Real Time L_∞-based Solution to Multi-view Problems with Application to Visual Servoing

A. H. Abdul Hafez
Automatic Control Lab.
Faculty of Electrical and Electronic Engineering
University of Aleppo, Syria
Email: hafez@research.iiit.ac.in

C. V. Jawahar
Center for Visual Information Technology
International institute of Information Technology
Gatchibowli, Hyderabad-500032, India
Email: jawahar@iiit.ac.in

***Abstract*—In this paper we present a novel real time algorithm to sequentially solve a class of multi-view geometry problems. The triangulation problem is considered as study case. The problem concerns with the estimation of 3D point coordinates given its images as well as the matrices of the concern cameras used in the imaging process. The algorithm has direct application to real time systems like virtual reality, visual SLAM, and visual servoing. Application to visual servoing is considered in detail. Experiments have been carried out for the general triangulation problem as well as the application to visual servoing.**

I. Introduction

Convex optimization has been widely accepted as a powerful tool to solve many engineering problems [1]. Its use has been extensively explored for solving a family of geometric reconstruction problems in computer vision. A wide variety of computer vision problems can be reformulated as convex optimization problems by some algebraic manipulations [2], [3], [4], [5]. Convex optimization, most importantly, do not have the pitfall of local minima assuring to achieve an optimal solution. In addition, Jacobean estimation through building a linearized model is not required in contrast to iterative minimization methods like gradient and Newton methods. The optimization is done here by replacing the L_2 norm objective function by the L_∞ norm *i.e.*, the maximum re-projection error (image point distance) for a set of image points.

Hartley and Schaffalitzky have shown in [6] that most of multi-view computer vision problems have a global minimum by minimizing the L_∞ norm. Soon after that Kahl in [4], and Ke *et al.* [7] have independently shown that this L_∞ is a quasi-convex function. Thus, it can be efficiently solved as a sequence of second order cone programs (SOCP) using the well known bisection algorithm. In fact, The solution proposed by Ke and Kanade [7] uses The mth smallest norm L_m instead of L_∞. This is due to the sensitivity of L_∞ to noise and outliers while the former one is robust. However, the solution in [7] becomes a local solution by considering the L_m norm as an objective function. Later, it was shown that another robustness technique can be introduced to the minimization of the L_∞ norm while keeping the ability of producing a global solution [8].

A wide variety of multi-view computer vision problems can be solved efficiently by minimizing the L_∞ norm using convex optimization framework. These problems may include triangulation, camera re-sectioning, plane to plane homography construction, and estimation of camera motion with a given rotation [3], [4]. However, the current framework is suitable for problems like building a model of a given environment. For example, given a video sequence with a calibrated camera, the 3D model of the scene can be efficiently built. This is usually known in computer vision community by the batch processing techniques. However, many computer vision problems need to be processed on-line satisfying the real time constraints.

This paper demonstrates the adaption of triangulation problem to the real time situations. To adapt to the real time requirements, we formulate the solution of the quasi-convex problem using bisection algorithm as an on-line recursive method. The real time solution has applications in many problems like visual SLAM, structure from motion, virtual reality, visual servoing. However, the adaptation to the real time case can be applied to a wide variety of computer vision problems solved by minimizing the L_∞ using convex optimization framework. The only sequential solution to the minimization of L_∞ norm has been recently proposed in [9]. The main difference between this work and ours is that the solution in [9] assumed an initial solution of the problem is available. This solution is obtained by running the full bisection algorithm which is usually time consuming. The algorithm keeps tracking of the changes in the estimated variables by updating the objective function and run only one or two iterations of the bisection algorithm. The algorithm proposed in this paper start from uniform distribution of the unknown variable. At each iteration it updates the objective function and gives an approximated estimate of the state variable. This estimates are assumed to converge to the accurate solution after a few iterations.

