Maximizing the Use of Computational Resources in Multi-Camera Feedback Control

Henriksson, Dan; Olsson, Tomas

Published in:
Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS04)

DOI:
10.1109/RTTAS.2004.1317282

2004

Maximizing the Use of Computational Resources in Multi-Camera Feedback Control *

Dan Henriksson and Tomas Olsson
Department of Automatic Control
Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden
{dan, tolsson}@control.lth.se

Abstract

In vision-based feedback control systems, the time to obtain sensor information is usually non-negligible, and these systems thereby possess fundamentally different timing behavior compared to standard real-time control applications. For many image-based tracking algorithms, however, it is possible to trade-off the computational time versus the accuracy of the produced position/orientation estimates.

This paper presents a method for optimizing the use of computational resources in a multi-camera based positioning system. A simplified equation for the covariance of the position estimation error is calculated, which depends on the set of cameras used and the number of edge detection points in each image. An efficient algorithm for selection of a suitable subset of the available cameras is presented, which attempts to minimize the estimation covariance given a desired, pre-specified maximum input-output latency of the feedback control loop.

Simulations have been performed that capture the real-time properties of the vision-based tracking algorithm and the effects of the timing on the performance of the control system. The suggested strategy has been compared with heuristic algorithms, and it obtains large improvements in estimation accuracy and performance for objects both in free motion and under closed-loop position control.

1. Introduction

Over the last decade there has been a growing interest in vision-based control, where objects are being positioned using visual feedback from one or several cameras. Much information about an unknown scene can be obtained from visual data, and with the increasing computational power available today, there is a potential for robust visual feedback (visual servoing) systems operating at camera frame rate. In mobile and industrial robotics in particular, many guidance and positioning techniques using camera feedback have been developed. A good review of different visual servoing techniques is given in [5]. Using new techniques for visual feedback from multiple cameras, it is possible to obtain high positioning accuracy, even in uncalibrated environments [9]. Positioning using standard digital cameras is also a very cost-effective solution, compared to other techniques such as, e.g., laser scanning.

In the past, the inaccurate and noisy nature of visual measurements, and the complex and time-consuming computations involved, have restricted the achievable bandwidth of the controlled systems. Even today these are very important issues, especially in systems using feedback from multiple cameras. However, relatively little work has been performed to study the timing performance of visual servoing systems and its effect on the dynamic performance of control loops. One exception is [14], that investigates the effects of timing on tracking accuracy in a simple setup.

For real-time control applications in general, the importance of minimizing the input-output latency, i.e., the delay from the reading of the sensors to the generation of the control output, is well-known. Unless compensated for, the input-output latency will compromise the performance of the control system, and may even cause instability. In vision-based control systems the latency is usually dominated by the image processing. This paper presents a method for resource allocation in multi-camera visual servoing. The method aims at achieving a close to constant input-output latency and to maximize the obtained accuracy of the estimated vision-based feedback information during this bounded pre-specified time interval.

Many methods for rigid body tracking work by minimizing some measure of the image space error as a function

* The work has been sponsored by the Swedish Foundation for Strategic Research via the program FLEXCON, the Swedish Research Council, and by LUCAS — the VINNOVA-funded Center for Applied Software Research.

Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04)
1080-1812/04 $ 20.00 © 2004 IEEE

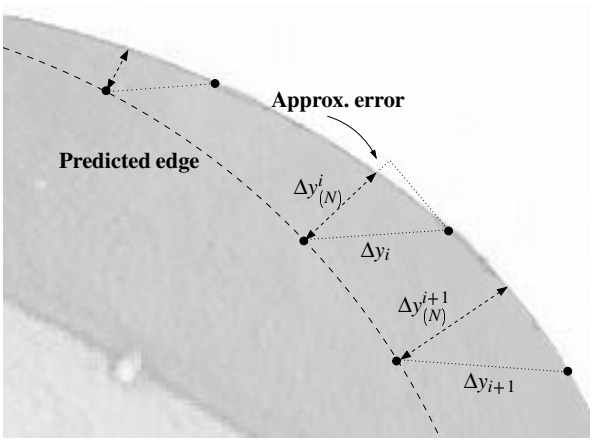

Figure 1. Edge detection in the normal direction of the predicted edges.

of the unknown position and orientation parameters, using standard non-linear optimization methods, or Kalman filtering techniques [7, 10]. The position and orientation can be parameterized in different ways, such as Euler- or roll-pitch-yaw angles. There are also various ways to measure the image space error, the most common measurements being the positions of point features [7], lines, or point-to-edge distances [3, 9]. The point-to-edge method has a major advantage in that it does not require the exact matching of features, only the distances in the normal direction from a number of *edge detection points* to the real edge, see Figure 1. This only requires a one-dimensional search in the image, reducing the computational complexity considerably [3].

