Optimizing Image and Camera Trajectories in Robot Vision Control using On-line Boosting

A. H. Abdul Hafez [1], Enric Cervera [2], and C.V. Jawahar [1]

Abstract— **In this paper, we present a novel boosted robot vision control algorithm. The method utilizes on-line boosting to produce a strong vision-based robot control starting from two weak algorithms. The notion of weak and strong algorithms has been presented in the context of robot vision control, in presence of uncertainty in the measurement process. Appropriate probabilistic error functions are defined for the weak algorithm to evaluate their suitability in the task. An on-line boosting algorithm is employed to derive a final strong algorithm starting from two weak algorithms. This strong one has superior performance both in image and Cartesian spaces. Experiments justify this claim.**

I. INTRODUCTION

Servo vision is the widely used vision-based robot control scheme, where control loop of the robot is closed using visual information. Visual information can be either from the 2D image space or from the 3D Cartesian pose space [1]. A typical control algorithm minimizes an error function between the current position of the camera and the desired one [2]. Based on the objective function used in the minimization process, we have 2D (image-based) and 3D (position-based) vision control algorithms. Chaumette [1] has shown that each of these two classes of algorithms has weak points (drawbacks) or potential problems. Owing to the complementary properties of the weak points in the two algorithms, hybrid methods [3], [4] have been recently proposed to integrate the advantages and discard the drawbacks.

Hybrid methods integrate the 2D and 3D information in the feature space [3], [4]. Methods like [5], [6] integrate the 2D and 3D information in the action space. Gans and Hutchinson [6] presented a switching system between image-based and position-based vision control algorithms, while Hafez and Jawahar [5] present a smooth linear combination of different methods. Another method of switching is presented by Chesi *et al.* [7]. It switches between elementary camera motions, mainly rotation and translation extracted by decomposing the homography matrix between the current and desired views. Here, we utilize on-line boosting to enhance the visual control from the two basic (image-based and position-based) vision control algorithms. Each of these weak algorithms performs well only over a subspace of the input domain. The final output is the boosted control signal derived from the independent control algorithms. The problem of deriving robot visual control reduces to that of identification of a weighted sum of the output of these weak algorithms using weights computed from the error function defined for each of the independent weak algorithms.

We use boosting [8] for efficient positioning task in active vision systems, and demonstrate on robot vision control problem. A learning algorithm has been used to learn the relationship between motions in the image space and the Cartesian space. This relation is called the image Jacobean or the interaction matrix [9], [10]. Another work uses an adaptive learning algorithm to minimize the control error [11]. Based on the analysis and results presented in this paper, we argue that introducing boosting can significantly improve the performance and enhance the range of applications of robot vision control algorithms.

II. BOOSTING AND VISION CONTROL ALGORITHM

Boosting is a general method for improving the performance of a given multiple weak hypotheses. In other words, boosting transforms a set of weak algorithms into a strong one. We first introduce the notion of weak and strong algorithms in the context of the robot vision control.

A. Weak and Strong Algorithms

1) Weak Algorithms:: Let the function $y = f(x)$ that maps $R^n \longrightarrow R^m$ represents the algorithm that takes x as an input and results the output vector y. A weak vision control algorithm is defined here as follows: For a subset $X \in R^n$ of the function domain, error e is greater than an ϵ. Some algorithms compute the error e using training data as in the case of classifiers or could be computed using the actual and desired output as in the case of control algorithms.

