chambers and Seth Hutchinson
Beckman Institute
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801-2325

Abstract— **Image matching via multiresolution critical-point hierarchies has been shown to be useful in feature point selection, real-time tracking, volume rendering, and image interpolation. Drawbacks of the method include computational complexity and a lack of constraints on rigid motion. In this paper we present a method by which robot end-effector velocities are tracked using the Condensation algorithm and critical-point image observations. By using a window-based approach, we immediately reduce complexity while imposing constraints on camera motion. We show that the critical-point observations are successful in estimating camera motion by evaluating the similarity of sample windows.**

I. Introduction

The problem of tracking has been a focus of computer vision research for many years now. One of the first applications was in radar imagery, but many modern-day tasks benefit from fast and efficient tracking algorithms that are robust to the uncertainties inherent to these environments. Throughout its evolution, tracking research has been focused on two vital areas: object definition (appearance), and object estimation (parameter estimation). Methods of modeling and evaluating object appearance and methods of estimation vary widely and depend on the problem space and available assumptions.

There are three basic approaches for defining an object's appearance for the purposes of tracking: segmentation-based, contour-based, and template-based. The approaches range from dealing with the problem at a low level (segmentation) to a high level (template) and each has its own advantages and disadvantages. Segmentation-based trackers, or "blob trackers," are in general easy to implement and fast to operate, but may lack in accuracy for some applications. Templates are computationally expensive but offer a great deal of accuracy [1]–[3]. Active contours or "snakes" are often a good compromise of speed and robustness and are well-suited to deformable objects [4].

Kalman filtering [5] has been a mainstay in tracking and estimation. With the introduction of the Condensation algorithm (a particle filter) [6], many tracking researchers have shifted their focus to this approach that uses multimodal probability densities and allows for multiple hypotheses of predicted parameters. Both Kalman filtering and Condensation have shortcomings, however, and addressing these issues has been a priority. Condensation can be improved using importance sampling to prevent wasted samples due to bad proposal distributions [7].

Normalized weights of two face tracking algorithms [8], [9] are used to form the observation model for a Condensation tracker in [10]. The use of two observations in tandem is beneficial because the layer of redundancy covers shortcomings of either one of the individual observations when used alone. Recently, researchers have exploited the advantages of the different methods by combining trackers so that where one tracker may fail another tracker will excel [7], [11], [12].

Image matching via multiresolution critical-point filters (CPFs) [13] has been shown to be useful in real-time tracking, volume rendering, object recognition, stereo photogrammetry, and image interpolation. The CPFs are nonlinear filters that preserve critical-points in images, keeping them crisp rather than blurring the image. We present a new window-based image matching method that makes fundamental use of CPFs. This approach tracks image features from a different topological standpoint than our previous point-based method [14]. The point-based matching method uses the hierarchical structure of the CPFs to iteratively search for individual point correspondences. Each of these mappings carries an energy that we can then use to indicate the quality of the mapping. The window-based matching method operates at an arbitrary resolution, within the hierarchies, to match small regions between images.

Visual servo control is the use of image data in the closed loop control of a robot end-effector's position or velocity. There are three general approaches to visual servo control: image-based visual servo (IBVS), position-based visual servo (PBVS) [14]–[17], and hybrid methods that incorporate techniques from both IBVS and PBVS [18]–[22]. Our new method of visual tracking makes use of many of the well-understood parameters of the visual servo controller to improve tracking performance.

We have shown that image pixels can be matched between two images in a manner that reduces complexity compared to previous methods [14]. Because the point correspondences are found using an energy function, we can use this energy to reflect our certainty in the mapping and can rank the mappings accordingly.

When tracking is performed for visual servoing, there is at least a crude estimate describing the motion of the camera in 3D. This knowledge of motion in 3D can be used as additional information to improve tracking. Because we have a motion model, we can use Kalman filtering to

0-7803-8463-6/04/$20.00 ©2004 IEEE

predict the motion of the image data. To be more robust in general, we will instead use the popular approach of condensation for our estimation. Given a region of interest or *window* in an image, the estimate of camera motion can be used along with the well-known image Jacobian to generate sample windows for image observations in subsequent frames. CPFs along with an intensity- and color-based energy function are used to discriminate between these complex image regions. Furthermore, condensation allows us to apply a dynamic motion model to robustly track the six components of the velocity screw describing the displacement of the camera (mounted on the robot's end-effector) from the current frame to the original image.

Details of the CPFs are provided in Section II. Section III outlines our approach to window-based matching and results will be provided in Section IV. Future work and conclusions are given in Sections V and VI respectively.

II. Critical-Point Filters

CPFs are a set of nonlinear filters introduced by Shinagawa and Kunii [13]. CPFs can be used to create multiresolution hierarchies of images that maintain critical points and do not blur the image like typical multiresolution hierarchies. CPFs have been shown to be a powerful tool in image matching, object recognition, stereo photogrammetry, and volume rendering [13]. They have recently been used as a means of real-time object tracking by tracking points in the interior of an object and then recovering the shape of the object with a painting scheme [24], [25]. This allows for the tracking of an object with no prior knowledge of its shape, color, or texture.

