Global Visual-Motor Estimation for Uncalibrated Visual Servoing

Amir massoud Farahmand, Azad Shademan, and Martin Jägersand

Abstract— **In this paper, we present two methods for the estimation of a globally valid visual-motor model of a robotic manipulator. In conventional uncalibrated visual servoing, the visuo-motor function is approximated locally with a Jacobian. However, for optimal task planning, or nonlinear controller design with global stability guarantee, one needs to know a model that provides some information about the behavior of the system over the whole workspace. Our presented methods remedy this drawback in uncalibrated visual servoing by incrementally building a global estimator based on the movement history. We implement two such methods. The first method is a K-nearest neighborhood regressor over Jacobian that uses previously estimated local models. The second method stores previous movements and computes an estimate of the Jacobian by solving a local least squares problem. Experimental results show that both methods provide better global estimation quality compared to the conventional local estimation method with much lower estimation variance.**

I. INTRODUCTION

Having a globally valid visual-motor model of a robot is an important component for designing a globally stabilizing visual-servoing controller or planning an optimal trajectory for visually-guided tasks (See [1] for a tutorial on visual-servoing). In many cases, the visual-motor model of the robot is not available analytically (or it is burdensome to find one). Examples include a mobile manipulator, a case that cameras are moving or placed arbitrarily by hand. Or in a situation where the control objective is not defined by the end-effector plate of the robot, but involves picking up and manipulating other objects where the desired visual alignments are defined with respect to those objects. In practical applications often neither object geometry, nor the exact grasp with which it is picked up is precisely known. Several researchers have developed methods for estimating a visual-motor model (i.e. Jacobian) online [2][3][4][5]. However, most of these research have been focused on estimating only locally valid models, and keeping only one such at a time. Although local models can be used to design locally stabilizing controllers, those models are not natural for global planning and controlling approaches which usually need the global knowledge of the system's model.

Visual servoing has been extensively studied with different control schemes such as position-based controllers [6], image-based controllers ([7], [8]), or hybrid controllers [9]. These schemes can be either model-based (calibrated), where the robot or the camera are calibrated, or model-free (uncalibrated), where the model of the system is unknown. Model-based visual servoing can be considered a solved problem for local tasks with non-redundant manipulators. However, there are still theoretical and practical implications for global visual servoing due to nonlinear model estimation that make visual servoing challenging.

The main contribution of this paper is introducing two methods to globally estimate the visual-motor Jacobian. Both methods utilize the available information more efficiently. In contrast to usual local estimation methods that discard far data-points (see Section II), our presented methods adopt them in an effective way (Section III). Our first method is an extension to the previous approaches that use Broyden methods. It uses a K-Nearest Neighborhood (K-NN) regressor to combine previously estimated Jacobians. In this method, we store local estimation of Jacobians and re-use them to estimate the Jacobian for a new point (Section III-A). The other method uses data-points directly, solves a locally-defined least squares problem, and finds the best hyper-plane that fits those data (Section III-B).

II. LOCAL METHODS

Local estimation of the visual-motor Jacobian without explicit knowledge of the kinematic structure or the camera-robot configuration for uncalibrated visual servoing has been studied in the past. Hosoda and Asada [2] modify and extend the weighted recursive least square (RLS) method to estimate the Jacobian in real-time. They use a forgetting factor to decay the influence of old data during the estimation process. This helps the method to keep track of the time-varying linear model which is caused by the joint variables' changes and the nonlinearity of the model. A similar approach to recursively update the Jacobian, adapted from Broyden's method in optimization was presented by Jägersand *et al.* [3]. They also add a trust region method, taking into account the comparatively slow visual updates in typical visual servoing systems.

The local estimation of the Jacobian can deteriorate if the motion of the robot is along a straight path in joint space or in a singular configuration of image features. Sutanto *et al.* study this problem and propose an exploratory motion to ensure that the robot moves in the right direction, where the numerical estimation of the Jacobian is stable [4]. These papers assume a static goal. Piepmier *et al.* introduce the quasi-Newton Jacobian estimation for the uncalibrated visual servoing of a moving target [5].