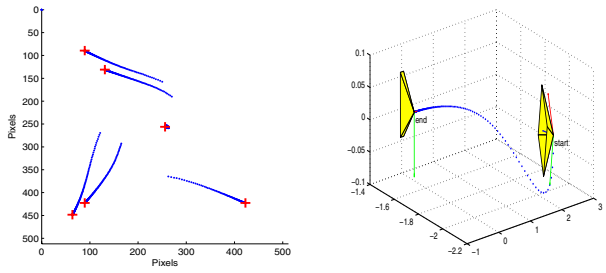
2) Strong Algorithms:: Given a set of N weak algorithms $y_i = f_i(x)$, a strong algorithm is defined as a linear combination of the N weak algorithms $y_i = f_i(x)$. This can be written as

$$Y = F(x) = \sum_{i=1}^{N} \alpha_i f_i(x) = \sum_{i=1}^{N} \alpha_i y_i. \qquad (1)$$

Let the function $e_i = p_i(y, x)$ represents the undesirability in the performance of the concerned weak algorithm. This error is increased when the performance of the weak algorithm $y_i = f_i(x)$ is less satisfactory. Weight that determines

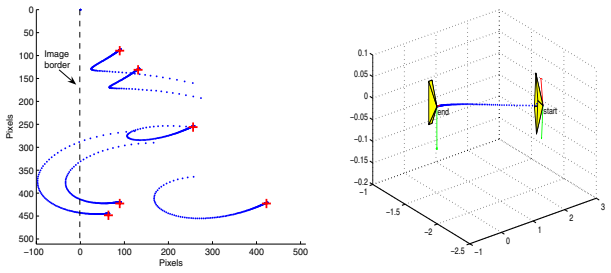
[1] Center for Visual Information technology, International Institute of information Technology, Gachibowli, Hyderabad-500032, India {hafez@reserach,jawahar@}iiit.ac.in

[2] Robotic Intelligence Lab, Jaume-I University of Castell, Spain. enric.cervera@icc.uji.es. The Robotic Intelligence Lab is partially funded by the EU-VI Framework Programme under grant IST-045269 - GUARDIANS of the EC Cognitive Systems initiative, and by the Spanish Ministry of Science and Education (MEC) under grant DPI2005-08203-C02-01



(a) (b)

Fig. 1. Feature trajectories in the image space in (a), and the camera trajectory in the Cartesian space in (b) for the image-based visual servoing algorithm. The desired positions of the image features are marked by +.



(a) (b)

Fig. 2. Feature trajectories in the image space in (a), and the camera trajectory in the Cartesian space in (b) for the position-based visual servoing algorithm. The desired positions of the image features are marked by +.

the contribution of the weak algorithm to the strong one is defined as

$$\alpha_i = \frac{1}{2}\ln(\frac{1-e_i}{e_i}). \tag{2}$$

We select this method of weight computation to add an upper limit to the weight values. By normalizing the error function in a certain way, the weights will have always positive values. This helps in avoiding the local minima. In other words, it is not allowed for (1) to be zero.

B. Robot Vision Control Schemes

1) Image-based Vision Control:: In image-based vision control, the task function is defined with respect to the error $E_i(s) = s - s^*$ in the image space, where s is the vector of the current features position in the image and s^* is the vector of the desired one. The velocity screw using image-based vision control is given as [12]

$$V_i = -\lambda_i J_i^+ E_i(s), \tag{3}$$

where J_i^+ is the pseudo-inverse of the Jacobian matrix J_i. It is easy to show that the feature trajectory in the image space is a straight line [1]. From any initial state, image points move along straight line toward its desired positions in the image (see Fig.1.(a)). This is subject to the availability of a good estimate of the depth and robust image measurements [13]. However, the camera trajectory in Cartesian space is unpredictable (see Fig. 1.(b)). Since the complex Cartesian camera trajectory may cause the robot arm to get out of its workspace, the performance of the image-based vision control algorithms is evaluated by its capability to keep the robot arm in its workspace.

2) Position-based Vision Control:: In position-based vision control, the camera velocity is defined as a function of the error between the current and desired camera pose. This error is the transformation T_C^{C*} represented as a (6×1) vector $E_p(s) = [t_C^{C*}, u\theta]^T$. The velocity screw using position-based vision control is given as [12]

$$V_p = -\lambda_p J_p^{-1} E_p \tag{4}$$

where J_p is the interaction matrix [14]. While position-based vision control algorithms minimize the error function in the Cartesian space, the camera trajectory is a straight line (see Fig. 2.(b)). However, the trajectories of image points are not predictable and may get out of the camera field of view as illustrated in Fig. 2.(a).