To solve the tracking problem, we require a means of finding point correspondences between two images. We assume that the images are different, but display a reasonable amount of coherence. For example, there should only be small changes in pixel position and intensity. The point-based matching algorithm described in [14] searches for these correspondences in a hierarchical approach using an energy function. The energy function uses pixel intensity, relation to neighboring pixels, and edge intensities to evaluate potential correspondences. The energy function we use is detailed in Section II-B. The original CPFs dealt only with pixel intensities in the energy function, but have since been modified by Habuka and Shinagawa [26] to incorporate color values as well.

The CPFs are named due to their intuitive similarity with the concept of critical points in calculus. Given an input image, we create four multiresolution hierarchies that represent the maximum, minimum, and saddle points of an image determined by Eqs. (1)–(4) below. We define $p_{(i,j)}^{(l,m)}$ as the point at (i,j) in the current image, where m is the type of hierarchy computed, and l is the level within the hierarchy. Table I summarizes the four hierarchy types.

$$p_{(i,j)}^{(l,0)} = \min\left[\min\left(p_{(2i,2j)}^{(l+1,0)}, p_{(2i,2j+1)}^{(l+1,0)}\right), \min\left(p_{(2i+1,2j)}^{(l+1,0)}, p_{(2i+1,2j+1)}^{(l+1,0)}\right)\right] \quad (1)$$

$$p_{(i,j)}^{(l,1)} = \max\left[\min\left(p_{(2i,2j)}^{(l+1,1)}, p_{(2i,2j+1)}^{(l+1,1)}\right), \min\left(p_{(2i+1,2j)}^{(l+1,1)}, p_{(2i+1,2j+1)}^{(l+1,1)}\right)\right] \quad (2)$$

$$p_{(i,j)}^{(l,2)} = \min\left[\max\left(p_{(2i,2j)}^{(l+1,2)}, p_{(2i,2j+1)}^{(l+1,2)}\right), \max\left(p_{(2i+1,2j)}^{(l+1,2)}, p_{(2i+1,2j+1)}^{(l+1,2)}\right)\right] \quad (3)$$

$$p_{(i,j)}^{(l,3)} = \max\left[\max\left(p_{(2i,2j)}^{(l+1,3)}, p_{(2i,2j+1)}^{(l+1,3)}\right), \max\left(p_{(2i+1,2j)}^{(l+1,3)}, p_{(2i+1,2j+1)}^{(l+1,3)}\right)\right] \quad (4)$$

A. Hierarchies

We compute four multiresolution hierarchies of depth d for each input image (we will refer to them as *source* and *destination* images), where each subimage within a hierarchy is $2^l * 2^l \quad (1 \leq l \leq d)$ pixels, and l represents the level of the hierarchy. A typical scenario is to use an original image of $256 * 256$ pixels, allowing $d = 8$ levels in each hierarchy. Figure 1 shows the resulting hierarchies $m = 0, 1, 2, 3$ (from right to left), at several levels $l = 8, 3, 2, 1$ (from top to bottom) after applying these four CPFs to the test image. Some images are scaled to improve clarity of the illustration. The CPFs preserve critical points in each of the respective hierarchies rather than blurring the image. The eyes and mouth are seen in the low-resolution images of the minimum hierarchy ($m = 0$), while the sky and highlights on the shirt and hair are seen in the low-resolution images of the maximum hierarchy ($m = 3$). The saddle point hierarchies preserve the skin tone areas in low-resolution images.

Parent-child relationships are defined in a hierarchy as follows. If p denotes the pixel (i,j) at level l, the *parent* P is the pixel (i',j') in the level $l-1$ such that $(i',j') = \left(\left\lfloor \frac{i}{2} \right\rfloor, \left\lfloor \frac{j}{2} \right\rfloor\right)$. Conversely, p is a *child* of P.

B. Energy in a Pixel Match

The energy associated with a match between a source and destination pixel is a function of pixel intensity, hue and saturation, and values related to the edge intensities in the images. Individual terms are weighted to contribute to an overall energy that is a direct indicator of the quality of a match. Throughout this section we will refer to the point in the source image as p and the candidate point in

TABLE I
HIERARCHY TYPES.

m	Type
0	minimum
1	saddle point
2	saddle point
3	maximum

the destination image as q. We drop the previously used superscripts and subscripts, but it should be noted that all energy terms are computed between p and q for each level l in each hierarchy m.

The intensity term is based on the difference in intesity of p and q. The intensity distance between the two points is defined as

$$C_I = \left|I(p) - I(q)\right|^2, \tag{5}$$

in which $I(p)$ and $I(q)$ are the intensity values of p and q, respectively.