II. Background and Previous Works

A. Multi-views Problems in Computer Vision

1) The Triangulation Problem: The triangulation problem is the estimation of the 3D coordinates of a scene points given their image measurements from N views. It is assumed that camera (or cameras) is fully calibrated. In other words, the camera position and rotation are known. The problem is reduced to estimate the 3D point coordinates M considering

that N images m_i of a 3D point M are collected with $\mathcal{P}_i$ cameras are available. In other words, $m_i = \mathcal{P}_i M$. Here $i = 1, \cdots, N$ and $\mathcal{P}_i$ is the i^{th} camera matrix. Since the camera matrices $\mathcal{P}_i$ are supposed to be estimated independently as pose distribution, the problem is reduced to the estimation of the 3D point M given its N images m_i and N camera matrices $\mathcal{P}_i$. This problem has been formulated within quasi-convex optimization framework [4]. Robust solutions with respect to outliers are proposed in [7], [8]

2) Camera Re-sectioning Problem: The camera re-sectioning problem is used in camera calibration and structure from motion. The problem is formulated as the estimation of the (3×4) camera matrices $\mathcal{P}_i$ where $i = 1, \cdots, N$, given the 3D coordinates of a scene point M and their image measurements m_i using the N cameras. Since the perspective projection model is given as $m_i = \mathcal{P}_i M$, The problem is to estimate the $x \in R^{11}$ vector as a function of the image measurements m_i and 3D coordinates M of the scene point.

3) Homography Estimation Problem: Let us have a set of 3D planar points M_i where $i = 1, \cdots, N$. These planar points are related to its corresponding image points m_i using any camera matrices by a projective transformation H such as $m_i = HM_i$. In addition, there is similar relation H_{12} between any two image correspondences m_1 and m_2 of the same 3D points in such a way that $m_1 = H_{12} m_2$. This what is called the inter-image homography. Homography has many applications including structure from motion, layer extraction and visual servoing.

4) Cameras Positions with Known Rotations: There are many situations where the camera rotation is known independently on the translations or camera positions. These situations may include the pure translation motion and the case where the rotation is estimated using another, non-vision, sensor such as inertial sensors. Modern gyroscopes provide an accurate rotation measurements while positions information from accelerometers is still noisy. Another case is that some structure from motion methods estimate the rotation in a first step [10]. However, the camera matrices $\mathcal{P}_i$ are given as $\mathcal{P}_i = (R_i, -R_i D_i)$. The unknown variable to be estimated here is the camera positions $\{D_i\}$ and the 3D points M simultaneously. This is with the availability of image measurements from N view.

B. L_2 and L_∞ Norms of the image Re-projection Error

The discussion in this section and the extension to the real-time focus on the triangulation problem but it is general and can be applied to all the above problems.

Reconstruction algorithms usually minimize the image re-projection error because it is geometrically meaningful comparing with the algebraic one that is geometrically meaningless. The image re-projection error is the distance between the image measurement $\hat{m}$ and its 3D corresponding point M. More formally, let us assume that $\hat{m}_i = \mathcal{P}_i \hat{M}$ are the images of M using the camera matrices $\mathcal{P}_i$. Considering the image measurement $\hat{m}_i$, the norm L_n of the image re-projection error is written as

$$L_n \;=\; \| \hat{m}_i - m_i \|_n \;=\; \| \hat{m}_i - \mathcal{P}_i M \|_n \,. \tag{1}$$

When the Euclidean distance is used, we have L_2 norm. More formally, we can write

$$L_2 \;=\; \left[\frac{1}{N} \sum_{i=1}^{N} d_i^2 \right]^{1/2} \tag{2}$$

where

$$d_i^2 = \left(\hat{m}_i^1 - \frac{P_i^1 M}{P_i^3 M} \right)^2 + \left(\hat{m}_i^2 - \frac{P_i^2 M}{P_i^3 M} \right)^2 . \tag{3}$$

Here, P_i^j is the j^{th} row of the i^{th} camera matrix $\mathcal{P}_i$. Similarly, the L_∞ norm is defined as

$$L_\infty = \max_i d_i \tag{4}$$

where d is the same as given in (3).

It was shown in [6] that the L_2 is difficult to be minimized since it contains many local minima. In contrast, it is easier to minimize the norm L_∞ as it has only one minimum. Thus any local solution to the minimization of L_∞ is a global solution. Unfortunately, the L_∞ norm is sensitive to outliers and it needs additional processing step to increase the robustness of the optimization process.

