Realtime Registration-Based Tracking via Approximate Nearest Neighbour Search

Travis Dick, Camilo Perez, Azad Shademan, Martin Jagersand
University of Alberta, Department of Computing Science.
{tdick,caperez}@ualberta.ca, {azad,jag}@cs.ualberta.ca

***Abstract*—We introduce a new 2D visual tracking algorithm that utilizes an approximate nearest neighbour search to estimate per-frame state updates. We experimentally demonstrate that the new algorithm capable of estimating larger per-frame motions than the standard registration-based algorithms and that it is more robust in a vision-controlled robotic alignment task.**

I. Introduction

An important goal of computer vision is to provide algorithms that allow cameras to be used as sensors in robotic settings; that is, to estimate useful state information from a real-time video sequence depicting the robot's environment. 2D visual tracking is specifically concerned with estimating the position and orientation of an object's image appearing in the frames of a video sequence. In many tasks, this 2D information is all that is required to achieve high performance, rather than the harder-to-estimate real world 3D pose. For example, as in visual servoing, we can accurately position a robotic manipulator by aligning 2D visual features.

Algorithms for visual tracking vary along two dimensions: their appearance model and their state estimation method. The appearance model predicts an object's appearance (position, orientation, colour, etc.) in an image as a function of some underlying state, and the state estimation method searches for a state that explains a given image in terms of the appearance model. For example, registration (or template) based tracking is a popular appearance model, where we assume that the object will appear in each frame as a warped version of a given template image. The set of warps is taken to be a parametric family, and the state estimation problem is to find warp parameters that register the template image in each frame. Typically, registration based tracking algorithms estimate warp parameters by numerically maximizing some measure of the quality of template alignment [1], [2], [3], [4]. More general appearance models might replace the single template image by a collection of template images [5], a linear subspace [6], a collection of features [7], [8], or a distribution of the relative frequencies of colours [9], [10]. Each of these appearance models, when combined with suitable state-estimation methods, give rise to tracking algorithms with different benefits.

The merits of each visual tracking algorithm should be considered when choosing which to use for a particular application. In this paper, we focus on registration based tracking, which is a good approach when the target object does not have many individual and distinct features such as corners or edges. Registration based tracking also takes into account all of the intensity information available for the object, instead of focusing on a few select regions. Often, this makes it more accurate than other methods.

In this work we introduce and experimentally validate a new state estimation method for registration based tracking that utilizes a nearest neighbour search to update the warp parameters from one frame to the next.

II. Notation and Problem Statement

We model the video stream as a sequence of grayscale image functions $I_0, I_1, \ldots : \mathbb{R}^2 \to [0,1]$ with the interpretation that $I_t(x)$ is the intensity of pixel location $x = (u,v) \in \mathbb{R}^2$ in the image captured at time t. Coordinates outside the bounds of captured images have intensity 0. We assume that we have a family of parameterized warping functions $\left\{w_\theta : \mathbb{R}^2 \to \mathbb{R}^2 \mid \theta \in \mathbb{R}^m\right\}$ that describe all the possible motions of the target object. Specifically, for a fixed set of reference points $R = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$ there exist ground-truth parameters θ_t^* such that, for each $i = 1, \ldots, n$, the warped points $\left(w_{\theta_t^*}(x_i)\right)_{t=0,1,\ldots}$ are the image coordinates of the same point of the target object in the images $I_0, I_1, \ldots$, respectively. One option would be to take θ_0^* to be the parameters of the identity warp and R to be the set of pixel locations belonging to the target object, but this is not the only choice. We call the image function $T(x) = I_0\left(w_{\theta_0^*}(x)\right)$ the template image. The goal of 2D registration based visual tracking is, for each time t, to estimate θ_t^* given the images $I_0, \ldots, I_t$, R, and θ_0^*.

We will abuse notation slightly and use the following shorthand to denote the vector of intensities obtained by sampling warped versions of each point in R:

$$I_t \circ w_\theta(R) = \left[I_t(w_\theta(x_1)) \ \ldots \ I_t(w_\theta(x_n))\right]^\top \in [0,1]^n.$$

Similarly, for any image function I, we write

$$I(R) = \left[I(x_1) \ \ldots \ I(x_n)\right]^\top \in [0,1]^n,$$

and for any warping function w_θ, we will write

$$w_\theta(R) = \{w_\theta(x_1), \ldots, w_\theta(x_n)\} \subset \mathbb{R}^2.$$

III. Related Work

A common assumption in registration-based visual tracking is image constancy, which states that as the object moves, the intensities of the points belonging to it remain constant. Formally, we assume that

$$I_0 \circ w_{\theta_0^*}(R) = I_t \circ w_{\theta_t^*}(R)$$

for every time t. The image constancy assumption leads naturally to tracking via minimizing the sum of squared intensity differences (SSD). In SSD tracking we produce our estimate $\hat{\theta}_t$ of θ_t^* at time t by minimizing

$$\hat{\theta}_t = \underset{\theta \in \mathbb{R}^m}{\arg\min} \left\| I_0 \circ w_{\theta_0^*}(R) - I_t \circ w_\theta(R) \right\|^2 \quad (1)$$

This is a challenging non-linear optimization problem and much of the registration tracking literature explores various minimization schemes. All of the related works we consider incrementally maintain an estimate, say $\hat{\theta}$, of the optimal warping parameters. We leave the time subscript off of this estimate with the understanding that it is always the most up-to-date estimate.