A minimum number of image points are required to avoid the singularity of the Jacobean matrix [1]. Computer vision algorithms that are used in the control methods like pose and homography estimation algorithms [15], [4] need a minimum number of image point to be reserved. In fact, the performance of position-based vision control algorithms is evaluated by its capability to keep the visibility of the image points in the camera field of view.

III. BOOSTED VISION-BASED ROBOT CONTROL

Boosted visual control algorithm considers 2D and 3D algorithms as weak algorithms. A linear combination of these two weak algorithms produce a strong algorithm with satisfactory performance. Weights which are used in the combination are computed based on error function associated with each one of these weak algorithms.

A. General Error Functions in Visual Servoing Algorithms

In robot vision control algorithms, the error $e_t = p(y_t)$ is a function of the *observation* or *measurement* y_t at the time moment t, while the input is an *action* Δu in a given *environment* m. This action Δu is nothing but a change added to the previous system state P_{t-1}. The distribution of the current observation is called the belief $\pi(y_t)$ and given as:

$$\pi(y_{t+1}) = Pr(y_{t+1} \mid y_0, \Delta u_0, \ldots, y_t, \Delta u_t, P_t, m). \tag{5}$$

By introducing the *static world assumption* or what is known as *Markov assumption*, equation (5) can be written as

$$\pi(y_{t+1}) = Pr(y_{t+1} \mid \Delta u_t, P_t, m). \tag{6}$$

This is interpreted as that the past is independent of the future given knowledge about the current state.

Since the distribution of the weights does not need to keep track of the previously seen examples, the error function e_t can be written as

$$e_t = \int_{\mathcal{Y}} Pr(p(y_t) > \epsilon \mid P_t, m) dy_t. \tag{7}$$

Here, $\mathcal{Y}$ is the uncertainty area of the observation y_t.

The details of this error function is dependent on the vision-based control algorithm. In image-based control algorithm, the measurement of interest is the distance of one arm-joint q to its threshold θ_q at the current time moment. In Position-based control algorithm, the measurement of interest is the distance d of one image point to the image border.

B. Error Function for Image-based Vision Control

The performance of image-based vision control is measured by the ability of the working point q^i of the ith arm joint to avoid the joint limits $\{q^i_{min}, q^i_{max}\}$ of the robot arm. The joints configuration $\mathbf{q}$ of a robot arm is acceptable when $\forall i, q^i \in \left[q^i_{min} + \theta^i_q, q^i_{max} - \theta^i_q\right]$. Here, θ^i_q is a threshold of the ith joint. The error in the performance of image-based control algorithm can be measured as a function of the distance of the working point q^i of the ith joint to its concerned joint limit θ^i. Let the parameter $\{r^i_t\}_{i=1}^{N_q}$ be the distance of the joint q^i to its threshold θ^i_q at time moment t, where

$$r^i = \min\{q^i - q^i_{min} - \theta^i_q, q^i_{max} - \theta^i_q - q^i\} \tag{8}$$

and N_q is the number of the joints of the arm.

Considering the uncertainty in the joint measurements, it can be represented using the normal distribution $\mathcal{N}(r_t; \mu_r, \sigma_r)$, where the mean $\mu_r = r^i$ is the measurement of the ith joint.

$$[a, b] = [\mu_r - n\sigma_r, \mu_r + n\sigma_r]\,. \tag{9}$$

Considering the error function given in (7), we can write

$$\begin{aligned} e^q_t &= \int_a^b \frac{1}{\sqrt{2\pi}(\sigma_q + \sigma_r)} \exp\left[-\frac{a^2}{2\sigma_q^2} - \frac{(r_t - a)^2}{2\sigma_r^2}\right] dr_t, \\ &= \frac{1}{\sqrt{2\pi}(\sigma_q + \sigma_r)} \exp\left[-\frac{a^2}{2\sigma_q^2}\right] \left[\text{erf}(\frac{r_t - a}{2\sigma_r^2\sqrt{2}})\right]_a^b. \end{aligned} \tag{10}$$