We associate a matching energy based on hue and saturation by the equation

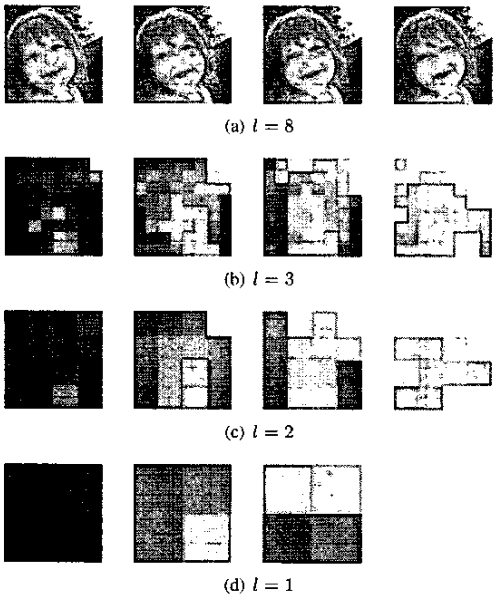
$$C_{HS} = \left(S(p)\cos\left(2\pi H(p)\right) - S(p)\cos\left(2\pi H(p)\right)\right)^2 + \left(S(q)\sin\left(2\pi H(q)\right) - S(q)\sin\left(2\pi H(q)\right)\right)^2 \tag{6}$$

in which the hue and saturation (H and S) are defined by Eqs. (7) and (8):

$$H = \cos^{-1}\left(\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{((R-G)^2+(R-B)(G-B))}}\right) * \frac{180}{\pi} \tag{7}$$

$$S = 1 - \frac{\min(R,G,B)}{R+G+B} \tag{8}$$

R, G, and B represent the red, green, and blue components of the pixels in Eqs. (7) and (8).



(a) $l = 8$

(b) $l = 3$

(c) $l = 2$

(d) $l = 1$

Fig. 1. Original image and resulting hierarchies after applying CPF. From top to bottom: $l = 8, 3, 2, 1$; from right to left: $m = 0, 1, 2, 3$.

TABLE II
COEFFICIENTS USED IN THE ENERGY FUNCTION.

Coefficient	Value
ψ	0.0125
η	0.5
λ	1
θ	2

Together the total energy related to pixel intensity, hue, and saturation is defined as

$$C = C_I + \psi C_{HS}, \tag{9}$$

in which ψ is a parameter used to determine the contribution of the hue and saturation term to C. The method of [13] used an autotuning procedure to determine a suitable value for ψ and other parameters. We will use static parameter values summarized in Table II.

To increase the accuracy of the mapping, we apply two edge-detection (Sobel) filters to the source and destination images. These two filters create horizontal and vertical edge images in both the source and destination images at the finest level d. The horizontal and vertical edge images are denoted by $edge_{(i,j)}^{(d,h)}$ and $edge_{(i,j)}^{(d,v)}$, respectively. We generate multiresolution hierarchies from $edge_{(i,j)}^{(d,h)}$ and $edge_{(i,j)}^{(d,v)}$ by averaging the value of the pixels from the filter at level $l+1$, giving

$$edge_{(i,j)}^{(l,h)} = \frac{1}{4}\left(edge_{(2i,2j)}^{(l+1,h)} + edge_{(2i,2j+1)}^{(l+1,h)} + edge_{(2i+1,2j)}^{(l+1,h)} + edge_{(2i+1,2j+1)}^{(l+1,h)}\right) \tag{10}$$

and

$$edge_{(i,j)}^{(l,v)} = \frac{1}{4}\left(edge_{(2i,2j)}^{(l+1,v)} + edge_{(2i,2j+1)}^{(l+1,v)} + edge_{(2i+1,2j)}^{(l+1,v)} + edge_{(2i+1,2j+1)}^{(l+1,v)}\right) \tag{11}$$

for the horizontal and vertical filters respectively at level l ($1 \leq l < d$). The edge hierarchies should not be thought of as structural elements, but rather as spatial derivatives that give us variation in intensities, as this is likely to be a persistent feature in source and destination images.

The edge distance between the two points at each level of the hierarchy is

$$E_{(p,q)}^{(l,m)} = \frac{1}{4}\left[\left(edge_{(i,j)}^{(l,h)} - edge_{(i',j')}^{(l,h)}\right)^2 + \left(edge_{(i,j)}^{(l,v)} - edge_{(i',j')}^{(l,v)}\right)^2\right], \tag{12}$$

which is the total energy related to the edge for a source pixel p at (i,j) in the source image and candidate pixel q at (i',j') in the destination image. E is multiplied by $\frac{1}{4}$ because in Chapter [14] the edge energy will be included

in the total energy four times (once for each CPF hierarchy $m = 0, 1, 2, 3$).

With the above elements, we can define the total energy of the match between p and q for level l in hierarchy m to be

$$U = \lambda C + \theta E, \tag{13}$$

where λ and θ are constants. Again, these parameters were found using an autotuning procedure in [13]. In [14], we will list the parameter values used in our experiments. We want to find the mapping of a pixel that provides a minimum value to this energy function.

