2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

# Simple and Robust Visual Servo Control of Robot Arms Using an On-Line Trajectory Generator

Torsten Kröger and Jose Padial

*Abstract*— **Common visual servoing methods use image features to define a signal error in the feedback loops of robot motion controllers. This paper suggests a new visual servo control scheme that uses an on-line trajectory generator as an intermediate layer between image processing algorithms and robot motion controllers. The motion generation algorithm is capable of computing an entire trajectory from an arbitrary initial state of motion within one servo control cycle (typically one millisecond or less). This algorithm is fed with desired pose and velocity signals that are generated by an image processing algorithm. The advantages of this new architecture are: *(a)* jerk-limited and continuous motions are guaranteed independently of image processing signals, *(b)* kinematic motion constraints as well as physical and/or artificial workspace limits can be directly considered, and *(c)* the system can instantaneously and safely react to sensor failures (e.g., if cameras are covered or image processing fails). Real-world experimental results using a seven-joint robot arm are presented to underline the relevance for the field of robust sensor-guided robot motion control.**

## I. Introduction

The development of manipulation capabilities for collision-free motion is key for robot manipulators to safely accomplish useful tasks in human environments. Realizing such systems is difficult because manipulation tasks in dynamic environments require the integration of sensors (force/torque, tactile, vision, distance, etc.) in high-level planning systems as well as in inner robot control loops. Safety concepts have to be investigated in order to guarantee stable, safe, and robust robot motions even in the presence of sensor failures.

Visual servo control is a category of robotic control in which robust safety concepts must be developed. Such controllers use image features to define a signal error in the feedback loops of robot motion controllers. In real-world robot applications, problems may appear because of dirty lenses, inappropriate and/or abruptly changing light conditions, objects and/or humans covering the scene. In such situations, a safe and deterministic robot motion behavior is required in order to meet safety requirements.

In former works [1], [2], a class of on-line trajectory generation (OTG) algorithms was presented that has become part of the Reflexxes Motion Libraries [3], which compute robot motion trajectories from arbitrary initial states of motion within one servo control cycle, such that robots can react instantaneously to unforseen sensor signals and events.

This paper suggests a new control scheme using these algorithms as an interface between image processing and low-level robot motion control. The new system consists of three layers:

**(1)** A digital image processing algorithm used to compute a desired pose and a desired velocity vector for the robot.
**(2)** Both vectors are used by the OTG algorithm to *instantaneously* compute a motion trajectory that connects to the current state of motion.
**(3)** The output signals of the OTG algorithm are used by a trajectory-following motion controller.

Due to the intermediate layer, a number of advantages are achieved:

- Jerk-limited and continuous motions are guaranteed independently of image processing signals.
- Acceleration and velocity constraints due to limited dynamic robot capabilities can be *directly* considered.
- Physical and/or artificial workspace limits can be explicitly applied.
- In cases of sensor failures or inappropriate image processing results, deterministic and safe reactions and continuous robot motions are guaranteed.
- The image processing hard- and software does *not* necessarily have to be real-time capable.
- High performance due to low latencies, because motion trajectories are computed within one low-level control cycle (typically one millisecond or less).
- The proposed architecture is of a very simple nature and can be integrated in many existing robot motion control systems.

Before the new architecture is described, Sec. II reviews related works about visual servo control and on-line motion generation, and Sec. III summarizes the used concept of on-line trajectory generation. To underline the mentioned advantages, real world experimental results achieved with a *KUKA Light-Weight Robot IV* are presented in Sec. V.

## II. Related Works

The intention of this paper is to propose a new robot motion control architecture for visual servoing using an on-line trajectory generator. Related works of both fields are reviewed in this section.

### A. Visual Servo Control for Robotic Systems

If computer vision data is used for robot motion control, we speak about *visual servo control*. Required image processing and computer vision methods are described in many textbooks (e.g., [4]–[6]); basic overviews about visual servo

T. Kröger and J. Padial are with the Artificial Intelligence Laboratory at Stanford University, Stanford, CA 94305-9010, USA, tkr@stanford.edu, jpadial@stanford.edu.

control were presented by Chaumette and Hutchinson [7]–[9]. The origins for most works on visual servo control can be found in the publications of Weiss *et al.* [10] and of Feddema *et al.* [11]. In general, such control concepts can be separated into image-based visual servoing (IBVS) and position-based visual servoing (PBVS). IBVS approaches use an error signal that is measured in the image and subsequently mapped to actuator commands. PBVS approaches estimate the camera position based on image features. The resulting position is used to compute a Cartesian position error that is subsequently used by the underlying motion controller.