It is standard practice in visual sensing to only search for the tracked object in a small region around the estimated position. Thus, when the object moves in an unpredictable way, e.g., by a disturbance acting on the system, the search has to be extended. This will lead to a significant increase in the execution time of the tracking algorithm. The potential schedulability problem of visual tracking algorithms resulting from these sudden execution time variations were treated in [2]. However, to circumvent this problem and to obtain a more or less predictable computational delay of the tracking algorithm, several cameras are being used. By using many cameras with different settings and distance to the tracked target, it is possible to reduce the likelihood of the object suddenly moving out of the tracking region.

The tracking method based on point-to-edge errors used in this paper is of anytime or *milestone* [8] nature, i.e., its computational requirement can be influenced by the number of edge detection points used. By having a fixed number of points it is possible to obtain a predictable input-output latency, that can be compensated for by the control algorithm. With a constant number of edge detection points, the on-line resource allocation problem is reduced to finding an optimal set of cameras and point distribution in every sample, i.e., to maximize the estimation quality by a proper choice of active cameras and distribution of the available edge detection points between these cameras. The camera selection algorithm proposed in this paper is based on successive minimization of the estimation covariance, by adding more cameras to the active camera set, until no further improvement of the estimation accuracy can be obtained.

The suggested algorithm is evaluated in a simulation study, using a setup of six cameras. The scheme is evaluated both in terms of estimation variance and control performance using a delay-compensating LQG-controller [1]. It is shown that the resource-optimizing algorithm yields higher performance than the heuristic camera choices of distributing all edge detection points in the best-placed camera, or distributing the points evenly between all cameras.

The rest of the paper is outlined as follows. Section 2 describes the tracking algorithm and its associated timing properties. The suggested resource allocation scheme is given in Section 3. The scheme is evaluated and compared with heuristic algorithms in a simulation study given in Section 4. Section 5 contains a discussion of the results, and the conclusions are given in Section 6.

2. The Tracking Algorithm

We assume that M cameras are placed in fixed locations, viewing a target object whose position and orientation with respect to some fixed (world) coordinate system should be estimated. The position and orientation is parameterized as an n-vector $\mathbf{x}$ where typically $n = 6$ or $n = 7$, depending on the selected parameterization of the orientation. The image data is compressed into a vector $\mathbf{y}$, usually the image space coordinates of corners, edges and other features. If the geometry of the target is known, $\mathbf{x}$ and $\mathbf{y}$ are related by the projection equations of the cameras

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{1}$$

which is usually a very complex non-linear function. The most commonly used camera model is the homogeneous form pinhole camera projection equation [12], which in our case becomes

$$\mathbf{y}_i = \mathbf{h}_i(\mathbf{x}) = \frac{1}{Z}\mathbf{K}\mathbf{T}_{cw}\mathbf{T}_{wo}(\mathbf{x})\mathbf{X}_i \tag{2}$$

where $\mathbf{K}$ is a matrix of internal camera parameters, $\mathbf{X}_i$ is the coordinates of the point in an object-centered coordinate system, Z is the depth of the point in the camera, and $\mathbf{T}_{wo}(\mathbf{x})$ and $\mathbf{T}_{cw}$ are the homogeneous coordinate transformation matrices between the target object and the world coordinate system, and between the world coordinate system

```
while (true) {
     read camera images;
     perform image pre-processing;
     for i = each edge detection point {
         dy[i] = edge distance;
     }
     x_est = x_est+invert(J)*dy;
     predict x_est one step ahead;
     determine visible features;
     place edge detection points;
     calculate Jacobians;
}
```

Figure 2. Algorithm for object tracking.

and the camera, respectively. The camera position $\mathbf{T}_{cw}$ is assumed to be known, while the parameterization $\mathbf{x}$ of $\mathbf{T}_{wo}$ is the unknown position/orientation to be estimated.