Hager and Belhumeur [3] approach the minimization of equation (1) by iteratively computing an increment $\Delta\theta$ and setting $\hat{\theta} \leftarrow \hat{\theta} + \Delta\theta$. The update $\Delta\theta$ is computed by minimizing a linearized form of

$$\Delta\hat{\theta} = \underset{\Delta\theta \in \mathbb{R}^m}{\arg\min} \left\| I_0 \circ w_{\theta_0^*}(R) - I_t \circ w_{\hat{\theta}+\Delta\theta}(R) \right\|^2.$$

Specifically, they set

$$\Delta\theta = J^+ \left[I_0(R) \circ w_{\theta_0^*} - I_t \circ w_{\hat{\theta}}(R) \right]$$

where J is the Jacobian of the function $f(\theta) = I_t \circ w_\theta(R)$ at $\theta = \hat{\theta}$ and J^+ denotes the Moore-Penrose pseudoinverse. This update is performed incrementally until either the estimate converges or until the next frame arrives. They show that the Jacobian J can be factored into a large constant factor and a small factor depending only on $\hat{\theta}$. When computing each increment $\Delta\theta$, only the non-constant factor needs to be recomputed and this leads to efficient tracking. They show that this Jacobian factorization is possible for up to affine 6-DOF warps.

Jurie and Dhome [11] use an identical approach except, instead of approximating the Jacobian J using finite differences, they fit hyperplanes to randomly generated data-points in a neighbourhood of $\hat{\theta}$. They demonstrate experimentally that their approach allows the algorithm to handle larger per-frame motions.

Baker and Matthews [1] propose a similar algorithm that computes an inverse-compositional update rather than an additive update. That is, they update their estimate according to $\hat{\theta} \leftarrow \hat{\theta} \circ \Delta\theta^{-1}$ where the notation $\theta \circ \varphi$ is used to denote the parameters of the composed warp $w_\theta \circ w_\varphi$ and the notation θ^{-1} is used to denote the parameters of the inverse of w_θ. For this notation to make sense, the set of parameterized warps must form a group under composition. This is not much of a restriction, though, since many interesting families of parameterized warps have this property. The update parameters $\Delta\theta$ are computed by minimizing a linearized form of

$$\Delta\theta = \underset{\Delta\theta \in \mathbb{R}^m}{\arg\min} \left\| I_0 \circ w_{\theta_0^*} \circ w_{\Delta\theta}(R) - I_t \circ w_{\hat{\theta}}(R) \right\|^2.$$

Specifically, they compute

$$\Delta\theta = J^+ \left[I_0 \circ w_{\theta_0^*}(R) - I_t \circ w_{\hat{\theta}}(R) \right]$$

where J is the Jacobian of the function $f(\Delta\theta) = I_0 \circ w_{\theta_0^*} \circ w_{\Delta\theta}(R)$ at $\Delta\theta = 0$. Again, this update is performed incrementally until either the estimate converges or the next frame arrives. In this approach, the Jacobian J is constant and can be computed before tracking begins. Baker and Matthews show that, to a first order approximation, the updates of this so-called inverse compositional algorithm are equal to the updates of the Hager and Belhumeur algorithm. This algorithm is very efficient and can be applied in the case where w parameterizes the set of homographies.

Benhimane and Malis [2] propose an algorithm that computes a compositional update. Their main contribution is to replace the Gauss-Newton like minimization procedure of the previous two algorithms with an efficient second order minimization (ESM). They update $\hat{\theta} \leftarrow \hat{\theta} \circ \Delta\theta$ where $\Delta\theta$ is computed by minimizing a second-order approximation to

$$\Delta\theta = \underset{\Delta\theta \in \mathbb{R}^m}{\arg\min} \left\| I_0 \circ w_{\theta_0^*}(R) - I_t \circ w_{\hat{\theta} \circ \Delta\theta}(R) \right\|^2.$$

Specifically, they compute

$$\Delta\theta = 2 \left(J_e + J_c \right)^+ \left[I_0 \circ w_{\theta_0^*}(R) - I_t \circ w_{\hat{\theta}}(R) \right]$$

where J_e is the Jacobian of the function $f(\theta) = I_0 \circ w_{\theta_0^*} \circ w_\theta(R)$ at $\theta = 0$ and J_c is the Jacobian of the function $g(\theta) = I_t \circ w_{\hat{\theta} \circ \theta}(R)$ at $\theta = 0$. Unlike the other methods, they must recompute J_c entirely for each update. Their method tends to converge in fewer (but more expensive) iterations and, since it is a second order approximation, it is less susceptible to local minima. Currently the ESM algorithm of Benhimane and Malis is the golden-standard for registration-based tracking.