In the remainder of this section, we formulate the visual-motor Jacobian estimation problem and review the local

Amir massoud Farahmand, Azad Shademan, and Martin Jägersand ({amir, azad, jag}@cs.ualberta.ca) are with the department of Computing Science, University of Alberta, Edmonton, Alberta, Canada.

Amir massoud Farahmand acknowledges the supported of Alberta Ingenuity Centre for Machine Learning (AICML). Azad Shademan is an Alberta Ingenuity Student and acknowledges the support of Alberta Ingenuity.

estimation method in more detail. We also specify our control method for visual servoing.

A. Problem Formulation

Consider a robotic manipulator that consists of a series of revolute and prismatic joints characterized by joint angles or displacement $q \in \mathbb{R}^n$ and a vision system that observes the environment. This vision system can be attached to the end-effector (eye-in-hand) or can observe the robot from a fixed position (eye-to-hand). By changing joint variables, the robot moves and the image projected on each camera changes. Defining some feature points on camera images, we can relate the position of the joint variables to the position of those feature points. If we denote $x_i \in \mathbb{R}$ as the value of one of the visual features, there is a relation between q and x such as $x_i = f_i(q)$. Depending on the way we define the feature, f_i depends on the kinematic model of the robot and the camera model. Note that this function is different for image-based or position-based visual servoing, but the essence of it -which relates feature variables to the joint variables- is the same. The goal of this section is to derive a method to locally estimate this model.

Generalizing the previous formalism to several features, say m, the feature vector $X \in \mathbb{R}^m$ and visual-motor function $F(q)$ are defined as follows

$$X := [x_1; x_2; ...; x_m]_{m\times 1} \tag{1}$$

$$F := [f_1(q); f_2(q); ...; f_m(q)]_{m\times 1} \tag{2}$$

$$X = F(q). \tag{3}$$

Let $q(t)$ be a function of time; the dynamics of feature points are

$$\frac{dX}{dt} = J(q)\frac{dq}{dt}, \tag{4}$$

where $J(q) = \partial F(q)/\partial q$ is the visual-motor Jacobian. To perform image-based tasks like regulating feature points to a specified position, one can do dynamic programming, use local controllers or design a complicated nonlinear controller to change $q(t)$ in such a way that the task would be achieved. In general, the visual-motor function F is not known for an arbitrary robot-camera system. One approach to find the Jacobian is to calibrate the parameters and analytically derive the Jacobian. Another approach that we pursue here is to approximate the visual-motor Jacobian matrix J. We do not assume any *a priori* knowledge about the robot or camera.

B. Estimation

Let $J_i \in \mathbb{R}^n$ be the i-th row of the visual-motor Jacobian J. We can formulate the estimation problem as the following least squares optimization problem

$$\hat{J}_i = \arg\min_{\hat{J}\in\mathbb{R}^n} \int_{t=0}^{\infty} \left(\frac{dx_i}{dt} - \hat{J}\frac{dq}{dt}\right)^2 dt \tag{5}$$

$$\approx \arg\min_{\hat{J}\in\mathbb{R}^n} \sum_{l=1}^{\infty} \left(\Delta x_i(l) - \hat{J}\Delta q(l)\right)^2 \tag{6}$$

where $\Delta x_i(l) = x_i(l) - x_i(l-1)$ and $\Delta q(l) = q(l) - q(l-1)$ for the discrete-time counterpart. However, F is a nonlinear function of q, so $J(q(t))$ is a time-varying quantity. If we simply solve the previous least square problem, the estimated $\hat{J}_i$ will not be a good approximation of $J_i(t)$ because it is minimizing the error over the whole experience trajectory. In order to remedy this problem, we can force the optimizer to pay more attention to errors in recent history by introducing a forgetting (or discounting) factor $0 < \lambda \leq 1$ to our least squares problem:

$$\hat{J}_i(t) \approx \arg\min_{\hat{J}\in\mathbb{R}^n} \sum_{l=1}^{\infty} \lambda^{t-l} \left(\Delta x_i(l) - \hat{J}\Delta q(l)\right)^2. \tag{7}$$