By stacking the projection equations for many object points $\mathbf{X}_i$ and their image projections $\mathbf{y}_i$ in all M cameras, we obtain Equation (1). In order to avoid the *correspondence problem* of matching a given image feature to a feature on the object model, the estimated position $\hat{\mathbf{x}}$ is usually updated iteratively, using the estimated value obtained in the last sample as a starting point. Equation (2) can be differentiated with respect to $\mathbf{x}$ and linearized around the previous estimate $\hat{\mathbf{x}}$. The linearized equations for N feature points can then be stacked to give

$$\Delta\mathbf{y} = \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}) = \mathbf{J}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}}) \tag{3}$$

where $\mathbf{J} = d\mathbf{h}/d\mathbf{x} \in \mathbb{R}^{2N\times n}$ is the *Jacobian* of the projection equation. Using this equation, the estimated position $\hat{\mathbf{x}}_k$ at sample k can be updated iteratively as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \mathbf{J}^{\dagger}(\hat{\mathbf{x}}_{k-1})\left(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_{k-1})\right) \tag{4}$$

where $\mathbf{J}^{\dagger}$ is the pseudo inverse of $\mathbf{J}$, given by

$$\mathbf{J}^{\dagger} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T. \tag{5}$$

In the case of point-to-edge measurements, only the distance between the predicted and real edge in the normal direction is measurable. This distance can be approximated well by the projection of the errors $\Delta\mathbf{y}$ in Equation (3) onto the normal directions, giving us the modified equation [3, 9]

$$\begin{aligned}\Delta\mathbf{y}_{(N)} &= \mathbf{N}^T(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})) = \mathbf{N}^T\mathbf{J}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}}) = \\ &= \mathbf{J}_{(N)}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})\end{aligned} \tag{6}$$

where $\mathbf{N}$ is a sparse $N \times 2N$ matrix having "diagonal" blocks representing the normal directions at the N different edge detection points along the edge, see Figure 1. $\mathbf{J}_{(N)} \in \mathbb{R}^{N\times n}$ is the new Jacobian for point-to-edge measurements. When using edge features, we have significant freedom in how to choose which features to measure, since any number of

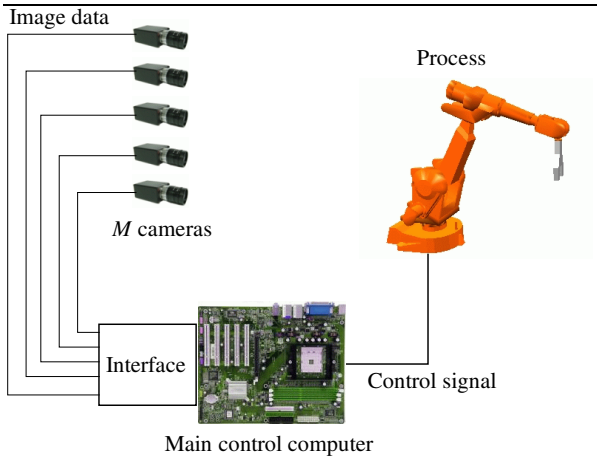

Figure 3. Hardware structure of the vision-based control system.

edge searches can be performed at any place along any object edge, in each camera. Typically, when using point-to-edge measurements, we will have $N \gg n$.

The algorithm for rigid body tracking is summarized in Figure 2. The image pre-processing step involves all image conversions and filtering necessary for each camera. The position at the next sample is predicted, and the prediction is used to determine where interesting image features will be visible next sample. Visible features are determined using a fast hidden-line removal algorithm based on a Binary Search Partitioning (BSP) tree representation of the object, see [13]. The BSP tree recursively divides the surfaces of the object into "in front" and "behind" giving a perfect front-to-back ordering. A large number of edge detection points are divided between the cameras and placed along the predicted edges of the object. The number of edge detection points is chosen based on the timing model below. Finally, the Jacobians for each camera are computed.

2.1. Timing

The assumed hardware structure of the vision-based control system is given in Figure 3. The M cameras are further assumed to have a synchronized image capture. Each camera node is connected to the main control computer via a standard bus, e.g., a IEEE-1394 (FireWire) or CameraLink interface, and is able to independently perform image capture, processing and data transfer. Typical data transmission rates range from between 400 MBit/s up to 3.2 GBit/s, while the data for each image is $320 \times 240 \times 2 = 150$ kB. This gives a total transmission time of between 0.4 and 3 ms. The main control computer will perform the necessary im-

age processing for each image, calculate the estimates and control signal, and update the data for next sample.