C. Convex and Quasi-convex Optimization

A function $f : R^n \rightarrow R$ is a *convex function* if its domain, dom$f$ is convex and $\forall x, y \in$ dom$f$ and $0 \leq \theta \leq 1$

$$f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y) \tag{5}$$

The above in-equation, called Jensen's inequality, geometrically means that the region bounded by the line segment between $(x, f(x))$ and $(y, f(y))$ and the curve f lies above the curve, see Fig 1. In fact, all linear functions are convex while most of non-linear functions are not convex. A function $f : R^n \rightarrow R$ is called *quasi-convex function* if its domain and all its sub-level sets

$$\{ x \in \text{dom} f \mid f(x) \leq \alpha \} \tag{6}$$

are convex $\forall \alpha \in R$. It is clear from the above definitions that all convex functions are quasi-convex, but the inverse is not true. In other words, not all quasi-convex functions are convex.

The convex optimization problem is the one that minimizes an objective function $f_0(x)$ subject to a set of constraints functions $f_i(x)$; where $i = 1, \cdots, N$, and the functions $f_i(x)$ with $i = 0, \cdots, N$ are convex. More formally, the convex optimization problem can be one of the form

$$\begin{gathered} \min_x f_0(x) \\ \text{s.t. ,} \\ f_i(x) \leq b_i, \quad i = 0, \cdots, N. \end{gathered} \tag{7}$$

Given the function $f_0(x)$ is quasi-convex and the constraint function are convex functions, the problem is quasi-convex optimization problem and can be efficiently solved for a global

solution by solving a set of convex feasibility problem using the bisection algorithm.

For a quasi-convex function $f_0(x)$, the problem given in (7) is quasi-convex problem. Given the parameter $\gamma \in R$ and optimal but unknown solution $f_0^*(x) \leq \gamma$ to problem (7), the following feasibility problem is feasible:

$$\begin{aligned} &\text{find } x \\ &\text{s.t. ,} \\ &f_0(x) \leq \gamma \\ &f_i(x) \leq b_i, \quad i = 0, \cdots, N. \end{aligned} \tag{8}$$

If this problem is not feasible, then the solution f_0^* of problem (7) satisfies that $f_0^*(x) > \gamma$.

The bisection algorithm proceed in the same way to solve the quasi-convex problem. As illustrated in Algorithm 1, it starts by a known lower and higher bounds $\left[\gamma^l, \gamma^h\right]$ of the unknown optimal solution f_0^*. The first step is to solve the feasibility problem given in (8) for the lower half of the range $\left[\gamma^l, \gamma^h\right]$. If it is feasible for this lower half, update the higher bound feasibility range as $\gamma^h = (\gamma^l + \gamma^h)/2$. If the problem in (8) is not feasible, the optimal solution satisfies $f_0^*(x) > \gamma$. In this case the lower bound is updated as $\gamma^l = (\gamma^l + \gamma^h)/2$. After certain number of iterations and solving a set of feasibility problems, the range of the feasibility becomes $\gamma^h - \gamma^l \leq \epsilon$ and the produced solution is optimal.

Algorithm 1 Off-line Solution to the Quasi-convex Problem via Bisection Algorithm

1: **Input:** Given N image measurements, the range $\left[\gamma^l, \gamma^h\right]$ of the optimal value $f_0^*(M)$, and tolerance $\epsilon > 0$.
2: **Repeat**
   a) $\gamma = (\gamma^l + \gamma^h)/2$.
   b) Solve the convex feasibility problem as in (11).
   c) **If** feasible, $\gamma^h = \gamma$;
      else, $\gamma^l = \gamma$.
3: **Until** $\gamma^h - \gamma^l \leq \epsilon$.

A Second Order Cone Programming (SOCP) problem is also a convex optimization problem with a linear objective function minimized over a set of affine and quadratic constraints. An SOCP problem with m in-equations will look as

$$\begin{aligned} &\min_x f^T x \\ &\text{s.t. ,} \\ &\left\|Ax - b\right\| <= c^T x + d \\ &\text{where, } x \in R^n, \ A, c \in R^{m \times n}, \ b, d \in R^m, \end{aligned} \tag{9}$$

An LP problem is a special case of an SOCP problem. Quadratic Programming (QP) problems can be reformulated to be solved as SOCP problems.