Here, σ_q is selected in such that only working points within a minimum distance to its joint limits will contribute to the error function. By using $n = 3$ in (9) and substituting in (10), we get the following error function for each ith joint.

$$e^q_t = \frac{1}{\sqrt{2\pi}(\sigma_q + \sigma_r)} \exp\left[-\frac{(\mu_r - 3\sigma_r)^2}{2\sigma_q^2}\right] \text{erf}(\frac{3}{\sqrt{2}}). \tag{11}$$

Finally, the total error function of the image-based vision control algorithm is given as

$$e_{t(ibvs)} = \sum_{q=1}^{N_q} e^q_t. \tag{12}$$

This is the error function of the performance of the weak image-based vision control algorithm. A plot of the error with respect to the distance to the joint limit is illustrated in Fig. 3. Substituting in (2), we get the associated weight to the image-based vision control algorithm.

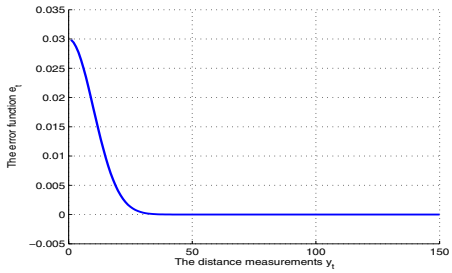


Fig. 3. The error function of the weak vision control algorithm. In position-based the measurement is the distance $y_t = d$ of the nearest image point to the image border, and in image-based is the distance of the nearest joint value $y_t = r$ to its limit.

C. Error Function for Position-based Vision Control

The performance of position-based vision control is measured by the ability of keeping the point features (u^i, v^i) visible in the camera field of view. The error in the performance of position-based vision control can be measured as a function of the distance of the ith point to the nearest image border or a threshold θ_I of the border. Let the parameter $\{d^i_t\}_{i=1}^{N_d}$ be the distance of the ith point to the nearest image border at time t, where

$$d^i = \min\{u^i - u_{min}, v^i - v_{min}, u_{max} - u^i, v_{max} - v^i\}, \tag{13}$$

and N_d is the number of image points.

Considering the uncertainty in the image measurements, this distance can be represented by the normal distribution $\mathcal{N}(d_t; \mu_d, \sigma_d)$, where $\mu_d = d^i$ the measurement of the ith image point . If we consider the same uncertainty range in (9) and similar development in (10), we can write the error function of the position-based vision control algorithm with respect to one image point as

$$e^i_t = \frac{1}{\sqrt{2\pi}(\sigma_i + \sigma_d)} \exp\left[-\frac{(\mu_d - 3\sigma_d)^2}{2\sigma_i^2}\right] \text{erf}(\frac{3}{\sqrt{2}}). \tag{14}$$

Here, σ_i is selected in such that only image points within a minimum distance to the image border will contribute to the error function. Finally, the total error function of position-based algorithm is given as

$$e_{t(pbvs)} = \sum_{i=1}^{N_d} e^i_t. \tag{15}$$

This is the error function of the performance of the weak position-based vision control algorithm. A plot of the error with respect to the distance to the image border is illustrated in Fig. 3. Substituting in (2), we get the associated weight to the position-based algorithm.

D. The Overall Boosting-based Algorithm

The objective is to position a robot arm with respect to a set of features or with respect to an object that contains a set of features. Equations (3) and (4) present two weak vision control algorithms, image-based vision control and position-based vision control algorithms, that can be used

Fig. 4. External view of the experimental setup.

to perform the said positioning task. The former has an undesired behavior in the Cartesian space while the later has an undesired behavior in the image space. On-line boosting produces a strong vision-based algorithm using the error functions defined in correspondence to each algorithm.