In addition to the various optimization techniques above, there has been exploration into replacing the sum of squared intensity differences (inspired by image constancy) with new similarity measures to improve the robustness of tracking algorithms. For example, Richa et al. propose using the sum of conditional variances [12], Hasan et al. propose using the cross cumulative residual entropy [13], and Dame et al. propose using mutual information [14]. We believe that these ideas are largely orthogonal to our work and anticipate that they may be extended to our new algorithm without much difficulty. In a later section we present such an extension of the sum of conditional variances.

IV. The Nearest Neighbour Tracking Algorithm

The most distinctive feature of our algorithm is that it uses a nearest neighbour search to update the warp parameters from frame to frame. The observation behind this approach is that computing the per-frame update parameters can be reduced to recognizing warped versions of the template T. By recognition we mean: given a warped version template image function $\tilde{T}(x) = T(w_\theta(x))$, determine the underlying warp parameters θ by examining the intensities of $\tilde{T}$. The reduction goes as follows: let $\Delta\theta_t^* = \hat{\theta}^{-1} \circ \theta_t^*$ denote the optimal update parameters. If we restrict our attention to image-points belonging to the target object, then we have

$$\begin{aligned} I_t(w_{\hat{\theta}}(x)) &= I_t(w_{\theta_t^*}(w_{\Delta\theta_t^{*-1}}(x))) \\ &= I_0(w_{\theta_0^*}(w_{\Delta\theta_t^{*-1}}(x))) \quad \text{(image constancy)} \\ &= T(w_{\Delta\theta_t^{*-1}}(x)). \end{aligned} \quad (2)$$

1: Let $\theta_1, \ldots, \theta_N$ be a collection of representative per-frame updates.
2: $v_i \leftarrow T \circ w_{\theta_i}(R)$ for $i = 1, \ldots, N$.
3: $\hat{\theta} \leftarrow \theta_0^*$
4: **for** each new frame I_t **do**
5: $\quad i \leftarrow$ index of the nearest neighbour of $I_t \circ w_{\hat{\theta}}$ in $\{v_1, \ldots, v_N\}$ according to d.
6: $\quad \hat{\theta} \leftarrow \hat{\theta} \circ \theta_i^{-1}$
7: **end for**

Figure 1. Psuedocode for the nearest neighbour tracking algorithm. The details of how the representative per-frame updates θ_i should be obtained are given in section 4.1.

At time t we can sample the intensities of the partially aligned image $I_t(w_{\hat{\theta}}(\cdot))$, which we have just shown to be the result of warping the template T with the inverse of the unknown optimal update parameters. A method for recognizing warped versions of T can therefore be used to estimate the optimal update parameters.

The intuition behind our approach to the recognition problem is: when two warped templates, say $\tilde{T}_1(x) = T(w_{\theta_1}(x))$ and $\tilde{T}_2(x) = T(w_{\theta_2}(x))$, have a small distance measured by

$$d\left(\tilde{T}_1, \tilde{T}_2\right) = \left\|\tilde{T}_1(R) - \tilde{T}_2(R)\right\|_2^2,$$

we should expect the underlying parameters θ_1 and θ_2 to be similar as well. We compare the warped templates on the points in R because, for relatively small θ, we expect the majority of the points in $w_\theta(R)$ to belong to the target object in the template T, which is the condition for 2 to hold. Given a set of representative warp parameters $\{\theta_1, \ldots, \theta_N\}$, we build a table that associates to each parameter θ_i the resulting vector $T \circ w_{\theta_i}(R)$. Then, given a new warped template $\tilde{T}$, we estimate the unknown underlying parameters θ to be the those associated with $\tilde{T}$'s nearest neighbour in the table:

$$\theta \approx \theta_i \text{ where } i = \underset{i=1\ldots,N}{\arg\min}\, d\left(\tilde{T}, T \circ w_{\theta_i}\right).$$

Combining the above ideas and taking $\{\theta_1, \ldots, \theta_N\}$ to be a representative sample of the feasible per-frame updates gives the NN tracking algorithm: upon receiving the template T we construct a table containing warped templates; then, for each time t, we find the index i of the table's nearest neighbour to the partially aligned image $I_t \circ w_{\hat{\theta}}$ and update $\hat{\theta} \leftarrow \hat{\theta} \circ \theta_i^{-1}$. Pseudocode is given in figure 1.