The total computation time required in each sample depends on the number of cameras, M, and the total number of edge detection points, N. The time required for preprocessing all images is proportional to the number of cameras used, whereas the total time used for finding edges, placing edge detection points, updating the estimation and building the Jacobians, is proportional to the total number of edge detection points. The total time T_{tot} from sampling the cameras until the new estimation is obtained can therefore be modeled by the equation

$$T_{tot} = T_0 + T_c M + T_f N \tag{7}$$

where T_0 is a constant time required for image capture and image data transfer for all cameras. The values of the time coefficients depend on many factors, such as camera sensor and interface type, camera shutter speed, platform, and implementation factors. The timing model of Equation (7) has been verified experimentally by measurements of actual computation and transmission times. Approximate values in our implementation have been determined to $T_0 = 6.0$ ms, $T_c = 1.0$ ms and $T_f = 0.01$ ms.

There are two main implications of the timing model in Equation (7). First, as the computation time is deterministic and roughly constant, for a given M and N, it can be compensated for by the control algorithm. Second, the relation between T_c and T_f shows the potential of gaining accuracy by switching cameras on and off, thereby allowing a larger total number of edge detection points. Assuming a desired input-output latency, T_{des}, and $M(k)$ active cameras at sample k, the number of edge detection points to distribute between the $M(k)$ cameras is given by

$$N(k) = \frac{T_{des} - T_o - T_c M(k)}{T_f} \tag{8}$$

The delay T_{des} is typically chosen in relation to the dynamics and closed-loop bandwidth of the control system. An exact analysis of the delay-sensitivity of a control system can, e.g., be performed using the JitterBug toolbox [6].

With a camera frame rate of 30 Hz, corresponding to a sample period of 33 ms, simple rules-of-thumb [1] give that a realistic closed-loop bandwidth should lie between 6 and 18 rad/s. A delay of 15 ms would then correspond to a phase loss of 5-15 degrees, which in most cases can be compensated for without too much performance loss. Using the estimated camera timing parameters in our implementation, $T_{des} = 15$ ms would correspond to a total of 300 edge detection points when using $M(k) = 6$ cameras, and 800 points when using only one camera.

Thus, depending on the complexity of the scene and the dynamics of the control system, the relation between computational delay and number of edge detection points can be chosen arbitrarily off-line. E.g., for certain robotics applications with highly non-linear dynamics it is non-trivial to compensate for input-output latency. In this case it may be beneficial to have a small number of detection points and a short delay. On the other hand, if the tracking scene is complex and a less delay-sensitive control system is used, it may be advantageous to instead use a larger total number of edge detection points.

2.2. Estimation Accuracy

It is clear that in general the accuracy of the estimation will improve with the number of image measurements N. If we assume that the errors in the image measurements $\Delta\mathbf{y}_{(N)}$ can be modeled by Gaussian, independent noise with variance σ^2, the covariance of the estimation error, $\tilde{\mathbf{x}} = (\mathbf{x} - \hat{\mathbf{x}})$, can be approximated as

$$\begin{aligned} E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] &= E[\mathbf{J}^{\dagger}_{(N)} \Delta\mathbf{y}_{(N)} (\mathbf{J}^{\dagger}_{(N)} \Delta\mathbf{y}_{(N)})^T] = \\ &= \sigma^2 (\mathbf{J}^T_{(N)} \mathbf{J}_{(N)})^{-1} = \sigma^2 (\sum_{i=1}^{M} \mathbf{J}_i^T \mathbf{J}_i)^{-1} \end{aligned} \tag{9}$$

where the Jacobian has been partitioned into M individual Jacobians for each camera as $\mathbf{J}^T_{(N)} = [\mathbf{J}_1^T \mathbf{J}_2^T \cdots \mathbf{J}_M^T]^T$.

The Jacobian for each camera is a function of the current position $\mathbf{x}$, as well as on the number of edge detection points, N_i, for that camera, and how they are distributed. If the points are distributed evenly along the visible edges of the object, we can use the approximation

$$\mathbf{J}_i^T \mathbf{J}_i \approx N_i \Phi_i(\mathbf{x}) \tag{10}$$

where Φ_i is a positive semidefinite $n \times n$ matrix independent of N_i. Using Equation (10) in Equation (9) we get

$$E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \sigma^2 \left(\sum_{i=1}^{M} N_i \Phi_i(\mathbf{x}) \right)^{-1} \tag{11}$$

which shows that the covariance of the estimation error is a direct function of the number, N_i, of edge detection points placed in each camera.