D. Triangulation by Convex Optimization given N Views

The triangulation problem can be formulated as quasi-convex optimization problem with respect to the 3D coordinates of the features (X, Y, Z). The model considers that

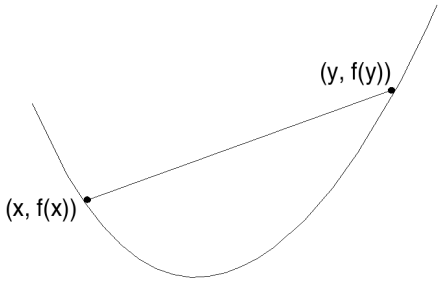


Fig. 1. The estimated 3D point coordinates over iterations.

N images of a 3D point M collected with $\mathcal{P}_i$ cameras are available. Here $i = 1, \cdots, N$ and $\mathcal{P}_i$ is the ith camera matrix. Since the camera matrices $\mathcal{P}_i$ are supposed to be estimated independently as pose distribution, the problem is reduced to the estimation of the 3D point M given its N images and N camera matrices. This problem has been formulated within quasi-convex optimization framework [4].

Kahl proposed in [4] to formulate the triangulation problem as

$$\begin{aligned} &\min_M \max_i d(\hat{m}_i - \mathcal{P}_i M) \\ &\text{subject to } \ \lambda_i(M) > 0 \qquad i = 1, \cdots, N. \end{aligned} \tag{10}$$

The function $d(\hat{m}_i - \mathcal{P}_i M)^2 = \frac{f_1(M)^2 + f_2(M)^2}{\lambda_i(M)^2}$ here is the Euclidean distance between the image measurement $\hat{m}_i$ and the projection of the 3D point M to the image of the camera $\mathcal{P}_i$, and $\lambda_i(M)$ is the depth of the point in image i. This problem can be solved using the bisection algorithm via a sequence of convex feasibility problems of the form

$$\begin{aligned} &\text{find } \ M \\ &\text{subject to } \ \| \left[f_{i1}(M)^2, f_{i2}(M)^2\right] \| \leq \gamma \lambda_i(M) \\ &\lambda_i(M) > 0 \qquad i = 1, \cdots, N. \end{aligned} \tag{11}$$

The convex feasibility problem tries to find out whether the optimal solution $f_0^*(M)$ is less or more than a given value γ. Thus, the quasi-convex function given in Eq. (10) can be solved through a sequence of feasibility problems using the bisection algorithm explained in Algorithm 1. The author in [4] solve the problem given a set of N images of the 3D point M. The bisection algorithm iterate over the measurement values k times to reach with the objective function to a value such that $f_0^*(M) \leq \epsilon$. In fact, the k iterations, usually k is from 5 to 10, cannot be performed in the real time (the video rate).

In the following subsection, We propose an efficient sequential bisection algorithm that is able to work and satisfy the real time constraints.

III. Real time Solution to the L_∞ norm

Let us assume a moving camera attached to frame F^t and has the camera matrix $\mathcal{P}_t$ at every time instance t. The camera frame F^0 is assumed to be the reference frame. Let us note that γ is set as an upper bound of the objective function in problem (10). Consequently, we can say that

$d(\hat{m}_i - \mathcal{P}_i M) < \gamma$ for $i = 1, \cdots, N$. Assume that the optimal value γ^* is bounded between a known lower threshold γ^l and higher one γ^h. Finally, the sequential bisection algorithm works as described in Algorithm 2.

The algorithm starts as soon as two views $N = 2$ are available. At each time instance t, the algorithm solves single convex feasibility problem as one iteration of the quasi-convex triangulation problem. Whenever new image measurements are available in the next time instance, the objective function $\max_i d(\hat{m}_i - \mathcal{P}_i M)$ and the bounding thresholds are updated based on the new measurements. Then, new convex feasibility problem is solved. The solution of the triangulation problem is expected to converge within 5-10 frames.