III. Window-Based Matching

We have shown in [14] how the CPFs can be used to match individual points in two images. On average, the individual mappings reflect the correct motion between frames, but many points are needed for this estimate to be reliable. Rather than compute the energy between individual points, we can use a slight variation of our energy function to compute the energy between image regions or *windows* in the source and destination image.

Given a preselected window in the source image, we select the lowest energy window in the destination image as the one best representing the current state. We could take an exhaustive approach and compute this energy for every possible window in the destination image, but this is computationally expensive. Instead, we use the condensation algorithm to represent the conditional density of the window's state given image observations. Condensation uses factored sampling, a means of representing complex distributions with a set of samples. Our sample windows are drawn from the prior density, and our energy function weights each sample to alter our representation of the conditional density. In Section III-A we will provide an overview of the condensation algorithm. Our method of tracking using condensation and CPFs is presented in Section III. Results are given in Section IV.

In related work, the probabilistic data association filter (PDAF) [11] is used to track objects in the presence of occlusion, distractors, and agile motion. The method uses sample windows to find the best match in an image to some template, perhaps a similar region in a previous frame. The Kalman filter is used to generate the sample windows that are evaluated using the methods of homogeneous regions, contours, and textures to determine the closest match to the template.

A. Condensation

The state parameter $\mathbf{x}$ describing our tracked object can be a simple representation, such as the x and y location of the center of the object, or more complicated as with B-spline curves. We will use the six terms of the velocity screw common to visual servoing applications:

$$\mathbf{r} = (T_x, T_y, T_z, \omega_x, \omega_y, \omega_z)^T. \tag{14}$$

The velocity screw represents the velocity of the camera frame, composed of a linear velocity $\mathbf{v} = (T_x, T_y, T_z)^T$ and angular velocity $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$.

In many prediction strategies we assume an n-variate Gaussian distribution for $\mathbf{x}$. This approach often works well, but the Gaussian assumption leads to failure in a number of situations. For example, when the object is fully or partially occluded for several frames, or when other forms of visual clutter are present, trackers such as Kalman-based trackers may lose the tracked object. This is because a measurement unrelated to the actual object is present and hence affects future predictions.

Condensation is a variation of particle filtering developed by Isard and Blake [6] for tracking of curves amongst visual clutter. Condensation achieves this by representing the conditional density $p(\mathbf{x}|\mathbf{z})$ of the target object's state $\mathbf{x}$ given image observations $\mathbf{z}$. This distribution is often multimodal and difficult at best to represent in closed-form. Instead, we can use the prior $p(\mathbf{x})$ and measurement density $p(\mathbf{z}|\mathbf{x})$ to approximate the distribution using factored sampling, a way of representing the distribution with a set of samples. Since its publication, condensation has been a popular approach to tracking thanks to its ability to represent multimodal distributions of $p(\mathbf{x}|\mathbf{z})$ using factored sampling, which proves particularly useful in visual clutter.

The state of the modeled object at a discrete time t is denoted $\mathbf{x}_t$ and the set of image features is $\mathbf{z}_t$. The history of each of these random variables is denoted $\mathcal{X}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ and $\mathcal{Z}_t = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$. An assumption is made that the object dynamics form a temporal Markov chain so that

$$p(\mathbf{x}_t|\mathcal{X}_{t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{15}$$

Therefore, object dynamics are defined by the conditional density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$.

The conditional density $p(\mathbf{z}_t|\mathbf{x}_t)$ defines the observation process at time t. Because the observation density is in general multimodal, the state density $p(\mathbf{x}_t|\mathcal{Z}_t)$ is also non-Gaussian. To evaluate the state density over time without incurring excessive computational load, a factored sampling approach can be taken.

In the iterative approach of the condensation algorithm, each weighted sample-set that represents $p(\mathbf{x}|\mathbf{z})$ at time t consists of $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}), n = 1, \ldots, N\}$. At each iteration, a sample $\mathbf{s}_t^{(n)}$ is chosen with probability $\pi_t^{(n)}$

$$\pi_t^{(n)} = \frac{p(\mathbf{z}|\mathbf{s}_t^{(n)})}{\sum_{j=1}^{N} p(\mathbf{z}|\mathbf{s}_t^{(j)})}. \tag{16}$$

The cumulative weight $c_t^{(n)}$ is the sum of all measurement weights $\pi_t^{(n)}$ up to sample n:

$$c_t^{(0)} = 0 \tag{17}$$

$$c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)} \tag{18}$$

A random number $r \in [0,1]$ is generated, and $\mathbf{s}_t'^{(n)}$ is selected by the first n for which $c_{t-1}^{(n)}$ exceeds r. In this way samples carrying high probabilities may be selected multiple times, and those with low probabilities may not be selected at all. We then apply a motion model to the selected sample to reflect its predicted change in state from the previous iteration. According to the observation model, $\pi_t^{(n)}$ is recalculated based on currect image observations. N must be picked appropriately for the problem space; a large value may give increased robustness but will be computationally inefficient, whereas a small value will give fast performance but will lead to failure more easily.