When $\lambda = 1$, this is equivalent to the previous optimization problem. We can solve this optimization problem using a modified version of Recursive Least Squares (RLS) method [10]. The method is defined by the following recursive equations for estimating each $\hat{J}_i$:

$$\hat{J}_i(t) = \hat{J}_i(t-1) + K_i(t)\left(\Delta x_i(t) - \hat{J}_i(t-1)\Delta q(t)\right) \tag{8}$$

$$K_i(t) = P_i(t-1)\Delta q(t)\left(\lambda I + \Delta q(t)^T P_i(t-1)\Delta q(t)^T\right)^{-1} \tag{9}$$

$$P_i(t) = (I - K_i(t)\Delta q(t)^T)P_i(t-1)/\lambda. \tag{10}$$

P_i is usually initialized as a sufficiently large positive definite matrix, e.g. $\Lambda \times I$ where Λ is a large number and I is the identity matrix. There are other online parameter estimation methods such as *Broyden*, *Projection Algorithm* or *Least Mean Square (LMS)* algorithms that can be used similarly (See [10] and [3]).

C. Control

Based on our estimation, we can take different approaches for designing a controller. Here, we review the linear control approach. Let the goal of the control be to regulate features X to a desired X^* (which can be function of time, i.e. a path). From (4), the dynamics of the error system follows:

$$e = X - X^* \Rightarrow \dot{e} = \frac{\partial F(q)}{\partial q}\dot{q} = J(q)e. \tag{11}$$

Assuming that $\dot{q} = Ke$, a linear controller, we have

$$\dot{e_i} = J(q)Ke_i. \tag{12}$$

The stability of this linearized system is not directly related to the stability of the original system in (11). However, we can expect that if this linearized model is stable and slowly-varying, the original system will also be stable. In order to make Equation (12) stable, we need to choose K such that JK becomes Negative Definite (N.D.). One choice is *Inverse Plant* controller where $K = -J^{-1}$ or $K = -J^{\dagger}$ for a more general case[1]. We can also assign eigenvalues arbitrarily using Singular Value Decomposition (SVD). First decompose $J = U\Sigma V^T$. We can write

$$JK = (U\Sigma V^T)(VDU^T) = U(\Sigma D)U^T = U\Lambda^* U^T. \tag{13}$$

in which $\Lambda^* = \Sigma D$. Since, U is unitary, singular values of $U\Lambda^* U^T$ is equal to singular values of Λ^*. Therefore, by

[1] $A^{\dagger}$ is the pseudo-inverse of A.

setting singular values of Λ^*, we can set singular values of $U\Lambda^* U^T$ and JK - which is our desire. By choosing $D = \Sigma^\dagger \Lambda$ and constructing K as $K = -VDU^T$, we can achieve our goal. Note that in all these controllers, K depends on $J(q(t))$ and so on $q(t)$.

III. GLOBAL ESTIMATION METHODS

The main drawback of local estimation methods is their local validity. For a nonlinear system such as a robot, it is difficult to design a controller to make the system globally stable without knowing the global model of the system. Even if one can design such a controller (e.g. by slowing-down the system and using conservative controllers), the problem of planning remains unsolved. For finding an optimal path (e.g. one that minimizes the control effort for reaching a far point, or a one that is smooth, or a one that avoids obstacles, the controller needs to know a global model of the system. This is the main motivation for developing two new estimation methods that are introduced in this section.

A. K-NN Regression-based Method

This method *combines* estimates obtained by a local estimation method (such as the RLS(λ) we mentioned in Section II) to give an estimation of the Jacobian $J(q)$ for a not-previously-seen point q. In a local method, we throw away all previous estimations and only rely on the most recent one. This works when you just want to locally control your robot, but it does not help you to do a global planning in which you need Jacobian estimation for other points in joint space too.