There are three final modifications that improve the efficiency and precision of the NN tracking algorithm. First, instead of building a single lookup table, we build a sequence of k tables, each containing parameters describing progressively smaller warps. We use each table in turn to compute a parameter update in the scheme above, each using the partially aligned image that takes into account updates computed using earlier tables. Second, the above scheme chooses its updates from a finite, albeit potentially large, set of candidates; rather than including many small warps to get high precision, it is more efficient to perform a final alignment by minimizing the SSD as in the related works. We found that running a fixed number of iterations of the inverse compositional algorithm provided very high precision. Finally, there are no known algorithms for computing nearest neighbours in high dimensional Euclidian spaces more efficiently than exhaustively checking each example. There are, however, very efficient algorithms for finding approximately nearest neighbours. We found that using the randomized kD-tree algorithm of Silpa-Anan and Hartley [15] provided a substantial speedup while not deteriorating tracking performance. We used the implementation provided by the excellent flann library of Muja and Lowe [16]. In fact, one of the major benefits of this approach is the ability to use the existing highly optimized approximate nearest neighbour algorithms, instead of being required to produce similarly optimized code. These modifications are not present in the pseudo-code because they add clutter and are relatively easy to implement.

A. Representative Per-Frame Updates

The set of representative per-frame updates $\{\theta_1, \ldots, \theta_N\}$ should be chosen so that the optimal updates $\Delta\theta_t^*$ are always well approximated by some θ_i. On the other hand, we need to keep N relatively small so that the nearest neighbour can be found in real time. We found that the following heuristic approach worked well in a variety of application domains: Given parameters σ_t and σ_s, let $A \in \mathbb{R}^2$ be a random vector sampled from a normal distribution with mean 0 and covariance matrix diag $\left(\sigma_t^2, \sigma_t^2\right)$ and $B_1, \ldots, B_4 \in \mathbb{R}^2$ be i.i.d. random vectors sampled from a normal distribution with mean 0 and covariance matrix diag $\left(\sigma_d^2, \sigma_d^2\right)$. Let $p_1, \ldots, p_4$ be the corners of the centered unit square $[-0.5, 0.5]^2$ and let Θ be the unique parameters such that $w_\Theta(p_i) = p_i + A + B_i$ for $i = 1, \ldots, 4$. We take $\theta_1, \ldots, \theta_N$ to be i.i.d. samples of Θ. The parameters σ_t and σ_d control the amount of translation and distortion, respectively, captured by the $\theta_1, \ldots, \theta_N$.

B. Reversed Sum of Conditional Variances

The Sum of Conditional Variances is an image-patch dissimilarity measure that is robust in the presence of global illumination variations. The main idea is to estimate the probability P_{ij} that, at any given point on the object, intensities i and j co-occur in the frames I_0 and I_t respectively. Given that we can estimate this distribution at time t, we can replace the sum of squared differences with the following objective:

$$O\left(\hat{\theta}\right) = \sum_{\mathbf{x} \in R} \left(\hat{I}_0 \circ w_{\theta_0^*}(\mathbf{x}) - I_t \circ w_{\hat{\theta}}(\mathbf{x})\right)^2$$

where $\hat{I}_0$ is defined by

$$\hat{I}_0(\mathbf{x}) = \mathbb{E}\left[C \mid T = I_0(\mathbf{x})\right]$$

and (T, C) are random intensities with joint distribution P_{ij}. This can be seen as taking all of the intensities present in I_0 and mapping them to our best guess of what they should be in the current image. As it is formulated, this dissimilarity measure is difficult to use with the nearest neighbour algorithm because we store many sampled image vectors generated from I_0 and performing the intensity replacement in all of them is

time consuming. Instead, we propose a very similar *reversed* sum of conditional variances:

$$O\left(\hat{\theta}\right) = \sum_{\mathbf{x}\in R}\left(I_0 \circ w_{\theta_0^*}\left(\mathbf{x}\right) - \hat{I}_t \circ w_{\hat{\theta}}\left(\mathbf{x}\right)\right)^2$$

where

$$\hat{I}_t\left(\mathbf{x}\right) = \mathbb{E}\left[T \mid C = I_t\left(x\right)\right].$$

This is the same as the sum of conditional variances, except the intensity replacement is performed in I_t instead of I_0. To use this dissimilarity measure with the N.N. algorithm, we only need to estimate P_{ij} and replace I_t with $\hat{I}_t$. See [12] for more information on the estimation of P_{ij} and a complete algorithm.

C. Available Implementations

A python implementation of the nearest neighbour algorithm as well as a ROS package for quick and easy use in robotics applications are both available at http://webdocs.cs.ualberta.ca/~vis/nntracker/.

V. Experimental Evaluation

In the following experiments, we compare the NN tracking algorithm with the IC algorithm of Baker and Matthews and the ESM algorithm of Benhimane and Malis referred to as NN+IC, IC, and ESM respectively. These algorithms were selected for comparison because they are the standard algorithms in the literature when the appearance model is taken to be a single template image. Our own implementations for all three algorithms were used.

A. Static Image Motion

Tracking failures can generally be attributed either a failure in the appearance model or a failure in the parameter estimation method. This experiment stress-tests the parameter estimation method while ensuring that the single-template appearance model does not fail; i.e. the target object appears in each frame exactly as a warped version of the template image.