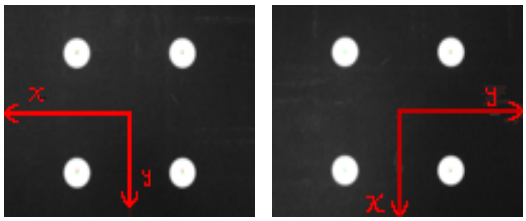
The general structure of the algorithm was explained in Sec II. For the specific image-based vision control and position-based vision control algorithms, the error functions explained in Sections III-B and III-C and given by (12) and (15) are evaluated and used to compute the corresponding weights α_i that gives the current importance to each algorithm. It was shown in [5] that a linear combination of two linear dynamic systems, as the system given in (1), is asymptotically locally stable. Details of the on-line boosted vision-based control algorithm are summarize in the following steps:

1) Collect the image measurements X_t where $X = \{x_m : m \in \{1, \ldots, M\}\}$, and compute the pose P_t at time t.
2) Compute the outputs of the weak vision-based control algorithms $V_j = f_j(X)$, as in (3) and (4). Here, $j \in \{IBVS, PBVS\}$.
3) Compute the error function for IBVS weak algorithm $e_{t(ibvs)} = \sum_{i=1}^{N_q} e_t^i$, where e_t^i is computed as in (11).
4) Compute the error function for PBVS weak algorithm $e_{t(pbvs)} = \sum_{i=1}^{N_d} e_t^i$, where e_t^i is computed as in (14).
5) Compute the weights $\alpha_j = \frac{1}{2}\ln(\frac{1-e_j}{e_j})$, where $j \in \{IBVS, PBVS\}$.
6) Normalize the weights α_j to $\bar{\alpha_j}$ to make them sum up to one.
7) Compute the output of the strong algorithm given in (1) as the linear combination of the weak algorithms outputs using the computed weights as follows $V = F(x) = \sum_{j \in \{IBVS, PBVS\}} \bar{\alpha_j} V_j(X)$.

IV. RESULTS

Our experimental setup consists of a Mitsubishi PA-10 robot arm, with a Point Grey Flea camera mounted on the end-effector. The camera delivers 60 fps at VGA resolution, and it is connected through a Firewire port to the computer which controls the arm. Camera intrinsic parameters are coarsely calibrated. Though the arm has 7 DOF, only 6 of them are controlled via Cartesian velocity commands in the end-effector frame.

The target used in our experiments is made of 4 white points in a 15 cm square, which is tracked with the ViSP



(a) (b)

Fig. 6. The initial (a) and desired (b) images of the first task: 90 degrees rotation about optical camera axis. The object frames are drown in red color. The camera optical axis is perpendicular to object plan and not visible in the image.

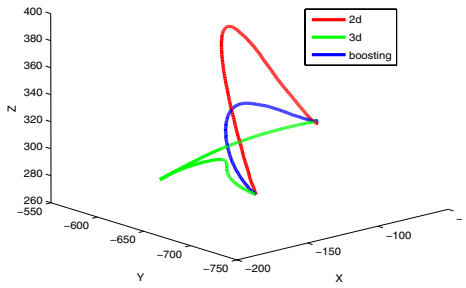


Fig. 7. The end-effector trajectories. Image-based in red color, position-based in green color, and boosting-based in blue color.

software [16]. Both 2D and 3D visual servoing tasks are defined, and the proportional control gain is set to 0.2. In the boosting algorithm, we set a threshold $\theta_I = 25.0$ pixels to the image border. Similar threshold $\theta_q = 30°$ is used for the joint parameters. The values of the parameters σ_i and σ_q are set to 50 pixels and 60 degrees respectively.