As can be seen from Equation (9), the estimation error is also a function of the object position $\mathbf{x}$. For one camera, $\Phi_i(\mathbf{x})$ is large and the resulting estimation error is small when the entire object is visible and close to the camera. Conversely, poorly conditioned situations occur when only part of the object is visible, or when the object is very far from the camera, where in extreme cases the problem could become ill-conditioned. A common example is when all visible image features lie on a straight line, causing rotations of the object around this line to become unobservable from the image feature data.

```
N_feat[N_cam] = number of possible edge detection points using N_cam cameras;
for i = all cameras
    Φ[i] = transpose(J[i])*J[i]/N[i];
clear selected set of cameras and best distribution of edge detection points;
set Φ_max = 0, best_val = 0, N_cam = 1, stop_flag = false;
while stop_flag == false {
    for k = all cameras not in currently selected set
        find the λ ∈ [0,1] which minimizes new_val[k] = inverse(λ*Φ_max+(1-λ)*Φ[k]);
    set best_new_cam = the camera cam with smallest new_val[cam];
    set best_new_ratio = the minimizing λ for camera best_new_cam;
    set Φ_new = best_new_ratio*Φ_max + (1-best_new_ratio)*Φ[best_new_cam];
    if (inverse(N_feat[N_cam]*Φ_new) < best_val) {
        update selected set by adding best_new_cam;
        redistribute edge detection points according to best_new_ratio;
        set Φ_max = Φ_new, best_val = inverse(N_feat[N_cam]*Φ_new);
        set N_cam = N_cam + 1;
    } else stop_flag = true;
}
```

Figure 4. Algorithm for selection of best camera set and edge detection point distribution.

3. Resource Allocation

When cameras are used for positioning, for instance in robotics, the cameras need to be distributed so that the entire workspace is covered. Because of the limited resolution and field of view of each camera, it is usually beneficial to place the cameras so that each camera covers only a part of the available workspace. Some cameras may be placed so that they cover a large part of the workspace, giving rough information on the location of the object, while other cameras cover only part of the workspace for a more accurate localization. If the object is moving, different cameras will give more or less useful or accurate information at different times, depending on the current object position.

In general, the most accurate estimation of the position is obtained when using a subset of the available cameras. When timing is important, for instance when the estimated position is to be used for feedback control, it would be an advantage to use only the 'best' subset of cameras. The reason is the extra processing time required for each camera, quantified by the parameter T_c in Equation (7). For a given total computational time, it is thus possible to use more edge detection points if using only the well-placed cameras.

Using the covariance of the estimation error as a measure of the estimation accuracy, we see from Equation (11) that for a given object position $\mathbf{x}$, we should choose the number of edge detection points, N_i, in each camera from a subset $\{C_k\}$ of all available cameras to minimize

$$E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \sigma^2 \left(\sum_{i \in \{C_k\}} N_i \Phi_i(\mathbf{x}) \right)^{-1} \tag{12}$$

where $\sum_{i \in \{C_k\}} N_i = N(k)$. In general, the covariance decreases with increasing N_i. From Equation (8) we see that the smaller we choose the set C_k, corresponding to a smaller $M(k)$, the more edge detection points can be distributed between the active cameras.

Finding the optimal camera set and edge detection point distribution from Equation (12) for a general $\mathbf{x}$ is non-trivial. Simple heuristic choices for the camera set, such as the best individual camera or all cameras, are possible, but can be arbitrarily far from the optimum. Additionally, using a single camera or a low number of cameras could cause the problem to become ill-conditioned, for instance in situations where only a small part of the object is visible in each camera.

3.1. Algorithm

Since timing is important, a fast algorithm for selecting a suitable camera set and feature distribution has been developed. The algorithm is outlined in Figure 4. The algorithm updates the active set of cameras and the distribution of edge detection points among the active cameras in every sample. This is done by successively adding new cameras and recomputing the distribution between edge detection points in the current active set and the added camera. If the covariance can be decreased, the active set and distributions are updated with the new camera. If there is nothing that can be gained by adding another camera, the algorithm stops and the current active set is used in the next sample.

The algorithm is very robust and easy to implement. It takes negligible time to execute, since all information about the relative accuracy of the cameras is contained in the small $n \times n$-matrices Φ_i. The algorithm will in general not achieve the optimum covariance, but will find a small subset of cameras which together give a significantly lower covariance than for the heuristic choices.