We set I_0 to be the famous Lenna image, R consists of points placed in a uniform grid within the centered unit square $[-0.5, 0.5]^2$ (the resolution of the grid is algorithm-dependent), and θ_0^* describes the warp that maps R onto the green square shown on the left of figure 2. I_1 is generated randomly as follows: let $\sigma > 0$, $p_1, \ldots, p_4$ be the corners of the green square shown on the left in figure 2, and $B_1, \ldots, B_4 \in \mathbb{R}^2$ be i.i.d random vectors sampled from a normal distribution with mean 0 and covariance matrix $\mathrm{diag}\left(\sigma^2, \sigma^2\right)$. Then, let $I_1 = I_0 \circ w_{\Theta^{-1}}$ where Θ is such that $w_\Theta\left(p_i\right) = p_i + B_i$ for $i = 1, \ldots, 4$. Given I_0, R, θ_0^* and I_1, the algorithm attempts to estimate the target's new position in I_1. We say that the an algorithm converges if the root mean squared error between the algorithm's predicted target corners and the target's true corners in I_1 is no more than 1; that is, when

$$\sqrt{\frac{1}{4}\sum_{i=1}^{4}\left\|p_i + B_i - w_{\hat{\theta}}\left(p_i\right)\right\|_2^2} \leq 1$$

Figure 2. A typical example of I_0 (left) and I_1 (right) for the static-image experiment. The target region is marked in green.

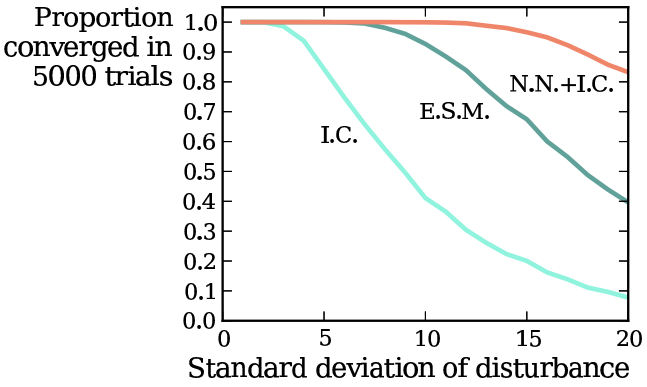

Figure 3. Proportion out of 5000 trials that each of NN+IC, IC, and ESM converged for various values of σ. See section 5.1 for details.

where $\hat{\theta}$ is the algorithm's estimate of the warp parameters. See figure 2 for an example of I_0 and I_1 with the target region marked in green. For σ values ranging from 1 to 20, we ran each algorithm on 5000 2-frame sequences (I_0, I_1); figure 3 shows the proportion of trials on which each algorithm converged.

The NN+IC algorithm used a set R with 50×50 resolution, and $k = 3$ lookup tables each containing 2000 example parameters generated with $(\sigma_d, \sigma_t) = (0.06, 0.04), (0.03, 0.02), (0.015, 0.01)$, respectively. The IC algorithm used a set R with 100×100 resolution and ran for a maximum of 30 iterations. The ESM algorithm used a set R with resolution 50×50 and ran for a maximum of 30 iterations. These parameters were chosen to maximize performance while giving each algorithm approximately the same running time. Running the entire experiment took 49 minutes for the IC algorithm, 55 minutes for the ESM algorithm, and 33 minutes for the NN+IC algorithm.

Despite being faster, the NN+IC algorithm was able to reliably recover the motion for larger σ values than either of the other two algorithms. We also see that the NN+IC algorithm substantially outperforms the IC algorithm, which gives us confidence that the final IC iterations used to perform the fine alignment aren't doing all of the work.

N.N.+I.C.	Angle	Range	Fast Far	Fast Close	Illumination
bump	77.8	57.9	58.5	25.1	61.1
stop	100.0	86.6	59.9	50.6	67.6
lucent	87.3	98.2	47.9	51.5	67.1
macmini	52.9	46.1	12.2	15.7	17.4
isetta	99.4	82.3	74.2	17.2	64.1
philadephia	99.7	95.5	49.8	75.0	63.4
grass	19.8	4.2	6.6	5.8	13.2
wall	75.0	79.8	56.8	22.8	56.4

E.S.M.	Angle	Range	Fast Far	Fast Close	Illumination
bump	77.4	90.9	50.7	40.8	96.3
stop	100.0	96.2	24.5	50.5	54.9
lucent	35.3	23.9	12.0	16.8	89.6
macmini	24.0	9.3	6.8	37.8	9.2
isetta	83.2	89.0	17.2	28.0	99.9
philadephia	82.1	72.3	14.5	71.9	100.0
grass	13.8	6.6	6.9	7.8	17.4
wall	47.4	34.3	8.3	28.4	96.9

I.C.	Angle	Range	Fast Far	Fast Close	Illumination
bump	77.9	79.1	13.4	30.4	95.2
stop	66.7	47.5	12.1	24.7	42.7
lucent	13.7	20.9	5.4	12.1	19.5
macmini	17.7	5.7	4.5	22.0	8.5
isetta	62.2	32.8	6.0	16.5	91.8
philadephia	61.2	31.2	6.5	46.6	96.7
grass	9.2	6.6	5.6	3.7	10.7
wall	34.8	24.0	7.9	12.2	35.3

Figure 4. Percentage of frames tracked in each benchmark videos for each of the NN+IC, IC, and ESM algorithms.