Algorithm 2 On-line Sequential Solution to the Quasi-convex Problem via Bisection Algorithm

1: Input: $N = 2$ image measurements, optimal value $f_0^*(M) \in [\gamma^l, \gamma^h]$ and tolerance ϵ. Initially, the time parameter t is set to N.
2: Repeat
   a) Collect the measurements from the N^{th} image.
   b) **If** $d(\hat{m}_N - \mathcal{P}_N M) \geq \gamma$ **then**, $\gamma^h = d(\hat{m}_N - \mathcal{P}_N M)$.
   c) $\gamma = (\gamma^l + \gamma^h)/2$.
   d) Solve the convex feasibility problem as in (11).
   e) **If** feasible, $\gamma^h = \max_i d(\hat{m}_i - \mathcal{P}_i M)$, **else**, $\gamma^l = \gamma$.
   f) $N = N + 1$
3: Until $\gamma^h - \gamma^l \leq \epsilon$

A. Uncertainty Estimation of Estimated 3D point coordinates

The uncertainty in the 3D feature estimates by solving the convex feasibility problem can be approximated as Gaussian. The 3D feature will be initialized as $M = [X,\ Y,\ Z]^T$ Recall that the on-line bisection algorithm solves one convex feasibility problem at each time iteration and after reaching a new image measurements. In addition, The solution returns one arbitrary value for the 3D estimate that satisfy the set of convex constraints which was set by the problem; and belong to the 3D domain of M that is determined by the current two bounds $[\gamma^l, \gamma^h]$.

Consider that the returned value by the convex feasibility program at time t is M_t where $t = 1, \cdots, N$, the mean of the 3D feature estimate is simply the sum $\bar{M} = \frac{1}{N}\sum_{t=1}^{N} M_t$ over the set of all the N seen estimates. Similarly, the covariance matrix $Q_M = \frac{1}{N+1}\sum_{t=1}^{N}[M - \bar{M}][M - \bar{M}]^T$. One may note that above mean and covariance can be updated recursively without need to keep storing the previous estimates. The mean 3D coordinates at time instance t is given as

$$\bar{M}_t = \frac{t-1}{t}\bar{M}_{t-1} + \frac{1}{t}M_t,$$

while the covariance matrix is updated as

$$Q_t = \frac{1}{t+1}\left\{t\ Q_{t-1} + [M_t - \bar{M}_t][M_t - \bar{M}_t]^T\right\}.$$

This estimate is assumed to converge to Gaussian distribution within a few iterations of the process of solving the convex feasibility problem.

IV. Application to Visual Servoing

The problem of visual servoing is that of positioning the end-effector of a robot arm such that a set of current features S reaches a desired value S^*. In image-based visual servoing, the set S can be composed of the coordinates of points that belong to the target object. The main objective of the visual servoing process is to minimize the error function given by

$$e(S) = S - S^*, \tag{12}$$

where S is a vector represents the current set of features and S^* is the vector represents the desired set of features.

By differentiating this error function with respect to time, with the desired features S^* remaining constant, we get

$$\frac{de}{dt} = \frac{dS}{dt} = (\frac{\partial S}{\partial P})\frac{dP}{dt} = L_S V, \tag{13}$$

Assuming a perspective projection model with a unit focal length, the interaction matrix L_{S_i} for each point (u, v) is given by:

$$L_{S_i} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{u}{Z} & uv & -(1+u^2) & v \\ 0 & \frac{-1}{Z} & \frac{v}{Z} & 1+v^2 & -uv & -u \end{bmatrix}. \tag{14}$$

For a set of N points, the set of features is S_i, $i = 1, ..., N$, the interaction matrix L_S is

$$L_S = \begin{bmatrix} L_{S_1}, \ldots, L_{S_N} \end{bmatrix}^T, \tag{15}$$

where L_{S_1} and L_{S_N} are the interaction matrices given in (14) and correspond to points 1 and N respectively.