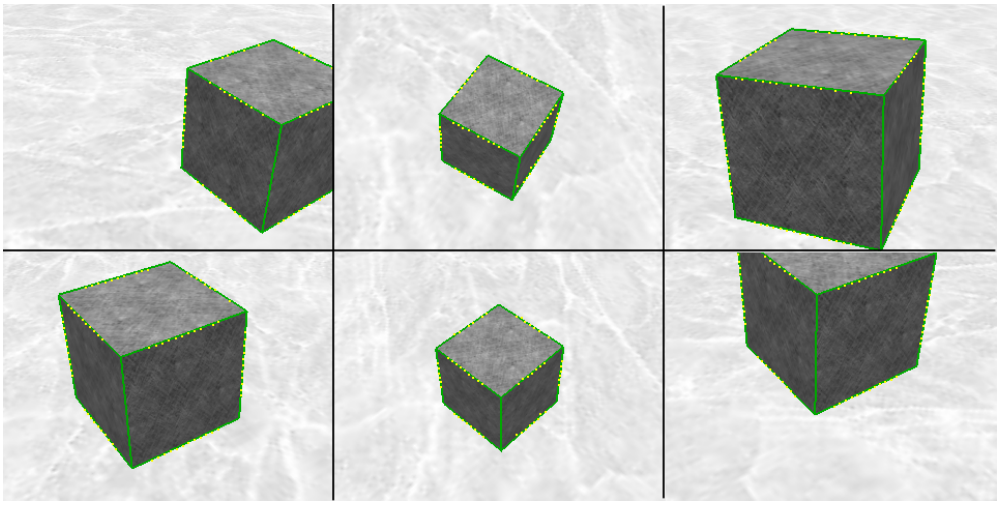

Figure 5. Example of six images from the simulated cameras with edge detection points indicated.

4. Simulations

The performance of the proposed resource allocation algorithm will be demonstrated by simulations. The simulations were performed using six cameras, where the images were generated using the standard graphics API OpenGL [11]. The object being tracked was a textured box of dimensions 18 cm×18 cm×18 cm which was moved around in front of a textured background. Figure 5 shows example images taken from a test sequence. We have assumed the timing model and values given in Section 2.1, the sampling period $h = 33$ ms, and a maximum desired control latency $T_{des} = 15$ ms.

4.1. Tracking Accuracy

The tracking accuracy for a stationary target was evaluated by measuring the estimation error variance for different image sequences, taken from different camera positions. Three different algorithms for resource allocation between the cameras were investigated:

1. Choosing the best camera, i.e. the camera for which Φ_i is 'largest', according to some chosen criterion.
2. Using all cameras, with equal distribution of edge detection points.
3. Choosing the best set of cameras, using the algorithm in Section 3.

The tracking accuracy was measured for image sequences taken with several different camera configurations as given by Table 1. The estimated standard deviations for the error in estimated orientation and translation are shown in Table 2. The estimation using the single-camera method did not converge for Sequence 2, since the problem becomes very poorly conditioned for any choice of a single camera. In Sequence 1, the minimum translation error was obtained by using all cameras, but the price is a significantly larger orientation error.

4.2. Control Performance

The visual feedback was applied in a feedback control setting, where the textured box was controlled one-dimensionally along the y-coordinate axis. The estimated y-position was used as feedback information to the controller. The simulated dynamics was described by a second order system, which after discretization [1] with the sampling interval $h = 33$ ms was given by the state-space model

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 1 & 0.033 \\ 0 & 0.97 \end{bmatrix} x(k) + \begin{bmatrix} 0.55 \\ 32.8 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{aligned} \tag{13}$$

The controller was a delay-compensating LQG-controller [1], designed to maximize the continuous-time function

$$J = \lim_{T\to\infty} \frac{1}{T} \int_0^T \left(\begin{bmatrix} x(t) & u(t) \end{bmatrix} Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \right) dt \tag{14}$$

with

$$Q = \mathrm{diag}(100, 0.1, 0.001) \tag{15}$$

As seen by the process model in Equation (13), only the first state is measurable. Therefore an observer [1] was designed to reconstruct the state vector. The state and output noise variances used in the design of the observer where chosen as $R_1 = 10$ and $R_2 = 0.01$, respectively.