We use a K-Nearest Neighborhood (K-NN) regression method to combine previous local estimates. Assume that we have a database of estimated $\{\hat{J}^{(i)}(q^{(i)})\}$ $(i = 1, ..., n)$ in which $q^{(i)}$ indicates the joint configuration q at the position where the estimate $\hat{J}^{(i)}(q^{(i)})$ is added to the database (we will discuss how these estimates are added to the dataset soon), and n is the number of estimates in the database. To estimate the Jacobian at a new point q (which is not necessarily the same as one of $\{q^{(i)}\}$, the K-NN regressor first sorts[2] estimates $\hat{J}^{(i)}(q^{(i)})$ $(i = 1, ..., n)$ based on the distance of their $q^{(i)}$ to q (this can be an Euclidean distance). Let's denote this set as $\{\hat{J}_{(i,n)}(q_{(i,n)})\}$ where $q_{(1,n)}$ is the closest point to q. Afterward, it computes $\hat{J}^{KNN}(q)$ (the global estimate of the Jacobian at q) in the following way:

$$\hat{J}^{KNN}(q) = \frac{1}{K} \sum_{i=1}^{K} \hat{J}_{(i,n)} \left(q_{(i,n)} \right). \tag{14}$$

In order to add a new estimate $\hat{J}^{(i)}(q^{(i)})$ to the database, it compares the current Jacobian estimate of the local estimator at q (which is $\hat{J}(q(t))$) with the global estimation $\hat{J}_{KNN}(q)$. To compare two Jacobians, we need to choose a matrix norm. Here, we use L_2 induced matrix norm:

$$dist(J_1, J_2) = \|J_1 - J_2\|_2 = \sup_q \frac{\|(J_1 - J_2)\Delta q\|_2}{\|\Delta q\|_2} \tag{15}$$

[2]In practice, there is no need for actually performing a sorting algorithm if one uses an elegant data-structure for storing data. Here we mention sorting in order to clarify the procedure.

If the difference between $\hat{J}(q(t))$ and $\hat{J}_{KNN}(q(t))$ is greater than some threshold ϵ, the current Jacobian estimation, $\hat{J}(q(t))$, will be added to the database. If it is not, it means that the global estimator approximates the current local estimates rather well, and there is no need to add any new data point.

Initially, the estimation database is empty. The robot starts moving from some point in the joint space and follows path $q(t)$. Meanwhile, the local estimator (e.g. RLS(λ)) approximates $J(q(t)$ by $\hat{J}(q(t))$. The global estimator compares this matrix to its global estimation. Because the initial database is empty, we need to define a corresponding global Jacobian in order to compare distances. We can simply define a zero matrix as the output of the global estimator in the empty database case. The distance between a matrix $\hat{J}(q(t_1))$ to zero matrix is large enough; therefore, it adds the current local estimate $\hat{J}(q(t_1))$ to the database. In the close vicinity of $q(t_1)$, the error between two estimations are small, and no point will be added. However, when the distance between $q(t_1)$ and the current joint values exceeds a certain amount (which depends on ϵ and the Lipschitz constant of $F(q)$), the error will be more than the threshold and a new point will be added. This process continues.

There are at least two important questions about this method. One is that whether we can use the convergence result of standard K-NN regression, where there is an *independent identically distribution (i.i.d.)* assumption about the source of data, for this problem [11]. However, samples that come from a moving robot are dependent with each other, and those convergence results (which relates the approximation accuracy to the number of samples) cannot not readily be applied to this situation. The other problem is that local estimates of Jacobian, $\hat{J}(q)$, which we use as a reference value, may be a biased estimation for the true Jacobian $J(q)$ at that point. This issue will be investigated in Section IV.

B. Local Least Squares (LLS) Method

Our second proposed method for globally estimating a Jacobian is directly fitting the best hyper-plane to the data points around the point q under consideration. Because the Jacobian is the tangent plane to the visual-motor function, finding the Jacobian is simply equivalent to fitting the best plane at given point q.

Data points for this method are all previous $q(t)$ and $X(t)$ in the history of the system. Therefore, for this approach one should keep the history of all movements the robot has made. In contrast to the method introduced in Section III-A, we do not need to store local estimates of Jacobian. Instead, we calculate the Jacobian whenever it is needed.