For exponential convergence of the minimization process and using a simple proportional control law, , we need $\frac{de(S)}{dt} = -\lambda e(S)$. The required velocity of the camera can be shown to be [11]

$$V = -\lambda L_S^+ e(S), \tag{16}$$

where $e(S)$ is a $(2N \times 1)$ error vector between the image coordinates (u, v) of N points. The velocity $V = \frac{dP}{dt} = (v^T, \omega^T)^T$ is the camera velocity, v is translational velocity and ω is rotational velocity. The pose vector $P = (x, y, z, \alpha, \beta, \gamma)$ is a (6×1) vector. The $(2N \times 6)$ matrix L_S^+ is called the pseudo-inverse of the image Jacobian. Image Jacobian relates the changes in the image space to the changes in the Cartesian space [11].

By substituting in (13) where L_S^+ is the pseudo inverse of the Jacobian matrix L_S, and λ is a scale factor. The Jacobian matrix can be written as

$$L_s = \frac{1}{Z}A(U, V) + B(U, V), \tag{17}$$

where U and V are the image coordinate vector of all points. One can note that image-based visual servoing a knowledge or information about the depth of the target points.

It was assumed that a rough estimates of the depth is enough for a stable control law in image-based visual servoing. Recently in [12], it was shown that the stability range with

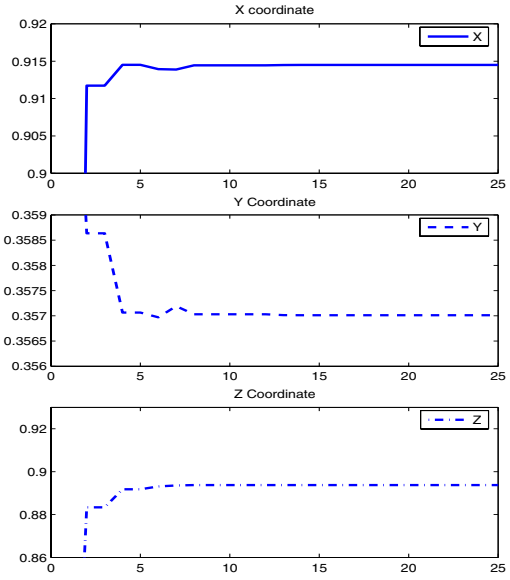


Fig. 2. The estimated 3D point coordinates over iterations.

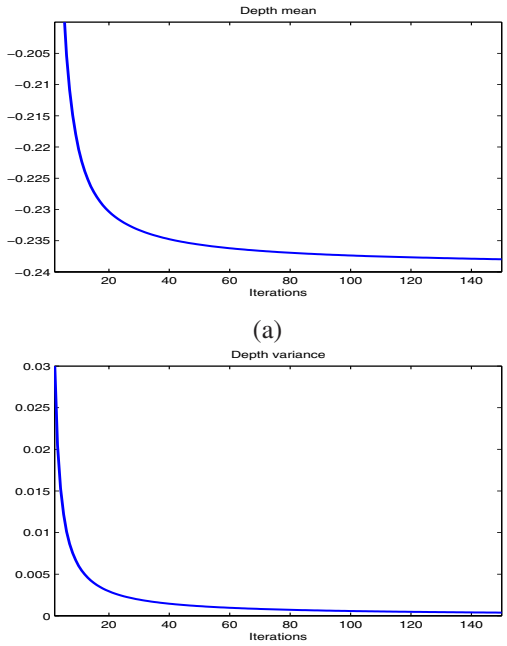


Fig. 3. The depth estimate over iterations expressed in the reference frame. The mean is shown in (a) and the variance in (b).

the depth estimation is not so much wide. Hafez has proposed in [13] a particle filter based camera pose tracking algorithm with application to visual servoing. The algorithm produce the current estimate of the camera pose using the image measurements and the velocity signal as an input. In other words, the algorithm provides an estimate of the camera matrix $\mathcal{P}_i$ at each iteration of the servoing process. The knowledge about the scene is used to initialize the depth parameters Z_i. During the servoing process, the depth estimates are updated on-line using the algorithm presented in Sec. III.

V. Results and Applications

The illustrated experimets have carried out in a simulation framework. We modify the original L_∞ algorithm which was built based on SeDuMi [14]. Firstly we show the results of applying the algorithm on a randomly generated data. Application to visual servoing is shown in the next section.