The most of robot vision algorithms work well for those task that involved simple motion. They have been developed in responce to specific task but they fail to perform some another specific challenging tasks. Some of these challenging tasks are stated in [17]. We carried out the experiments for two of these challenging tasks. The first task is rotation of $90°$ about the camera axis. The second one is rotation of $180°$ about the same axis.

(b)

Fig. 8. The object target as seen by arm in the middle of the task using position-based method. The feature has partially got out of the field of view.

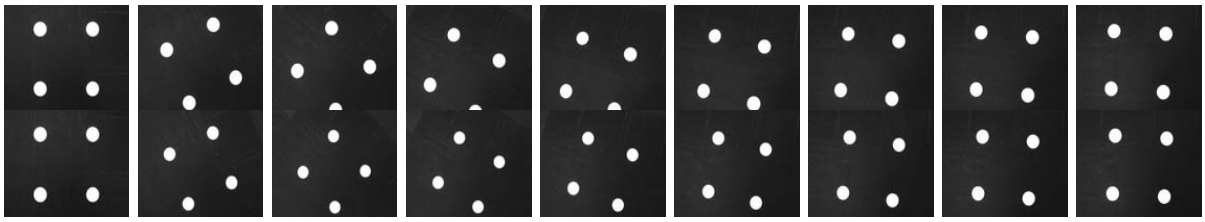
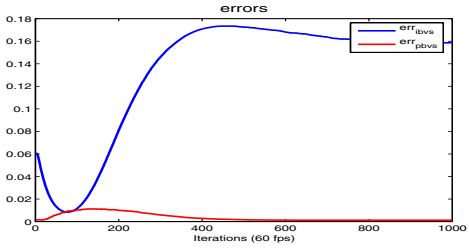
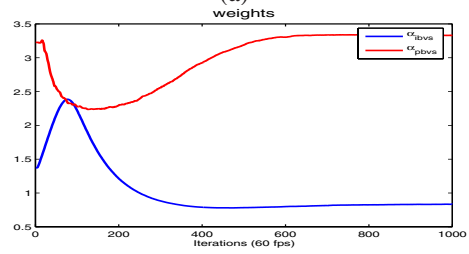


Fig. 5. Sequence of images show the image feature trajectories for position-based (top) and boosting-based (down) algorithm. The task is a 90° about the camera optical axis. One can not the that one point has partially got out of the field of view



(a)



(b)

Fig. 9. The error functions of image-based and position-based weak algorithms in (a), the corresponding weights in (b), of the boosting-based vision control algorithm. The task is a 90° about the camera optical axis.

A. Rotation of 90° about the Camera Optical Axis

This visual task consists of a 90° rotation around the camera optical axis. See Fig. 6.(a and b). Such task is troublesome for the classical image-based and position-based methods. In position-based method, the image features can get out of the camera field of view easily, while using 2D image-based method the camera retreats backward along the camera Z axis trying to perform a non-realizable motion by robot arm. Image trajectories using image-based vision control are shown in Fig. 10.(a). The straight line trajectories cause the camera to retreat backward along its Z axis, as shown in Fig. 7 and Fig. 10.(b).

In the second case, that is position-based, the camera trajectory is pure rotation about its optical axis. Thus, the image points describe a 90° arcs around the principal point of the image. Unfortunately, these arcs may get out of the camera field of view. The image trajectory and the screw velocity of this case are shown in Fig. 10.(c,d), while the trajectory of the end-effector is shown in Fig.7.

In Fig. 8, there is an image of the target object in the middle of the task using 3D position-based algorithm. One feature gets partially out of the camera field of view. The tracking keeps going on, but the centriod of the circle representing the feature moves away from its correct position. Then the model of the square is wrongly fitted. Indeed, the 3D pose is wrong. According to the red color frame, the camera is normal to the surface, thus the Z axis should go inside the image, but it departs quite significantly. That explains the strange motion of the camera frame in the middle of the task using 3D or position-based algorihtm in Figs. 10.(d) and 7.