Seq.	Camera 1		Camera 2		Camera 3		Camera 4		Camera 5		Camera 6	
	z[mm]	Vis.%	z[mm]	Vis.%	z[mm]	Vis.%	z[mm]	Vis.%	z[mm]	Vis.%	z[mm]	Vis.%
1	840	100	840	100	880	100	890	100	870	100	910	100
2	320	20	3800	100	3900	100	445	20	4700	100	4400	100
3	510	100	4500	100	5000	100	500	0	6500	100	7000	100
4	550	30	360	10	330	5	450	10	380	5	280	5
5	600	100	600	100	400	0	600	100	1000	0	1000	0
6	650	100	280	30	450	0	700	0	1000	0	1000	0
7	300	30	330	35	500	70	700	0	1000	0	1000	0
8	400	30	350	30	280	20	400	40	370	15	300	25

Table 1. Camera configurations, distances z to target and part of object visible in each camera.

#	Orientation error [°]			Translation error [mm]		
	Single	All	Sel.	Single	All	Sel.
1	0.18	0.36	0.18	0.75	0.49	0.68
2	∞	0.49	0.17	∞	0.37	0.25
3	0.12	1.23	0.10	0.26	1.64	0.21
4	0.22	0.20	0.16	0.97	0.28	0.26
5	0.12	0.29	0.10	0.35	0.36	0.31
6	0.14	0.20	0.08	0.36	0.46	0.17
7	0.22	0.25	0.07	1.10	0.28	0.16
8	0.08	0.09	0.05	0.24	0.14	0.12

Table 2. Tracking error standard deviations, orientation and translation, when using the best single camera, all cameras and the best selection of cameras, respectively.

The real-time simulations of the control system were performed in MATLAB/Simulink using the TrueTime simulator [4]. This simulator allows for co-simulation of continuous plant dynamics and discrete controllers implemented as tasks in a computer. The simulation environment communicated with the 3D-visualization environment over TCP.

The true position of the continuous-time plant model was used to update the camera images. The controller was implemented as a periodic task with the sampling interval 33 ms. In the beginning of the sample, only the images of the cameras in the active set were read and processed. The position estimate was then fed into the LQG-control algorithm, after which the computed control signal was actuated. After the control signal actuation, the remaining images were read and the resource allocation algorithm was executed to obtain the camera set and distribution of edge detection points for next sample.

The simulation camera setup corresponded to scenario 4 in Table 1, and the objective of the control was steady-state regulation of the position around $y = 0$. Simulation results are shown in Figure 6, showing the control signal and the controlled position variable. It is seen that the suggested resource allocation algorithm results in better control performance than both heuristic camera choices.

5. Discussion

The heuristic selection of the best single camera will fail in many common configurations such as the one in Sequence 2, where no single camera gives enough information. Using all cameras works well in most situations, but is rarely necessary, and may not even be feasible if there is a large number of available cameras. The best selection algorithm will find a small set, typically consisting of 1–3 cameras, that will give sufficient information to accurately and robustly estimate the position and orientation. It also scales well with the number of available cameras.

There is usually a spatial correlation between the measurements, making the assumed model of the image noise as independent and Gaussian unrealistic. However, the resource allocation algorithm does still give good results. The timing model in Equation (7) and the expression for the covariance given from Equation (10) are also approximations, since not all N edge searches will result in an edge being found. Thus, the computation time of Equation (7) is a function of the number of searches, rather than the number of obtained measurements. This could be compensated for by including the ratios of successful edge detections in the equations, assuming these to be independent of N_i.

Since not all image searches will be able to find a clear single edge, due to, e.g., occlusions, specular reflections, noise, and object texturing, the image data is needed in order to correctly calculate the matrices Φ_i and the Jacobians $\mathbf{J}_i$. Therefore, the pre-processing and edge detection steps need to be performed for every image in every sample. However, for the cameras not used in the estimation, these steps are performed *after* the estimated position has been obtained, and will thus not affect the control delay.

The strategy proposed in this paper is based on a control design compensating for a constant input-output latency. However, an alternative approach would be to exploit the anytime nature of the tracking algorithm, and dynamically change the computational requirement by the allocation of edge detection points. This approach could be used for dynamic scheduling of several vision-based control tasks to optimize control performance under resource constraints.