A. Preliminary Results

These preliminary results have been carried out by considering a synthetic scene [4]. This scene consists of randomly generated camera matrices, a set of 3D points, and their images. The camera matrices and the measurements of image points are the input data to the algorithm. The process starts as and when two images and two camera matrices are available. The SuDeMi is called for the minimal number of iterations over the bisection algorithm. This number of iterations is enough to return a feasible solution as well as to enable the whole system to run in the real time. This is repeated when new image measurements and camera matrix are available as described in Algorithm 2. Figure 2 shows the estimated 3D coordinates over iterations.

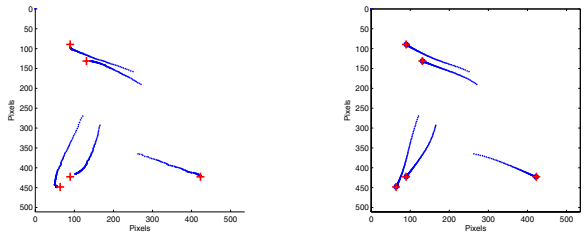
B. Results from Visual Servoing

The sequential triangulation algorithm has been applied to estimate the depth of image features for visual servoing system. The experiments have been carried out within a simulation framework. As mentioned in the previous section, we assume that a pose tracking algorithm is running to provide a pose estimate independent on the 3D estimate of the concern features. These pose estimates are used along with the image measurements to estimate the 3D coordinates with respect to a reference frame.

The estimated depth of a single point feature is represented as mean and variance. Figure 3.(a) shows the mean depth of a selected point. The variance of the depth is shown in Fig. 3.(b). A comparison has been carried out between two visual servoing process with the same initial and desired configurations. The difference between the two process is the depth distribution used in the control law. In the first case, a depth distribution ith error about 25% of the actual depth value. In the second case, we use the depth distribution produced by our sequential triangulation algorithm. The image trajectories in case of 25% depth error is shown in Fig. 4.(a). It is clear that the system has converged to final state which is different from the desired one. The desired positions of the image features are marked by (+). The image trajectories in case of using the depth estimate produced by our sequential triangulation algorithm is illustarted in Fig. 4.(a). The velocity of the camera for the two cases is shown in Fig. 6.

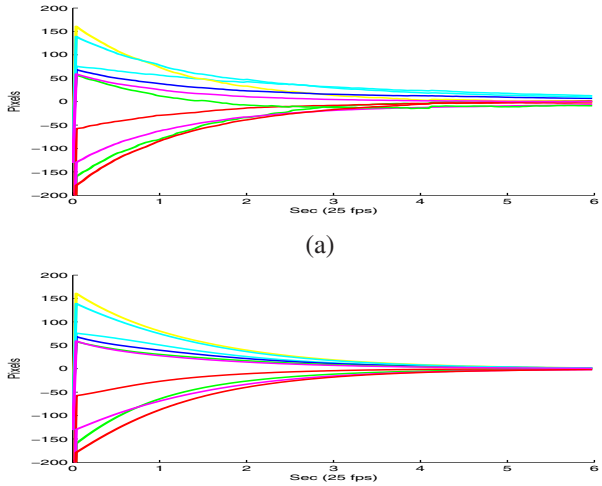
VI. Conclusion

We have presented a sequential algorithm for solving a set of problems in multi-view geometry. The algorithm provides an efficient solution to these problems in the real time. The algorithm start from unknown estimate and its output



(a) (b)

Fig. 4. The image trajectories during the visual servoing process. In (a), the trajectory is shown when there is 25% error in the depth distribution. In (b), the trajectory is shown when the estimate of the depth by our algorithm is used.



(a)

(b)

Fig. 5. The image coordinates error during the visual servoing process. In (a), the error is shown when there is 25% error in the depth distribution. In (b), the error is shown when the estimate of the depth by our algorithm is used.

converges to the actual values within a few iterations. This algorithm is used to estimate the depth of the image features for visual servoing system. The visual servoing process has converged properly by using the 3D estimates produced by our algorithm.