The boosting method combines both schemes, and it seamlessly achieves the task while keeping the visual features in the field of view, see Fig. 10.(e,f), the performance in the image is similar to the performance of the case of using 2D algorithm. The end-effector is moving smoothly to the goal location, see Fig. 7. The shown end-effector trajectory is satisfactory and keeping the arm in its workspace. Fig. 9 shows the error functions of each of the image-based and position-based algorithms as a boosted weak algorithms and the corresponding weights in Fig. 9.(b).

Fig. 5 shows two image sequences. The top one is for the position-based (3D) algorithm. One of the feautres has partially got out of the field of view. The sequence at the bottom is using the boosting algorithm. The features have been successfully kept in the field of view.

B. Rotation of 180° about the Camera Optical Axis

This task is more troublesome for the classical image-based and position-based methods. In image-based, since the rotation is 180°, the camera retreats back to infinity. The robot arm obviously gets out its work space. As illustrated in Fig. 11.(a,b), the process stop after around 200 iterations. In position-based, one feature, that is near to the image border, gets out of the camera field of view. The process stop after approximatly 80 iterations as it illustrated in Fig. 11.(c,d). As it is shoown in Fig. 11.(e,f), the process completed successfully using the boosting-based control law. The performance in the image is queit improved since the image points can avoid the image border. In addition, The algorithm keeps the camera retreat as less as possible.

V. CONCLUSIONS

The on-line boosting algorithm has been used to improve the performance of each of image-based vision control and position-based vison control in the image and Cartesian spaces. It is shown that boosting algorithms can be generalized to much more than classifications and recognition like vision control application. Experiments has been curried out and showed a significant improvement to the performance of the classical vision-based control algorithms.

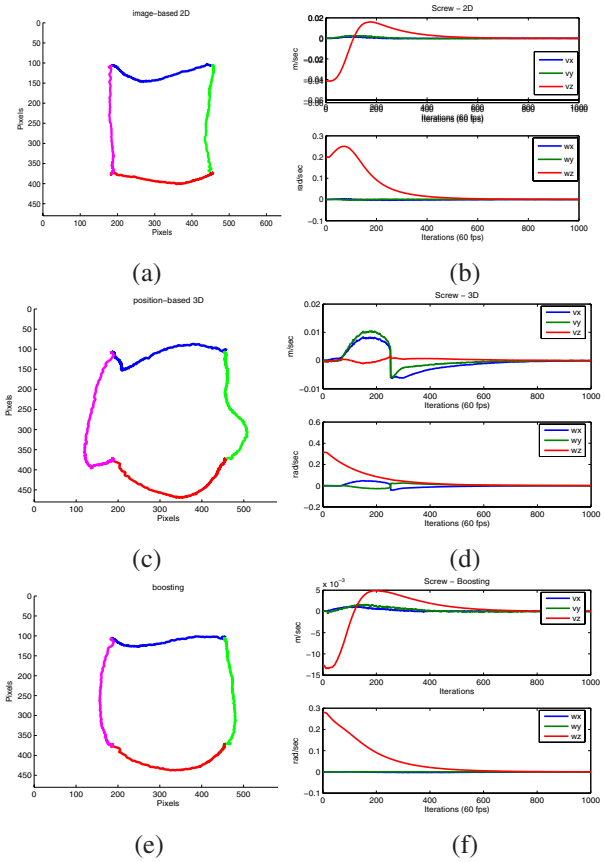


Fig. 10. The features' trajectories in the image space in (a,c,e) and the screw velocity in (b,d,f) of the image-based, position-based, and boosting-based vision control algorithms respectively. The task is a 90° about the camera optical axis.