B. Our Implementation

We will assume that our target is static and that motion only occurs by moving the camera, which is mounted on a robotic arm end-effector. Our tracking will be constrained and improved by our estimated motion of the camera. We will propose and show experimental results for an observation model based on the CPFs.

We are interested in the CPFs' ability to measure the quality of a match on a small image region or window. This is beneficial in two senses: the size of the window will be smaller than the entire image, and we can use the concept of spatial and temporal coherence to only consider a direct mapping of points from one window to another (i.e., we will not be searching for matching points, only measuring the quality of each match among a number of samples). Because we assume rigid body motion and have an estimate of that motion, we can exploit the motion model and say that a window containing the region of interest will also undergo this rigid motion and contain roughly the same data in subsequent frames.

When representing the conditional density using factored sampling, it is important that the number of samples used be large enough to capture an accurate approximation of the distribution, yet small enough to minimize complexity. To make efficient use of the samples, it is important to generate relevant samples from the prior density. Condensation alone does not incorporate observations in the transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. This fact can lead to creation of samples in low likelihood areas. Importance sampling has been applied to condensation via an auxiliary tracker as a potential remedy to this problem [7]. The approach uses an auxiliary tracker in an attempt to generate relevant samples. Auxiliary trackers still require good proposal distributions such as the transition prior however. The unscented particle filter (UPF) [27] is a hybrid of the unscented Kalman filter (UKF) [28] and particle filtering that incorporates image observations in the transition prior. The advantage of this method is that the particle filter allows for multimodal distributions, while the UKF takes into account the most recent image observations. Our method uses knowledge of camera motion via an end-effector state description in the condensation framework to generate samples that will be useful in the tracking process.

In our implementation, $\mathbf{s}_t^{(n)}$ will be a vector containing the six elements $T_x, T_y, T_z, \omega_x, \omega_y$, and ω_z of the velocity screw common in visual servoing applications, which describes end-effector velocities from one pose to another. The measurement weights $\pi_t^{(n)}$ are used to weight each of the state samples using a novel observation model defined in Section III-B.2.

In the following sections, we describe our motion and observation models as well as define the extent of user input needed to initialize the tracker. Pseudocode for the proposed method is shown as Fig. 2.

1) Motion model: We apply a dynamic motion model to the samples to use our knowledge of predicted motion to enhance tracking accuracy. Our deterministic element of motion simply indicates a predicted motion. Others have used offline training or a supplementary tracker [7], [29]. There is also a stochastic element of the motion model that prevents samples from converging to a single state. When a new sample set is generated from the previous step, it is possible to pick a sample multiple times, meaning it carries a high probability. The new samples will move together after we apply the deterministic portion (also known as *drift*) of our motion model, but the stochastic element (diffusion) works to separate these samples and help avoid local minima.

The drift is a product of the object's estimated motion. Conveniently, the tracked parameters (the velocity screw) directly describe the camera motion. We use the highest weighted sample from the previous iteration $\mathbf{s}_{t-1}^{\max}$ as our state to update all samples and multiply each element according to a gain matrix K to come up with the next predicted position for each sample:

$$\mathbf{s}_t^{(n)} = \mathbf{s}_{t-1}^{(n)} - K\mathbf{s}_{t-1}^{\max}. \tag{19}$$

If individual samples from the previous iteration are

Given two images (source and destination) of size $2^d * 2^d$ pixels, an initial estimate of the velocity screw $\mathbf{r}_i$, and a source window R of size $2^l * 2^l$, where $l < d$, we can generate our initial sample-set $\mathbf{s}_0^{(n)}$ by sampling from $N(\mathbf{r}_i, \Sigma^2)$

for all t **do**
  for all n **do**
    1. Project source window vertices into destination image with $\mathbf{s}_t^{(n)}$ to obtain sample window $R'^{(n)}$
    2. Apply CPF to $R'^{(n)}$ at level, k, where $k < l$
    3. Compute energy $E_t^{(n)}$ between R and $R'^{(n)}$
    4. Update weight $\pi_t^{(n)} = \frac{1}{E_t^{(n)}}$
    5. Apply motion model to sample states
  end for
  Acquire new destination image
end for

Fig. 2. Velocity screw tracking algorithm.

selected multiple times to serve as new samples, they will undergo the same drift. After applying the drift, we separate samples and account for the uncertainty of the true distribution by letting each sample undergo a random walk. In our experiment, we use a Brownian motion model for this random walk.

We will not estimate depth at each iteration. Rather, we will select the depth as a constant that corresponds to the depth in the source pose, which is assumed to be known. By creating sample windows of various scales (a result of our random motion), we should be able to capture the correct depth over time.