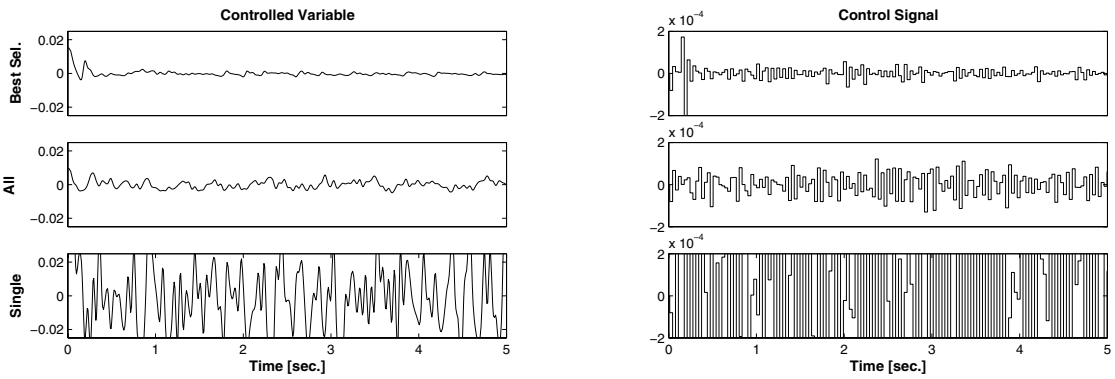

Figure 6. Closed-loop position control using camera setup scenario #4 of Table 1. The plots to the left show the controlled output and the plots to the right the control signal for the three tracking algorithms described in Section 4.1. The resource allocation algorithm gives the lowest steady-state variance. Using only one camera gives bad performance, which corresponds well with the results in Table 2.

6. Conclusions

This paper has presented a method for dynamic resource allocation in multi-camera based feedback control. It is shown how the covariance of the position estimation error depends on the set of cameras used and the number of edge detection points in each camera image. These parameters in turn affect the timing of the tracking algorithm and thus the input-output latency of the feedback control loop.

The objective of the resource allocation algorithm was to minimize the variance of the position estimate by a proper choice of active cameras and point distribution between these cameras. The total number of edge detection points was chosen to obtain a desired input-output latency.

Simulations were conducted that capture the real-time behavior of the tracking algorithm and its effect on closed-loop position control. It was shown that the resource allocation scheme significantly outperformed heuristic choices of using only the best-placed camera, and distributing all edge detection points evenly between all cameras. The algorithm was evaluated both in terms of tracking accuracy and closed-loop control performance.

References

[1] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems*. Prentice Hall, 1997.

[2] M. Caccamo, G. Buttazzo, and L. Sha. Elastic feedback control. In *Proc. of the 12th Euromicro Conf. on Real-Time Systems*, pages 121–128, Stockholm, Sweden, June 2000.

[3] T. Drummond and R. Cipolla. Real-time tracking of complex structures with on-line camera calibration. In *British Machine Vision Conference*, volume 2, pages 574–583, 1999.

[4] D. Henriksson, A. Cervin, and K.-E. Årzén. TrueTime: Simulation of control loops under shared computer resources. In *Proceedings of the 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, July 2002.

[5] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651 – 670, October 1996.

[6] B. Lincoln and A. Cervin. Jitterbug: A tool for analysis of real-time control performance. In *Proc. of the 41st IEEE Conf. on Decision and Control*, Las Vegas, NV, Dec. 2002.

[7] V. Lippiello, B. Siciliano, and L. Villani. Objects motion estimation via BSP tree modeling and Kalman filtering of stereo images. In *IEEE Int. Conference on Robotics and Automation*, pages 2968–2973, Washington D.C., 2002.

[8] J. Liu, K.-J. Lin, W.-K. Shih, A. Yu, J.-Y. Chung, and W. Zhao. Algorithms for scheduling imprecise computations. *IEEE Trans on Computers*, 1991.

[9] F. Martin and R. Horaud. Multiple camera tracking of rigid objects. *International Journal of Robotics Research*, 21(2):97–113, February 2002.

[10] T. Olsson, J. Bengtsson, A. Robertsson, and R. Johansson. Visual position tracking using dual quaternions with hand-eye motion constraints. In *IEEE Int. Conf. on Robotics and Automation*, pages 3491–3496, Taipei, Taiwan, Sept. 2003.

[11] OpenGL. "The Industry's Foundation for High Performance Graphics" Home page, http://www.opengl.org/, 2003.

[12] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, New Jersey, 1998.

[13] A. van Dam, L. Foley, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Boston, MA, 2nd edition, 1991.

[14] M. Vincze. Dynamics and system performance of visual servoing. In *Proceedings of the IEEE Conference on Robotics and Automation*, San Francisco, CA, April 2000.