The approach of Gans *et al.* [12], [13] suggests two visual servo controllers as submodules in a hybrid switched-system. Assuming an eye-in-hand camera setup, the first control submodule uses PBVS and the second one IBVS. Stability is proven by means of a state-based switching scheme. Deng *et al.* [14] suggest a similar approach, in which an artificial potential field is used to generate robot trajectories. Recent works [15], [16] also focus on visual servo control approaches for mobile robot manipulation platforms. Jeong *et al.* [17] focus on a Kalman filter based PBVS approach to improve the robustness against outliers.

Mostly related to this paper is the work of Chesi [18], who introduced an on-line *path* planning method based on linear matrix inequalities to take into account positional system constraints such as workspace and joint limits. The method presented here is of much simpler nature. Furthermore, the approach proposed in this paper can directly take into account kinematic motion constraints (max. velocity, max. acceleration, and max. jerk).

### B. On-Line Trajectory Generation in Robot Motion Control

The works mostly related to this paper are [19]–[23]. Broquère *et al.* [19], [20] published a method that uses an on-line trajectory generator for an arbitrary number of independently acting degrees of freedom (DOF). The approach is very similar to the one of Liu [21] and is based on the classic seven-segment acceleration profile [24]. With regard to [2], it is a Type V on-line trajectory generation approach designed for handling several DOFs individually. Real-world results are presented and described at [25].

The work of Haschke *et al.* [22] presents an on-line trajectory planner in the very same sense as [1], [2] do. The proposed algorithm generates jerk-limited trajectories from arbitrary states of motion, but suffers from numerical stability problems that may prevent a jerk-limited trajectory from being calculated. In such a case, a second-order trajectory with infinite jerks is calculated. Furthermore, the algorithm only allows for the specification of target velocities of zero. Ahn *et al.* [23] proposed a work for the on-line calculation of one-dimensional motion trajectories for any given state of motion *and* with arbitrary target states of motion, that is, with target velocities and target accelerations unequal to zero. The major drawback of this work is that no kinematic motion constraints, such as maximum velocity, acceleration, and jerk values, can be specified.

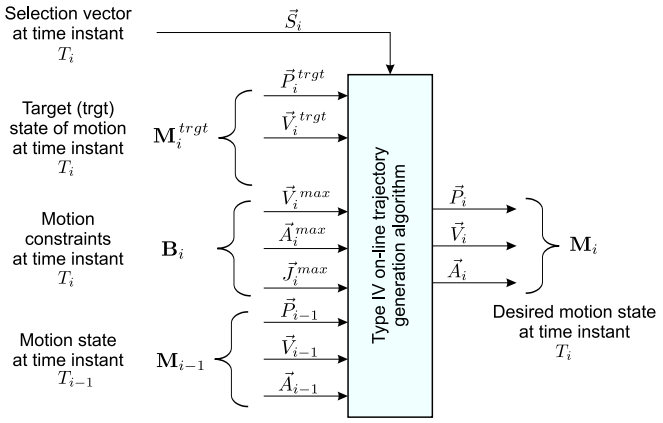


Fig. 1. Input and output values of the Type IV OTG algorithm for multiple DOFs (cf. [1], [2] and Fig. 2).

## III. On-Line Trajectory Generation

An OTG algorithm can be considered as a state-feedback position or pose controller using the current state of motion for command variable generation [1], [2]. We consider PC- or micro-controller-based systems for robot motion control and assume a time-discrete overall system with a cycle time of $T^{cycle}$. A state of motion at an instant $T_i$ is represented by the matrix

$$\begin{aligned} \mathbf{M}_i &= \left(\vec{P}_i,\, \vec{V}_i,\, \vec{A}_i\right) \\ &= \left({}_1\vec{M}_i,\, \ldots,\, {}_k\vec{M}_i,\, \ldots,\, {}_K\vec{M}_i\right)^T \quad , \end{aligned} \tag{1}$$

where $\vec{P}_i$ contains the position, $\vec{V}_i$ the velocity, $\vec{A}_i$ the acceleration, and ${}_k\vec{M}_i$ represents the state of motion of degree of freedom $k$ in a system with $K$ degrees of freedom (DOF) at instant $T_i$. In this paper, we make use of the Type IV OTG algorithm, whose input and output values are illustrated in Fig. 1. The OTG algorithm computes a time-optimal motion trajectory from an arbitrary initial state of motion $\mathbf{M}_{i-1}$ to a desired target state of motion

$$\begin{aligned} \mathbf{M}_i^{trgt} &= \left(\vec{P}_i^{\,trgt},\, \vec{V}_i^{\,trgt},\, \vec{0}\right) \\ &= \left({}_1\vec{M}_i^{\,trgt},\, \ldots,\, {}_k\vec{M}_i^{\,trgt},\, \ldots,\, {}_K\vec{M}_i^{\,trgt}\right)^T \end{aligned} \tag{2}$$

under consideration of the kinematic motion constraints

$$\mathbf{B}_i = \left(\vec{V}_i^{\,max},\, \vec{A}_i^{\,max},\, \vec{J}_i^{\,max}\right) \quad . \tag{3}$$