To be more precise, suppose that we want to estimate $J_i = \frac{\partial f_i(q)}{\partial q}$, which we can approximately write as $\Delta x_i \simeq \frac{\partial f_i(q)}{\partial q} \Delta q$. Denoting the set of all previous features and joint parameters as $\{x_i(l)\}$ and $\{q(l)\}$ for $l = 1, \cdots, t$, respectively - in which t is the current time. We define the

following optimization problem:

$$\hat{J}_i^{LLS}(q,\epsilon) = \arg\min_{\hat{J}\in\mathbb{R}^n} \sum_{r=1}^{t}\sum_{s=1}^{t} \left(\Delta x_i^{[r,s]} - J\Delta q^{[r,s]}\right)^2 \times I\left(\|q(r)-q\|<\epsilon\right) \times I\left(\|q(s)-q\|<\epsilon\right) \quad (16)$$

in which $\Delta x_i^{[r,s]} := x_i(r) - x_i(s)$, $\Delta q^{[r,s]} := q(r) - q(s)$, and $I\{P\}$ is the indicator function, i.e. it is one if P is true and zero otherwise[3].

This optimization problem fits the best hyper-plane $\hat{J}$ to a local neighborhood of q. If this neighborhood is small enough while the number of samples in it is large (so the total number of samples should be very large), we can expect that the result of this optimization problem be close to $J_i = \frac{\partial f_i(q)}{\partial q}$.

We can also define a similar optimization problem based on the K-nearest neighborhood of q. Changing K changes the radius of ball containing the K-nearest points. The exact relation between K and the radius depends on the geometry of the space and the way data points are generated. Roughly speaking, if we have t samples and the space is effectively d-dimensional, $\frac{K}{t} = \Theta(\epsilon^d)$.

In our implementation, we use a K-nearest neighborhood scheme, and the following optimization problem:

$$\hat{J}_i^{LLS}(q,K) = \arg\min_{\hat{J}\in\mathbb{R}^n} \sum_{r=1}^{t}\sum_{s=1}^{t} \left(\Delta x_i^{[r,s]} - J\Delta q^{[r,s]}\right)^2 \times I\left(q(r)\in\{q^{(1,t)},\cdots,q^{(K,t)}\}\right) \times I\left(q(s)\in\{q^{(1,t)},\cdots,q^{(K,t)}\}\right) \quad (17)$$

in which $q^{(r,t)}$ is the r-th nearest neighborhood to q and the $x_i^{(r,t)}$ is the corresponding x_i.

If the model is time-variant, old data points may not accurately encode the model of the system. In these cases, we need a forgetting mechanism such as weighting old data or throwing them away.

It is apparent that if the robot has no previous experience in a certain region of its joint variable space, this estimate may be not so accurate. This is true for all methods that do not use any kind of a priori assumption (like the parametric class of the model or the smoothness of it).

IV. EXPERIMENTS

In this section, we study different aspects of our proposed methods and compare them with the local estimator. The experiments are performed in MATLAB using Corke's Robotics Toolbox [12] and the Epipolar Geometry Toolbox [13]. Comparisons results on measures such as the global estimation error, the effect of noise, the effect of number of neighbors, and the feasibility for visual servoing task are reported.

[3]Considering all pairs of nearby data points may be redundant, and a subset of them may be enough. We have not analyzed the subset selection effect, specially when measurements are noisy and the estimation has error too.

For all these experiments, we use the *Puma 560* model with an eye-to-hand configuration. To define our visual features, we use a stationary stereo rig setup. The visual features are the projections of the end-effector's position onto the image space of each of those cameras. Therefore, the feature space is four dimensional ($x \in \mathbb{R}^4$). The position of the end-effector has only three dimensions, therefore we used the first three joints of the *Puma* arm, $q \in \mathbb{R}^3$, and J is 4×3 matrix. By adding more feature points (e.g., observing other rigid points on the manipulator), the generalization of our proposed method to estimate the full $m \times 6$ Jacobian is straightforward.

A. Global estimation error

We define a distance measure between the true Jacobian $J(q)$ and the estimated Jacobian ($\hat{J}(q)$ for the local estimation method or $\hat{J}^{KNN}(q)$ and $\hat{J}^{LSS}(q)$ for global methods) to compare the quality of global estimation. A small distance is important for planning and controlling approaches which use global model of the system. To define that distance, we extend (15) from a point-wise distance to global distance by taking an integral of it over all possible joint values:

$$Dist(J_1,J_2) = \int_{q\in D(q)} \|J_1(q) - J_2(q)\|_2 dq \quad (18)$$

where $D(q)$ is the domain of possible joint values which depends on the specific robot under consideration. If some specific subspace of $D(q)$ is more important (e.g. the robot usually works around those places), we can use weighted version of the integral.