We use our velocity screw estimate and source window vertices to come up with sample window vertices in the destination image according to

$$\dot{\mathbf{f}} = J_{im}(u, v, z)\mathbf{s}_t^{(n)}, \tag{20}$$

where $J_{im}(u, v, z) =$

$$\begin{bmatrix} \frac{\lambda}{z} & 0 & -\frac{u}{z} & -\frac{uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} & -v \\ 0 & \frac{\lambda}{z} & -\frac{v}{z} & \frac{-\lambda^2-v^2}{\lambda} & \frac{uv}{\lambda} & u \end{bmatrix} \tag{21}$$

is the image Jacobian in which λ is the focal length of the camera. Derivations of this can be found in a number of references including [15]– [31]. We will use $\mathbf{f} = (u, v)^T$ to represent the source window vertices and $\mathbf{f}' = \mathbf{f} + \dot{\mathbf{f}}$ to represent the corresponding destination window velocities.

2) Observation model: We propose an observation model that indicates the likelihood $p(\mathbf{z}|\mathbf{x})$ of an observation coming from the predicted state. Samples from the distribution are weighted by image observations at each iteration such that samples with a large weight represent observations with a high probability of coming from the predicted state. Those samples that do not gain support over time (i.e., brief occlusions) will eventually die out and the true distribution will remain.

To compare our $2^l * 2^l$ source window with a sample window that is defined by our sample velocity screw and is a projective transformation of the first, we must compute the inverse transformation and warp the destination window to also be $2^l * 2^l$. We use bilinear interpolation to generate these synthetic views. Given warped sample windows, we can compute the energy of pixel intensity and color across each hierarchy m at a given level l. A low total energy indicates a close matching of windows.

The energy E is a simple function of intensity and color differences between windows in the source and destination images. We drop the previously used edge intensity term because in this window-based approach we sum the individual energy terms across the region and the variation in intensity becomes less meaningful in discriminating between windows.

Critical-point filters are applied to a specified level in each window, but the full hierarchy does not need to be computed. The energy function is given by

$$U_t^{(n)} = C_{I,t}^{(n)} + C_{HS,t}^{(n)} \tag{22}$$

where $C_{I,t}^{(n)}$ and $C_{HS,t}^{(n)}$ are the intensity and hue and saturation terms, respectively, of the n^{th} sample at time t, as defined by Eqs. (23) and (24):

$$C_{I,t}^{(n)} = \sum_{i,j}\sum_{m=0}^{3} \left(I_s(i,j,m) - I_d(i,j,m)\right)^2 \tag{23}$$

$$\begin{aligned} C_{HS,t}^{(n)} \quad = \quad & \sum_{i,j}\sum_{m=0}^{3} \Big(S_s(i,j,m)\cos\left(2\pi * \left|H_s(i,j,m)\right|\right) - \\ & -S_s(i,j,m)\cos\left(2\pi * \left|H_s(i,j,m)\right|\right)\Big)^2 + \\ & + \Big(S_d(i,j,m)\sin\left(2\pi * \left|H_d(i,j,m)\right|\right) - \\ & -S_d(i,j,m)\sin\left(2\pi * \left|H_d(i,j,m)\right|\right)\Big)^2, \end{aligned} \tag{24}$$

where i and j are the coordinates of a pixel in the image, and m is the hierarchy. H_s, H_d, S_s, and S_d are the hue and saturation at i, j, and m in the source and destination windows, respectively, as defined by Eqs. (7) and (8).

Furthermore, each term is normalized (Eqs. (25) and (26)) so that we have a more accurate representation of the term's influence when weighting the sample as in [10]:

$$\sum_n C_{I,t}^{(n)} = 1 \tag{25}$$

$$\sum_n C_{HS,t}^{(n)} = 1 \tag{26}$$

The value used to weight samples should be large if the sample is good. The CPF energy function is minimized when two images are alike. Therefore, we weight our samples as in Eq. (27):

$$\pi_t^{(n)} = \frac{1}{U_t^{(n)}} \tag{27}$$

The energy should be zero if a window is an exact match, but in practice we will look to minimize the energy. By definition, the weights are normalized as in Eq. (28):

$$\sum_n \pi_t^{(n)} = 1 \tag{28}$$

3) Initialization: In our experiments, we will use the known velocity screw we use to generate the destination image as the initial velocity screw $\mathbf{r}_i$. To generate the initial sample-set $\mathbf{s}_0^{(n)}$, we sample directly from a Gaussian distribution with mean equal to the initial velocity screw $\mathbf{r}_i$ and covariance Σ. N samples are generated for each degree of freedom $T_x, T_y, T_z, \omega_x, \omega_y$, and ω_z. We will select a template window $(2^l * 2^l)$ in the source image, where $2^l < 2^d$ and 2^d equals the width of the source and destination images.

IV. Results

Our experimental setup involves a PUMA 560 robotic arm with Sony DFW-V500 digital camera mounted on the end-effector. Computation is performed on a 2.4 GHz PC running uncompiled Matlab code. The camera is assumed to be calibrated. A source image is obtained at this point before the end-effector is then displaced to an offset position where the first destination image is obtained and the servoing will begin. Figure 3(a) shows our source image, with the manually selected $64 * 64$ pixel source window outlined.