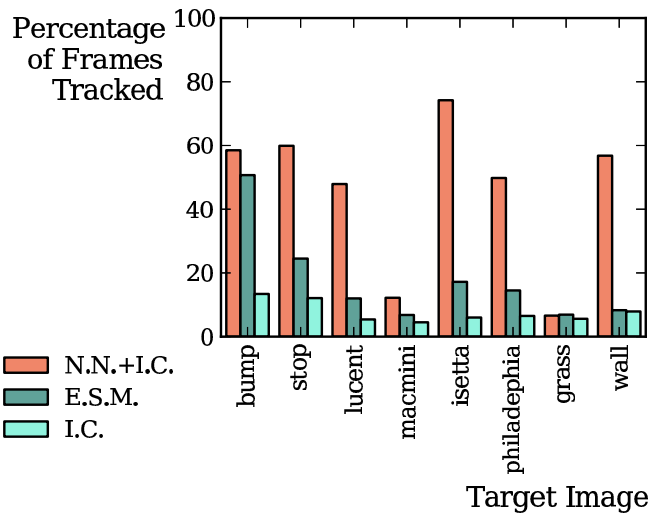

Figure 5. Percentage of frames tracked in the "fast far" video sequences.

Figure 6. Experimental setup for a visual servoing experiment using the WAM arm with a wrist-mounted camera. Patches of a highly textured surface in the camera's field of view were tracked and used as input to a visual servoing routine in order to precisely align the camera position. The algorithms were compared based on the speed and reliability of the overall system.

B. Metaio Benchmark

Lieberknecht et al. [17] provide a set of benchmark videos designed specifically for comparing 2D tracking algorithms. The videos feature 8 different target objects in videos exemplifying each of the following: angle, range, fast far, fast close, and illumination. The ground truth parameters θ_t^* are known for every frame of all videos, but only the ground truth for every 250^{th} frame is packaged with the benchmark. The authors offer to evaluate a sequence of estimated parameters $\hat{\theta}_0, \hat{\theta}_1, \ldots$ for each of the benchmark videos, reporting the percentage of frames that the estimated parameters were sufficiently close to the ground truth.

We ran our implementations of the NN+IC, IC, and ESM algorithms on the benchmark videos; results are shown in 4. In the "fast far" sequences, NN+IC often substantially outperforms the other methods, as shown in 5. This agrees with the results of the static-image experiment, in that the NN+IC algorithm is accurately estimating larger parameter updates than the other algorithms. Many of the failures of the NN+IC algorithm in the other videos appear to be caused by the target object partially leaving the frame or becoming motion blurred. The other algorithms were not as affected by these situations.

The algorithm parameter settings for this experiment were identical to those in the static-image experiment.

C. Application to Visual Servoing

In this experiment, we compare the three algorithms in a robotic control setting. There is a camera mounted on the wrist of a 4-DOF WAM robotic arm (see 6) and the goal is to move the arm such that the camera is in a desired pose. This is achieved by aligning the corners of a tracked rectangular patch with a goal configuration. In these experiments, we did not have a model of the arm or calibration information relating the pose of the camera to the arm or the arm to the environment.

Each configuration of the arm's joint angles determine the pixel-locations of the four corners of a rectangular tracked patch. Let $v : \mathbb{R}^4 \to \mathbb{R}^8$ be the so-called motor-visual function that maps a vector of four joint angles to a vector consisting of the four corner pixel-locations concatenated into a single vector. Let $s^* \in \mathbb{R}^8$ denote the goal corner locations, $q^* \in \mathbb{R}^4$ denote the unknown corresponding joint angles, s_c denote the current corner locations, and q_c denote the arm's current joint angles. The WAM arm has a built-in controller to set motor torques in order to achieve a given vector of joint angles. Our control algorithm updates the desired joint angles, denoted by q_d, at each time according to $q_d = q_c + \lambda J^+ (s^* - v(q_c))$ where J is the Jacobian of the motor visual function $v(q)$ at $q = q^*$. The parameter λ is called the gain and determines

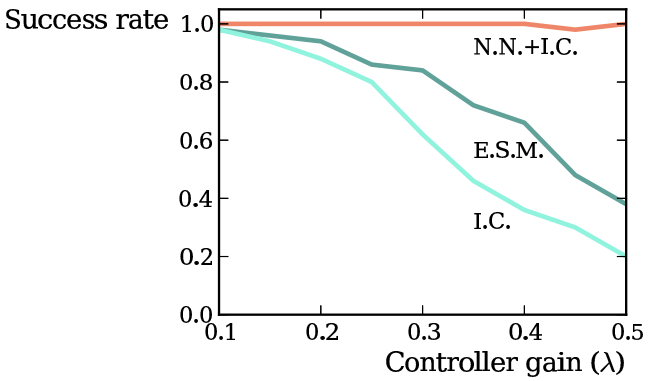

Figure 7. The percentage of successful trials for each each algorithm for various gains λ in the visual servoing experiment.