REFERENCES

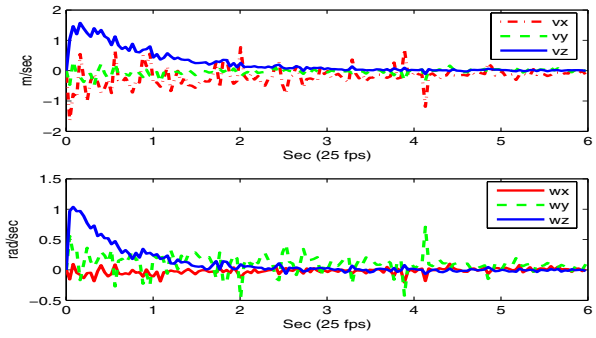
[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[2] M. Salzmann, R. Hartley, and P. Fua, "Convex optimization for deformable surface 3-d tracking," in *Proceedings of the Tenth IEEE International Conference on Computer Vision, ICCV'07*, Rio de Janeiro, Brasil, October 2007.

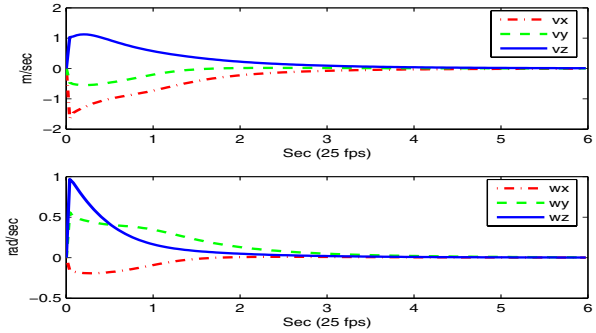
[3] R. Hartley and F. Kahl, "Optimal algorithms in multiview geometry," in *Computer Vision – ACCV 2007*, ser. LNCS, Y. Yagi, S. B. Kang, I. S. Kweon, and H. Zha, Eds., vol. 4843. Springer, 2007, pp. 13–34.

[4] F. Kahl, "Multiple view geometry and the L_∞ norm," in *Proceedings of the Tenth IEEE International Conference on Computer Vision, ICCV'05*, Beijing, China, 2005, pp. 1002–1009.

[5] Q. Ke and T. Kanade, "Uncertainty models in quasiconvex optimization for geometric reconstruction," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'06*, New York, NY, USA, 2006, pp. 1199–1205.



(a)



(b)

Fig. 6. The image coordinates error during the visual servoing process. In (a), the error is shown when there is 25% error in the depth distribution. In (b), the error is shown when the estimate of the depth by our algorithm is used.

[6] R. Hartley and F. Schaffalitzky, "L_∞ minimization in geometric reconstruction problems," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition, CVPR'04*, vol. 01. Los Alamitos, CA, USA: IEEE Computer Society, 2004, pp. 504–509.

[7] Q. Ke and T. Kanade, "Quasiconvex optimization for robust geometric reconstruction," in *Proceedings of the Tenth IEEE International Conference on Computer Vision, ICCV '05*, Beijing, China, 2005, pp. 986–993.

[8] K. Sim and R. Hartley, "Removing outliers using the l_∞ norm," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'06*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 485–494.

[9] Y. Seo and R. Hartley, "Sequential l¡subscript¿ ∞ ¡/subscript¿ norm minimization for triangulation," in *Computer Vision – ACCV 2007*, ser. LNCS, Y. Yagi, S. B. Kang, I. S. Kweon, and H. Zha, Eds., vol. 4844. Springer, 2007, pp. 322–331.

[10] M. Uyttendaele, A. Criminisi, S. B. Kang, S. Winder, R. Szeliski, and R. Hartley, "Image-based interactive exploration of real-world environments," *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 52–63, 2004.

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

[12] E. Malis and P. Rives, "Robustness of image-based visual servoing with respect to depth distribution errors," in *IEEE Int. Conf. on Robotics and Automation, ICRA'03*, vol. 1, Taipei, Taiwan, Sept. 2003, pp. 1056–1061.

[13] A. H. Abdul Hafez, "Enhancing hybrid visual servo control by probabilistic techniques," Ph.D. dissertation, Osmania University, Hyderabad, India, May 2007.

[14] J. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999, special issue on Interior Point Methods (CD supplement with software). [Online]. Available: citeseer.ist.psu.edu/sturm99using.html