In order to numerically evaluate this integral (e.g. $Dist(J,\hat{J}^{KNN})$), we estimate it by a Monte Carlo method, i.e. picking a random sample and evaluating the integrand at that point, and finally averaging all those values. We use 1000 random sample points in our experiments with sample points selected from a normal distribution with zero mean and unit variance (thus, our integral is biased toward the joint values closer to the $q = 0$).

For the first set of experiments, we force the robot to follow a randomly generated piece-wise smooth trajectory in the joint space and measure the quality of global estimation. Trajectories are made by selecting random points in the joint space and smoothly connecting them. In our implementation, we use 101 random points in the joint space (with joint values selected from a normal distribution $N(0, I_{3\times 3})$) and connecting each pair of them by 50 intermediate smoothly connected points. We add some level of noise to these trajectories. Actuator noise, which exists in real-world problem anyway, improves the quality of identification as it increases the local spanning dimension of the trajectory. Without this noise even the local estimation method may fail to accurately estimate the Jacobian.

We compare $Dist(J,\hat{J}^{KNN})$, $Dist(J,\hat{J}^{LLS})$, and $Dist(J,\hat{J}(t))$ at several points during the movement of the robot in which J is the true Jacobian and $\hat{J}(t)$ is the last estimation made by the local method (t is the time we are

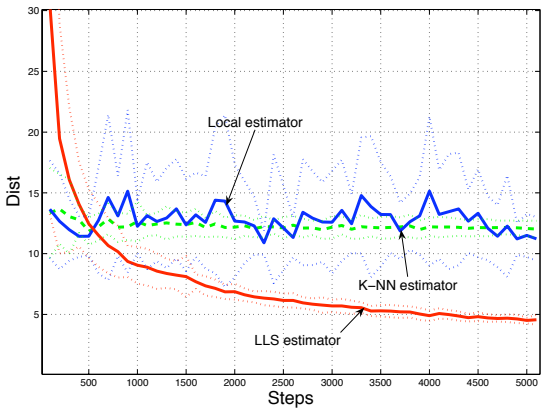

Fig. 1. Estimation error (18) for different estimation methods during the movement of the robot. Dotted lines are one standard deviation error around the average error. Blue lines are for the local estimator, green lines are for K-NN global estimator, and red lines are for local least squares global estimator.

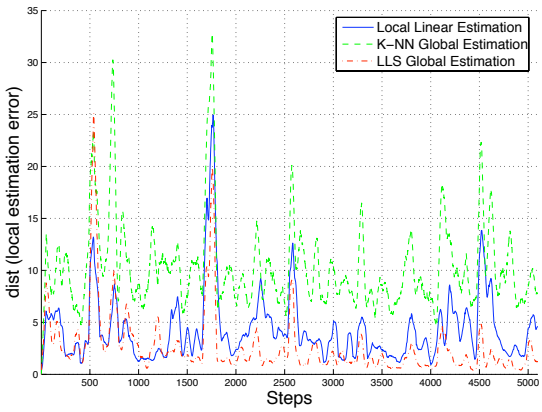

Fig. 2. Local Jacobian error (15) for different estimation methods during a sample movement of the robot. The data are smoothed by a moving average filter with the window size of 50.

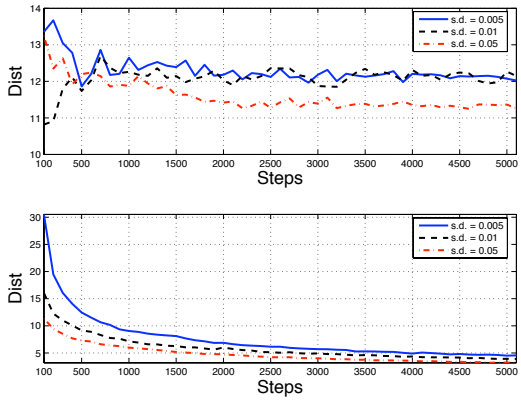

Fig. 3. The effect of noise level (measured as the standard deviation of a normal random variable) on the quality of global estimation for K-NN (top) and LLS (bottom) global estimators.

calculating these values). The result is depicted in Fig. 1[4].