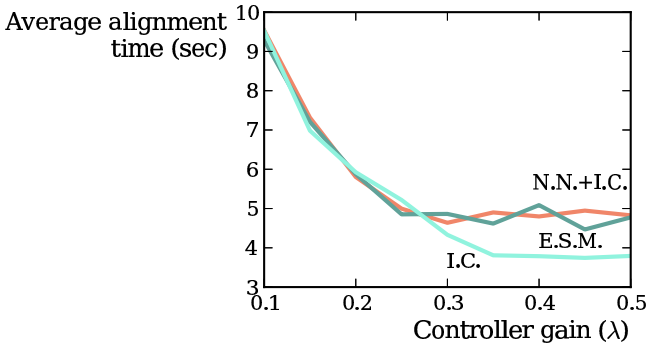

Figure 8. The average alignment time taken on successful trials for each algorithm for various gains λ in the visual servoing experiment.

how quickly the arm moves.

We generated 50 goal joint positions q^* and starting positions q_s. An offline procedure was used to compute the Jacobians of the motor-visual function at each q^* and the vector of corners s_s used to initialize the tracker when the arm is in it's initial position q_s. A single run of the experiment consists of moving the arm to position q_s, initializing the tracker with the quadrilateral region bounded by corners s_s, and then updating the desired joint angles until either $\|s_c - s^*\| < 10$ or the controller began to diverge. We ran the experiment with each algorithm and with different gains in the range 0.1 to 0.5. We record the proportion of trials that were successful and the average time taken for each successful alignment, shown in figure 7 and figure 8, respectively.

This task in this experiment is more challenging than the others because the tracker performance influences the incoming video sequence. If the algorithm does not compute its per-frame updates efficiently the frequency of the control loop drops and causes sometimes violent motion of the arm which in turn produces a challenging video sequence.

In all but one trial, the NN+IC algorithm tracked successfully and efficiently enough for the control algorithm to converge quickly and smoothly. The IC algorithm could not track with higher gains because the target was moving too quickly. With higher gains, ESM was often able to track the early motions, but at a slow frame rate, and the tracking failed in the consequently jerky camera motion. In this setting, the reliability of the visual servoing routine was greatly improved by using the NN+IC algorithm. In the average time plot, the I.C. algorithm appears to be doing well for very high gains. This is due to the fact that the average time is only taken over successful trials and, for high gains, the I.C. algorithm was only successful for trials with very short motions.

We tried several parameter settings for each algorithm, but only display the results for the most successful settings. For ESM we used a resolution of 30×30 and a limit of 15 iterations, for IC we used a 100×100 patch and a limit of 30 iterations, and for the NN+IC algorithm we used the same parameter settings as in the previous experiments, except with 30×30 resolution.

D. Case Studies

In addition to the above experiments, we ran each of the three algorithms on video sequences captured in different domains.

The first sequence we show is the video captured during the descent of the Mars Curiosity Rover (images publicly available at http://mars.jpl.nasa.gov/msl/multimedia/raw/?s=0&camera=MARDI). This video sequence was captured at only 4 frames per second, so the per-frame camera motions are quite large. We tried all three algorithms on several patches throughout the video, but only the N.N.+I.C. algorithm was able to track for more than 2 or 3 frames. Example frames from the video sequence are shown in figure 9.

For a more typical every-day domain, we recorded a video of pouring cereal into a bowl. Human manipulation of objects actually presents a difficult application domain because our motions tend to be abrupt and unconstrained. Like in the Mars descent video, only the N.N.+I.C. algorithm was able to track for a substantial portion of the video. Example frames are shown in figure 10.

Finally, we present a sequence captured from a small unmanned aerial vehicle (U.A.V.). In this experiment we only compare the N.N. algorithm with the I.C. algorithm (referred to as "SSD" in the video). Additionally, in these video sequences the warping function w is taken to be a parameterization of the 4-DOF similarity transformations that only allow for translations, rotations, and scaling. As in the previous examples, the N.N. algorithm is more reliable.

VI. Conclusions

We have introduced the nearest neighbour registration-based tracking algorithm that uses an approximate nearest neighbour search to estimate the per-frame homography parameter updates. We empirically showed that the algorithm is more efficient and capable of accurately estimating updates corresponding to large motions than the standard algorithms with the same appearance model. Finally, we showed that the nearest neighbour algorithm was more robust than existing methods in a robotic visual alignment task.