The Type IV OTG algorithm only allows for the specification of $\vec{P}_i^{\,trgt}$ and $\vec{V}_i^{\,trgt}$. The target acceleration vector $\vec{A}_i^{\,trgt}$ is always zero (cf. eqn. (2)). Depending on $\mathbf{M}_{i-1}$, an acceleration profile is selected from a finite set of profiles. Based on this profile, a system of nonlinear equations is set up [1], [2], whose solution contains all trajectory parameters to transfer the system from its current state of motion $\mathbf{M}_{i-1}$ to a desired target state of motion $\mathbf{M}_i^{trgt}$ while considering given kinematic motion constraints $\mathbf{B}_i$. Each system of equations features a proper input domain, for which a valid solution can be found. For this class of algorithms, it is essential that

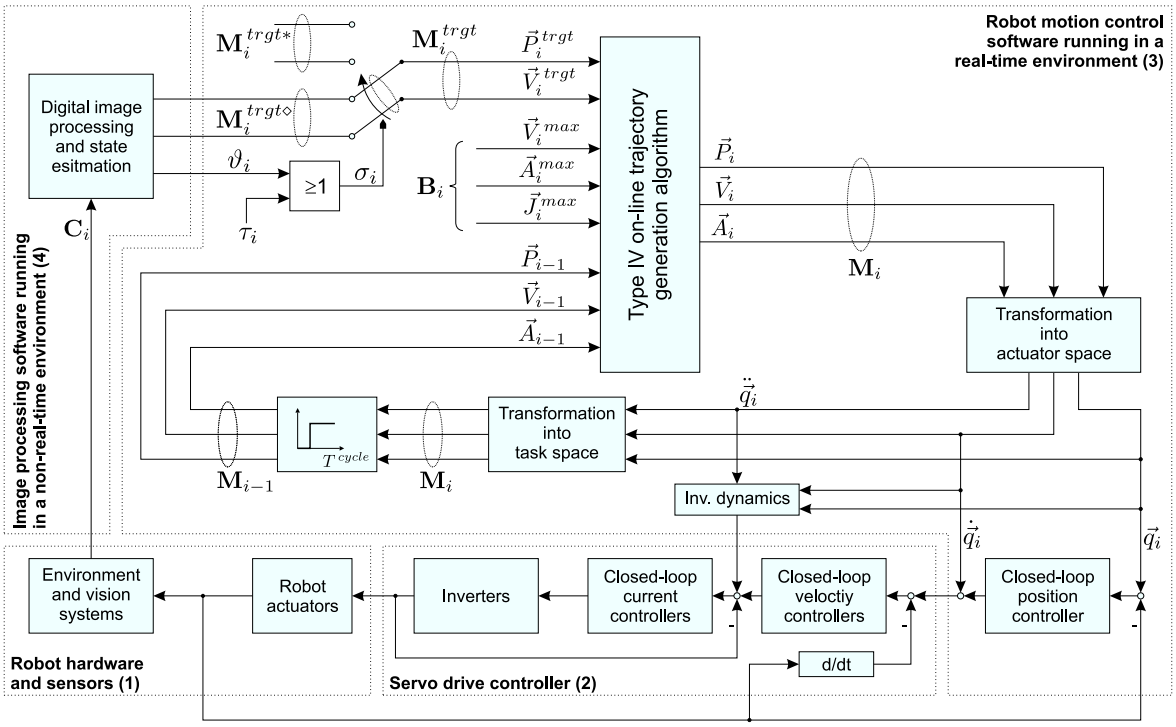


Fig. 2. Proposed visual servo robot control scheme using the Type IV on-line trajectory generation algorithm of Fig. 1.

*the union of all input domains equals the entire input domain* of the algorithm.

The Boolean selection vector $\vec{S}_i$ may be used to mask single DOFs that do not have to be considered by the OTG algorithm; in this paper, we assume $\vec{S}_i = \vec{1} \; \forall \; i \in \mathbb{Z}$. The algorithm is periodically executed every control cycle $T_i$ with $i \in \mathbb{Z}$, and its output values consist of the desired state of motion $\mathbf{M}_i$ for the current control cycle. Internally, a time-synchronized trajectory to transfer the current state of motion $\mathbf{M}_{i-1}$ to the target state of motion $\mathbf{M}_i^{trgt}$ under consideration of the constraints $\mathbf{B}_i$ in the shortest possible time (i.e., time-optimally) is generated.