We will now evaluate a simple translation of 10 mm in the x and -10 mm in the y directions of the robot's world coordinate frame. Using our estimate of the camera motion, we will generate $n = 200$ sample windows in the destination image that should contain similar data to that of the source window.

Our first iteration is shown in Fig. 3(b) where we see that many samples are generated with a large variance (in x and y) to express some doubt in our initial prediction of the camera's motion between the first and second frame. Because we are evaluating only the simple translation with no scaling, we set the variances of the z translation and all rotations to zero for now. These variances are only used with the Gaussian distribution which provides sample windows at the initialization step. The best match window is outlined in bold.

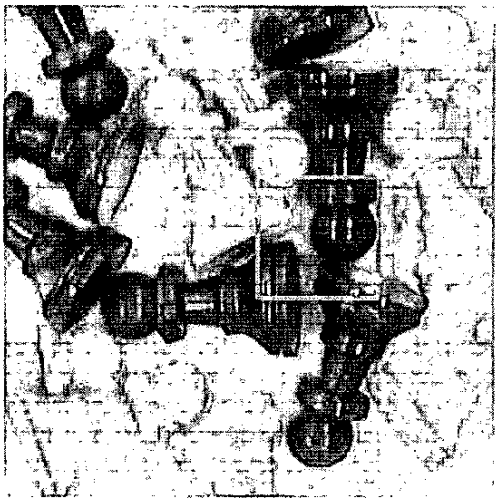
A. CPF at different resolutions

Figure 4 shows the energy of the two hundred $64 * 64$ pixel sample windows plotted against the error between the estimated pose describing each sample and the known pose used to generate the test images for $l = 1, 2, 3, 4, 5$, and 6.

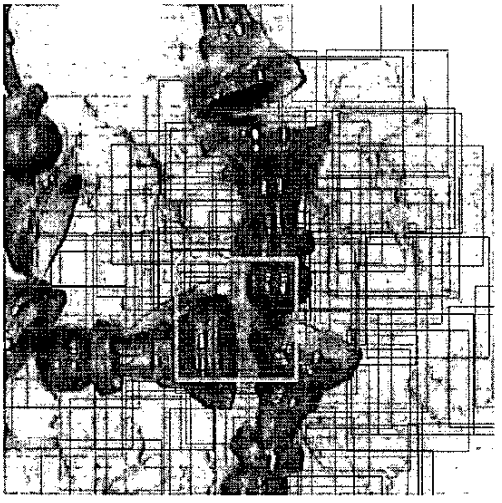
In each case we see that there is little visible discrimination between the energy of samples with a small error, but at all levels there is a clear cutoff at about 10-mm error where samples energies clearly indicate a poor match. The low energy samples appear to have a generally linear and increasing behavior with increase in error. Though a distinctly low energy sample corresponding to a distinctly small error is not always evident, each of these samples lends support to multiple hypotheses of the actual camera pose in subsequent frames as the condensation algorithm takes over. We can conclude from this that tracking may perform as accurately at low levels of resolution which offers much reduced computational complexity.

V. Future Work

In future work, we would like to further investigate the abilities of our window-based matching scheme by presenting more complex target object motions. To this point, we have not shown results for translations about x, y, and z in the world coordinate frame, nor have we demonstrated the ability of the sampling method to accurately capture scale changes.

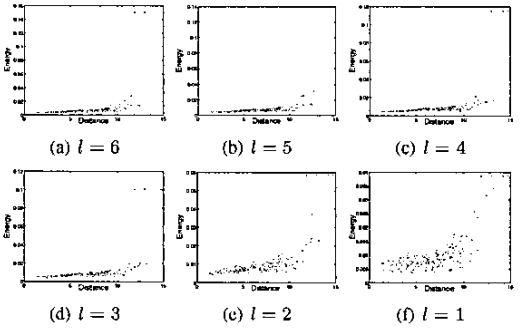


(a) Source window shown in bold outline.



(b) Initial samples and best match in bold outline (some sample windows omitted for clarity).

Fig. 3. Source and destination images with samples.



(a) $l = 6$ (b) $l = 5$ (c) $l = 4$

(d) $l = 3$ (e) $l = 2$ (f) $l = 1$

Fig. 4. Energy of samples vs. distance from actual pose.

VI. Conclusion

To address this flaw, we proposed a window-based matching method. Because we are considering rigid body motion and a window can be considered a rigid body, we can take into account neighboring energies produced by the CPF energy function. We only need to define a region of interest or window around a part of the source image. We can then evaluate potential mappings of that window in subsequent frames. Furthermore, we exploit our knowledge of camera motion in 3D by using it in our motion model of the condensation algorithm. We can propagate the probability densities representing our estimate of the camera motion through a video sequence and reliably track the window.