This comparison shows that the performance of the LLS estimator (Section III-B) gradually increases (by decreasing (18)) while the robot moves around. The performance of the K-NN estimator (Section III-A) has an increasing behavior too, but the rate of change is much smaller comparing to the LLS method.

One may wonder why the K-NN does not perform so well. Fig. 2 suggests that this may be due to the high amount of error in each local estimates $\hat{J}^{(i)}$ which is used by K-NN method (14). This figure shows the local Jacobian error through time evolution[5]. It reveals that the local estimates that we get from RLS(λ) has a rather high amount of error and this error does not go to zero. If the underlying system was linear, we should expect the convergence to zero, but for this nonlinear system the property does not hold. One may get better results by tweaking λ, but the qualitative behavior should not be that different. Each of those estimates are possibly biased because of RLS(λ) procedure we use. Whether the main source of error is this or some other phenomenon is not clear yet.. Nevertheless, we can see that the performances of both proposed methods are better than the local method except in the early movement stages.

Another important property, which is seen in Fig. 1, is the reduced variance of global methods. This means that both methods provide a more stable estimation of the Jacobian than local method does.

[4]The data in this figure is the average result of 20 different runs, each of them consists of 5010 robot's steps (101 random points as defined previously). The number of neighbors for K-NN method is selected as $ceil(\log(steps))$, which is around 2 to 4 for this range of values, and the number of neighbors for LLS method (as defined in (17)) is fixed as 30. λ for RLS(λ) method is 0.99 for all experiments including this one. The standard deviation of the contaminating noise is 0.005. No optimization was made for setting these parameters.

[5]The error between the Jacobian at $q(t)$ at time t and the true Jacobian at the same point. Note that here we use (15) instead of (18)

B. *Effect of noise*

One important question is the effect of noise on the estimation accuracy. We know that for identifying a dynamical system, we need to excite it properly, i.e. the input signal must be persistent exciting [10]. By analogy from identification theory of linear time invariant systems, we expect that adding a white noise to the input signal makes it persistent exciting. To study this phenomenon, we add different levels of noise to the predefined smooth trajectories we used in the previous experiment. More precisely, the noise comes from $N(0, \sigma^2 I_{3\times 3})$ with $\sigma = 0.005, 0.01, 0.05$. Fig. 3 shows the effect of noise on the quality of global estimation[6].

This figure indicates that the increased level of noise leads to better estimates. This is more evident for LLS method, though even $\sigma = 0.05$ works much better than e.g. $\sigma = 0.005$ for K-NN too.

C. *Effect of neighborhood size*

Now we study the effect of K, the size of neighborhood, on the performance of K-NN and LLS methods (See 14 and 17). Fig. 4 shows the effect of K for both methods. Here, we

[6]The data in this figure is generated similar to the previous experiment.

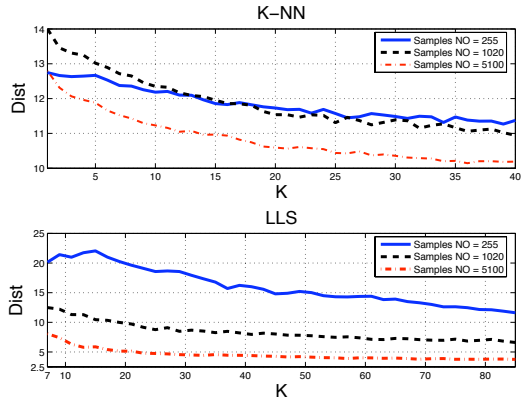

Fig. 4. The effect of K (number of neighborhoods) on estimation error (18) for *K-NN* (top) and *LLS* (bottom) methods with different number of samples.