For a robot system with six DOFs, the worst-case execution time of the Type IV OTG algorithm is $490\,\mu s$ on an *Intel Core i7-930 CPU*; the average execution time is $95\,\mu s$.

How this highly reactive motion generation algorithm is used in visual servo control schemes, is described in the next section.

## IV. Visual Servo Control

### A. Control Scheme

In the proposed robot motion control scheme, an OTG algorithm is used as an intermediate layer between digital image processing and robot motion control algorithms in general. In order to keep it simple, we consider a PBVS algorithm and a classical position control scheme for serial robots, such that the approach can be directly applied to common industrial robots. Figure 2 shows the scheme and separates into in four parts: **(1)** robot hardware with sensors and cameras, **(2)** servo drive controllers, **(3)** robot motion control software, and **(4)** an image processing software.

A state in actuator space at instant $T_i$ is represented by the actuator positions $\vec{q}_i$ and its derivatives $\dot{\vec{q}}_i$ and $\ddot{\vec{q}}_i$. The output of the OTG algorithm is a Cartesian state of motion $\mathbf{M}_i$ that is transformed into actuator space by an unambiguous transformation method (in the most simple case an inverse kinematic model and the inverse Jacobians [26]). In actuator space, a classic cascaded position control scheme is applied [27], and an inverse dynamic model is used for linearization [28]. In the same cycle, a kinematic forward transformation is applied to $\left(\vec{q}_i,\, \dot{\vec{q}}_i,\, \ddot{\vec{q}}_i\right)$, such that $\mathbf{M}_i$ is obtained again.

The approach can also be applied in other control schemes; for instance, $\mathbf{M}_i$ could be applied in the task space of an on operational space controller [29], which is part of a whole-body controller [30], and which maps the desired state of motion to force and/or torque values in actuator space.

**Please note** that the proposed control scheme consists of two loops: **(1)** The internal state feedback loop of the on-line trajectory generator using $\mathbf{M}_i$, and **(2)** the camera feedback loop for closed-loop visual servo control using the camera signals $\mathbf{C}_i$. As a result, the actual visual servo control feedback loop is closed by $\mathbf{C}_i$, and the OTG algorithm only acts as a open-loop controller that takes into account the capabilities of the robot $\mathbf{B}_i$ in order to guarantee a smooth and executable trajectory for the underlying controllers.

### B. The Role of On-Line Trajectory Generation

An important aspect of the scheme shown in Fig. 2 is that the state feedback loop for the OTG module bypasses the plant, that is, the output of the cycle $T_i$, $\mathbf{M}_i$ is the input of the next control cycle $T_{i+1}$. The feedback loop is closed only through the cameras, such that the vision signals are part of

the feedback loop, and the OTG module acts as a filter that directly considers the kinematic motion constraints $\mathbf{B}_i$ (cf. eqn. (3)). Independent of $\mathbf{M}_i^{trgt}$, the resulting state motion $\mathbf{M}_i$ has the following properties w.r.t. $\mathbf{M}_{i-1}$ and $\mathbf{B}_i$:

$$\begin{aligned}
&\forall\, k \;\in\; \{1, \ldots, K\} : \qquad (4)\\
&|{}_kP_i \;-\; {}_kP_{i-1}| \leq {}_kV_{i-1}\; T^{\,cycle} \;+\; \tfrac{1}{2}\,{}_kA_{i-1}\;\left(T^{\,cycle}\right)^2\\
&\qquad\qquad \pm\, \tfrac{1}{6}\,{}_kJ_i^{max}\;\left(T^{\,cycle}\right)^3 \;\wedge\\
&|{}_kV_i \;-\; {}_kV_{i-1}| \leq {}_kA_{i-1}\; T^{\,cycle} \pm\; \tfrac{1}{2}\,{}_kJ_i^{max}\;\left(T^{\,cycle}\right)^2 \;\wedge\\
&|{}_kA_i \;-\; {}_kA_{i-1}| \leq {}_kJ_i^{max}\; T^{\,cycle} \;\wedge\\
&\qquad |{}_kV_i| \leq {}_kV_i^{\,max} \qquad \wedge \qquad |{}_kA_i| \;\leq\; {}_kA_i^{\,max}\;.
\end{aligned}$$

The important aspect of these properties is that the resulting motion trajectory will never exceed the values of $\mathbf{B}_i$ independently of how the values $\mathbf{M}_i^{trgt}$ behave. In fact, $\mathbf{M}_i^{trgt}$ can even be a set of unsteady signals that arbitrarily jump at unforeseen instants.