We have shown once again that condensation is a powerful tracking tool. The generality of the algorithm allows us to apply novel motion and observation models to our chosen state parameterization. Our choice of parameterizing the rigid body motion by end-effector velocities allows a more meaningful description of the motion of the target. The limitations imposed on motion by the robotic arm allow us to generate samples that will not be wasted and eliminates the need for an auxiliary tracker. The observation model allows tracking of more complex and a priori unknown objects with minimal user initialization. The measurement also performs similarly at coarse and fine resolutions allowing for fast running implementations. Finally, the redundancy inherent in using four hierarchies to contribute to an overall matching score leads to a robust means of discriminating between candidate samples.

References

[1] F. Jurie and M. Dhome, "Hyperplane approximation for template matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 996–1000, July 2002.

[2] T. Kaneko and O. Hori, "Feature selection for reliable tracking using template matching," in *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR'03)*, vol. 1, June 2003, pp. 796–802.

[3] G. Hager and P. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, Oct. 1998.

[4] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes-active contour models," *Intl. Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[5] R. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME, Journal of Basic Engineering*, vol. 82 (Series D), pp. 35–45, Mar. 1960.

[6] M. Isard and A. Blake, "Condensation–conditional density propagation for visual tracking," *Intl. Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[7] ——, "Icondensation: Unifying low-level and high-level tracking in a stochastic framework," in *Proc. of European Conference on Computer Vision*, vol. 1, 1998, pp. 893–908.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (darpa)," in *Proceedings of the 1981 DARPA Image Understanding Workshop*, Apr. 1981, pp. 121–130.

[9] J. Yang, W. Lu, and A. Waibel, "Skin-color modeling and adaptation," in *Proceedings of ACCV'98*, vol. II, 1998, pp. 687–694.

[10] C. Luo, T. Chua, and T. Ng, "Face tracking in video with hybrid of Lucas-Kanade and condensation algorithm," in *Proc. of Intl. Conference on Multimedia and Expo (ICME'03)*, vol. 3, July 2003, pp. 293–296.

[11] C. Rasmussen and G. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 560–576, June 2001.

[12] N. Gupta, P. Mittal, S. Roy, S. Chaudhury, and S. Banerjee, "Condensation-based predictive eigentracking," in *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec. 2002, pp. 49–54.

[13] Y. Shinagawa and T. L. Kunii, "Unconstrained automatic image matching using multiresolutional critical-point filters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 994–1010, Sept. 1998.

[14] B. Chambers, J. Durand, N. Gans, and S. Hutchinson, "Dynamic feature point allocation using multiresolutional critical-point filters," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Oct. 2003, pp. 504–509.

[15] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[16] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 5, pp. 404–417, Oct. 1987.

[17] J. Feddema and O. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 5, pp. 691–700, Oct. 1989.

[18] P. Martinet, J. Gallice, and D. Khadraoui, "Vision based control law using 3D visual features," in *Proc. of the World Automation Congress*, vol. 3, May 1996, pp. 497–502.

[19] E. Malis, F. Chaumette, and S. Boudet, "2-1/2D visual servoing," *IEEE Trans. on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, Apr. 1999.

[20] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot, "Explicit incorporation of 2D constraints in vision based control of robot manipulators," in *Experimental Robotics VI*, ser. Lecture Notes in Cont. and Info. Sci., P. Corke and J. Trevelyan, Eds. Berlin Heidelberg: Springer-Verlag, 2000, vol. 250, pp. 99–108.

[21] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS'98)*, vol. 2, Victoria, BC, Canada, Oct. 1998, pp. 705–711.

[22] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, 2001.

[23] N. Gans and S. Hutchinson, "An experimental study of hybrid switched system approaches to visual servoing," in *Proc. IEEE Int. Conf. Robots and Automation (ICRA'03)*, vol. 3, May 2003, pp. 3061–3068.

[24] J. Durand and S. Hutchinson, "Real-time object tracking using multi-resolution critical point filters," in *Proc. IEEE Int. Conf. Robots and Automation (ICRA'03)*, vol. 2, Sept. 2003, pp. 1682–1687.

[25] J. Durand, "Real-time object tracking using multi-resolution critical point filters," Master's thesis, University of Illinois at Urbana-Champaign, 2002.

[26] K. Habuka and Y. Shinagawa, "Image interpolation using enhanced multiresolution critical-point filters," *Intl. Journal of Computer Vision*, to be published.

[27] Y. Rui and Y. Chen, "Better proposal distributions: Object tracking using unscented particle filter," in *Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR'01)*, vol. 2, Dec. 2001, pp. 786–793.

[28] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Dept. of Engineering Science, University of Oxford," Tech. Rep. RRG, Nov. 1996.

[29] H. Kang, D. Kim, and S. Bang, "Real-time multiple people tracking using competitive condensation," in *Proc. of the Intl. Conference on Pattern Recognition*, vol. 1, Aug. 2002, pp. 413–416.

[30] J. Aloimonos and D. P. Tsakiris, "On the mathematics of visual tracking," *Image and Vision Computing*, vol. 9, no. 4, pp. 235–251, Aug. 1991.

[31] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Reading, MA: Addison-Wesley, 1993.