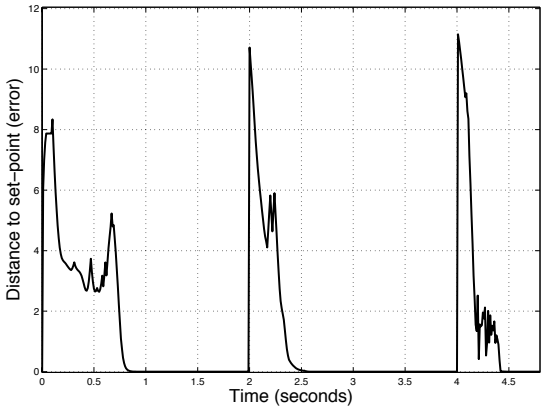

Fig. 5. The error of visual servoing using LLS estimator. The error is the Euclidean distance of the current feature position to the goal feature.

show results with different number of sample points. In this range of Ks, it seems that the bigger K results in smaller global error[7].

D. Visual servoing performance

Finally, we show the feasibility of these estimations for visual servoing task. We select several random end-effector positions, project them to our image spaces (produces a four dimensional feature point) to define random feature space goals. We use the visual-motor Jacobian estimate provided by LLS estimator combined with SVD-based design introduced in Section II-C. All singular values are set to 0.5 with an extra saturating controller. Fig. 5 shows the regulation result. The set-points are changed several time during the course of action, causing a spike in the errors seen in the figure. In each case, the robot regulates the error to zero successfully.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed two methods for global estimation of visual-motor models. The first method uses K-NN regressor to combine previously estimated visual-motor Jacobians. The second method directly uses data points to fit the best hyper-plane to a small neighborhood of the query point. Our simulations showed that these methods, specially LLS, outperform the conventional local estimation method. This result is important because of two reasons. First, it presents a new approach to uncalibrated visual servoing: a one in which designer can use previously unavailable tools such as dynamic programming and global nonlinear controllers to develop high-performance systems. We are going to pursue this direction in our future research. The second is showing the potential benefit of using the rich set of available regression tools instead of simple local model.

Nevertheless, there are several other research directions like extending ideas of Lapresté *et al.* and Mansard *et al.* for directly estimating global inverse Jacobian matrix instead of the global forward Jacobian which we did in this paper ([14][15]), developing a systematic method for selecting K in both K-NN and LLS, incorporating prior knowledge such as smoothness of visual-motor Jacobian into the estimator, and implementing these methods on a real-world hardware.

[7]These diagrams are the average result of 10 runs.

REFERENCES

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

[2] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true Jacobian," in *IEEE/RSJ International Conf. Intelligent Robots and Systems (IROS)*, vol. 1, September 1994, pp. 186–193.

[3] M. Jägersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *IEEE International Conf. Robotics and Automation (ICRA)*, vol. 4, April 1997, pp. 2874–2880.

[4] H. Sutanto, R. Sharma, and V. Varma, "The role of exploratory movement in visual servoing without calibration," *Robotics and Autonomous Systems*, vol. 23, pp. 153–169, 1998.

[5] J. A. Piepmeier, G. V. McMurray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *IEEE Trans. Robotics and Automation*, vol. 20, no. 1, pp. 143–147, February 2004.

[6] W. J. Wilson, C. C. W. Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Trans. Robotics and Automation*, vol. 12, no. 5, pp. 684–696, October 1996.

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

[8] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robotics and Automation*, vol. 8, no. 3, pp. 313–326, June 1992.

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

[10] K. J. Astrom and B. Wittenmark, *Adaptive Control*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[11] L. Gyorfi, M. Kohler, A. Krzyzak, and H. Walk, *A Distribution-Free Theory of Nonparametric Regression*. New York, USA: Springer-Verlag, 2002.

[12] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, Mar. 1996.

[13] G. Mariottini and D. Prattichizzo, "Egt: a toolbox for multiple view geometry and visual servoing," *IEEE Robotics and Automation Magazine*, vol. 3, no. 12, December 2005.

[14] J. Lapresté, F. Jurie, M. Dhome, and F. Chaumette, "An efficient method to compute the inverse Jacobian matrix in visual servoing," in *IEEE Intl. Conf. Robotics and Automation*, April 2004, pp. 727–732.

[15] N. Mansard, M. Lopes, J. Santos-Victor, and F. Chaumette, "Jacobian learning methods for tasks sequencing in visual servoing," in *IEEE/RSJ International Conf. Intelligent Robots and Systems*, October 2006, pp. 4284–4290.