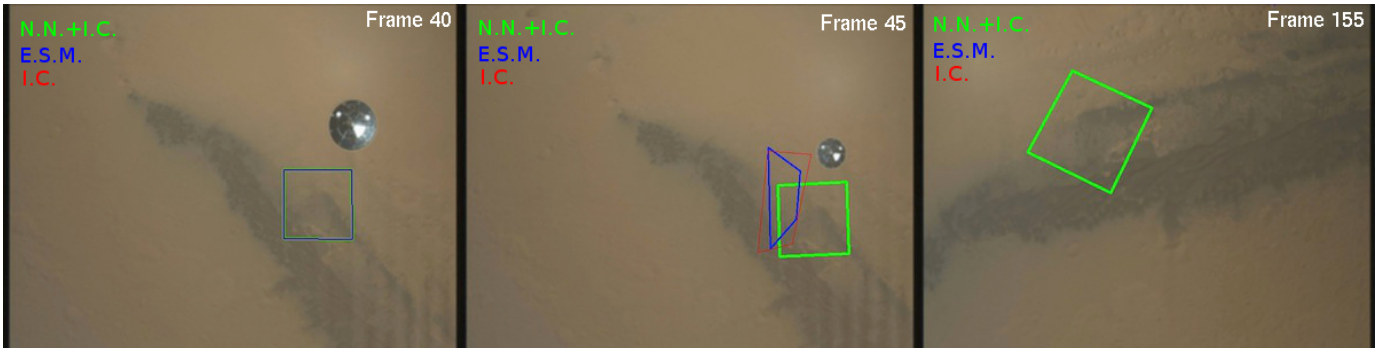

Figure 9. Each of the visual tracking algorithms was run on the video sequence captured by the Mars Curiosity rover during its descent. The shown sampled frames show that our algorithm was able to track for more than 100 frames, while both ESM and IC failed after fewer than 5.

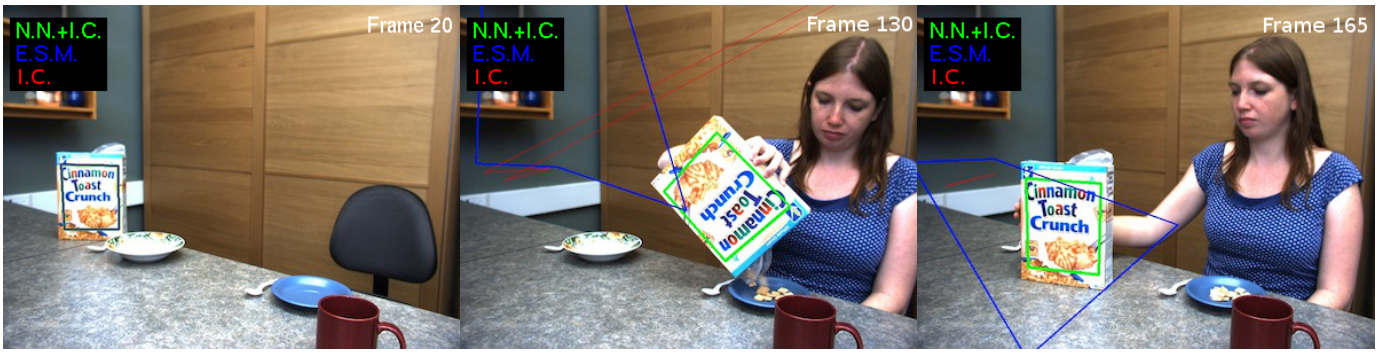

Figure 10. Each of the visual tracking algorithms was run on a video sequence depicting a bowl of cereal being poured. This video is interesting because human motion is less constrained than in the other case studied. Again, our algorithm is able to track for substantially more frames than either ESM or IC.

Figure 11. This case study compares the our algorithm, labeled by "NN" in red, with the IC algorithm, labeled by "SSD" in green, on a video capture by a small UAV.

References

[1] S. Baker and I. Matthews, "Equivalence and efficiency of image alignment algorithms," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–1090–I–1097, 2001.

[2] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," *Proceedings of the International Conference on Intelligent Robots and Systems*, vol. 1, pp. 943–948, 2004.

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

[4] T. Kanade, "an iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981, pp. 674–679.

[5] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[6] D. Ross, J. Lim, R. Lin, and M. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 125–141, 2008.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

[8] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Proceedings of the European Conference on Computer Vision*, pp. 430–443, 2006.

[9] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, vol. 2, no. 2, pp. 1–15, 1998.

[10] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, vol. 2, no. 142–149, 2000.

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

[12] R. Richa, R. Sznitman, R. Taylor, and G. Hager, "Visual tracking using the sum of conditional variance," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2953–2958, 2011.

[13] M. Hasan, M. Pickering, A. Robles-Kelly, J. Zhou, and X. Jia, "Registration of hyperspectral and trichromatic images via cross cumulative residual entropy maximisation," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 2329–2332, 2010.

[14] A. Dame and E. Marchand, "Accurate real-time tracking using mutual

information," *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 47–56, 2010.

[15] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[16] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *Proceedings of the VISAPP International Conference on Computer Vision Theory and Applications*, pp. 331–340, 2009.

[17] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A dataset and evaluation methodology for template-based tracking algorithms," *IEEE International Symposium on Mixed and Augmented Reality*, pp. 145–151, 2009.