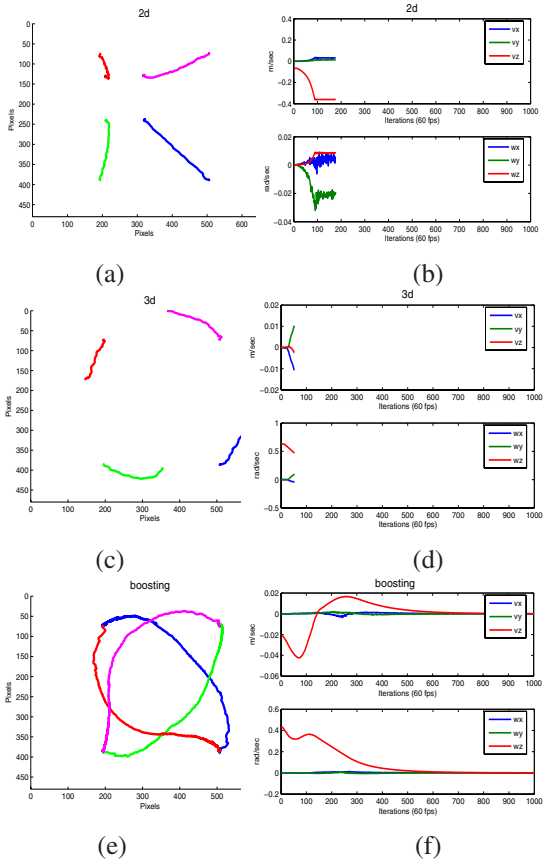


Fig. 11. The features' trajectories in the image space in (a,c,e) and the screw velocity in (b,d,f) of the image-based, position-based, and boosting-based vision control algorithms respectively. The task is a 180° about the camera optical axis.

REFERENCES

[1] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, D. Kriegman, G. . Hager, and A. Morse, Eds. LNCIS Series, No 237, Springer-Verlag, 1998, pp. 66–78.

[2] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *IEEE Int. Conf. on Robotics and Automation, ICRA'04*, New Orleans, USA, April 2004.

[3] V. Kyrki, D. Kragic, and H. I. Christensen, "New shortest-path approches to visual servoing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'04*, Sendai, Japan, September-October 2004, pp. 349–354.

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

[5] A. H. Abdul Hafez and C. V. Jawahar, "Probabilistic integration framework for improved visual servoing in image and cartesian spaces," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'06*, Bejing, China, October 2006.

[6] N. R. Gans and S. A. Hutchinson, "An asymptotically stable switched system visual controller for eye in hand robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'03*, Las Vegas, October 2003, pp. 3061–3068.

[7] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino, "Keeping features in the field of view in eye-in-hand visual servoing: A switching approach," *IEEE Trans. on Robotics*, vol. 20, no. 5, pp. 534–549, Oct 2004.

[8] R. Schapire, "The boosting approach to the machine learning: An overview," in *MSRI Workshope on Nonlinear Estimation and Clasification*, 2001.

[9] J.-T. Laprest, F. Jurie, M. Dhome, and F. Chaumette, "An efficient method to compute the inverse jacobian matrix in visual servoing," in *IEEE Int. Conf. on Robotics and Automation, ICRA'04*, vol. 1, New Orleans, LA, April 2004, pp. 727–732.

[10] N. Mansard, M. Lopes, J. Santos-Victor, and F. Chaumette, "Jacobian learning methods for tasks sequencing in visual servoing," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'06*, Beijing, China, October 2006, pp. 4284–4290.

[11] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'94*, 1994.

[12] S. Hutchinson, G. Hager, and Cork, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 18–27, 1996.

[13] 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.

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

[15] D. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code," in *The Second European Conference on Computer Vision, ECCV '92*. London, UK: Springer-Verlag, 1992, pp. 335–343.

[16] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, December 2005.

[17] N. Gans, S. Hutchiinson, and P. Corcke, "Performance test for visual servo control systems, with application to partitioned approaches to visual servo control," *International Journal of Robotics Research*, vol. 22, no. 10-11, pp. 955–981, October-November 2003.