### C. Image Processing and Robustness

The desired state of motion for the robot that results from the image processing algorithm is represented by $\mathbf{M}_i^{trgt\diamond}$. While the robot motion control environment runs at a rate of $f^{\,cycle} = 1/T^{\,cycle}$, the image processing system can run at a rate $f^{\,vision} \ll f^{\,cycle}$. In the most simple case, the image processing algorithm detects features in the current image $\mathbf{C}_i$, which can be transformed to a position in Cartesian space, such that a desired state of motion

$$\mathbf{M}_i^{trgt\diamond} \;=\; \left(\vec{P}_i^{\,trgt\diamond},\, \vec{V}_i^{\,trgt\diamond},\, \vec{0}\right) \qquad (5)$$

results, where $\vec{V}_i^{\,trgt\diamond} = \vec{0}$ holds. In more advanced systems that also consider velocities of detected object features, a desired Cartesian velocity vector $\vec{V}_i^{\,trgt\diamond}$ can be estimated, which will be reached exactly in $\vec{P}_i^{\,trgt\diamond}$.

To achieve a high robustness against

- outliers,
- insufficient image quality (e.g., due to spots on lenses),
- obstacles covering the camera,
- camera failures, and
- misbehavior of the image processing system,

a safe state of motion $\mathbf{M}_i^{trgt*}$ can be defined. The values of $\mathbf{M}_i^{trgt*}$ can be either predefined or dependent on the system state (e.g., dependent on the current state of motion or on further sensor signals); in the most simple case, $\mathbf{M}_i^{trgt*}$ only contains a safe target position vector $\vec{P}_i^{\,trgt*}$ to move the robot to a safe backup position.

Whether the output of the image processing algorithm $\mathbf{M}_i^{trgt\diamond}$ or the safe state $\mathbf{M}_i^{trgt*}$ is applied to the robot, depends on the Boolean switching variable $\sigma_i$:

$$\mathbf{M}_i^{trgt} \;=\; \begin{cases} \mathbf{M}_i^{trgt*} & \text{if } \sigma_i \;=\; 1 \\ \mathbf{M}_i^{trgt\diamond} & \text{otherwise .} \end{cases} \qquad (6)$$

$\sigma_i$ is defined as

$$\sigma_i \;=\; \tau_i \;\vee\; \vartheta_i\;, \qquad (7)$$

where $\vartheta_i$ is a Boolean value provided by the image processing module. If $\mathbf{M}_i^{trgt\diamond}$ is valid, $\vartheta_i$ is set to one; if no valid values can be calculated for $\mathbf{M}_i^{trgt\diamond}$, $\vartheta_i$ is set to zero (e.g., an obstacle or a human could cover the region of interest). To achieve the highest possible robustness, the Boolean variable $\tau_i$ is introduced; $\tau_i$ triggered by a timer that fires if the the image processing algorithm does not respond within a certain time interval.

### D. Physical and Artificial Workspace Limitations

In order not to overload Fig. 2, the feature of workspace limitation was not embedded in the control scheme. This feature requires that the target position is in the robot's workspace, that is,

$$\vec{P}_i^{\,min} \;\leq\; \vec{P}_i^{\,trgt\diamond} \;\leq\; \vec{P}_i^{\,max}\;, \qquad (8)$$

which can be achieved by a simple limiter function between the image processing algorithm and the OTG module. $\vec{P}_i^{\,min}$ and $\vec{P}_i^{\,max}$ represent the physical and/or artificial workspace limits. Equation (8) only holds for the case of $\vec{V}_i^{\,trgt\diamond} = \vec{0}$. The handling of cases, in which $\vec{V}_i^{\,trgt\diamond}$ is different from zero, is future work and described in Sec. VI.

### E. Real-Time Aspects

While the robot motion control software of Fig. 2 runs in a real-time environment (i.e., the *worst-case execution time* of all algorithms is less than $T^{\,cycle}$), the image processing software does not necessarily need to be real-time capable. It is fully sufficient if the *average execution time* of all algorithms is less than $T^{\,vision}$. Even if the execution time of the computer vision algorithms that calculate $\mathbf{M}_i^{trgt\diamond}$ significantly exceeds $T^{\,vision}$, the Boolean variable $\tau_i$ acts as a watchdog, such that the robot system can *always* be kept stable as a valid and jerk-limited trajectory is generated in any case (cf. eqn. (7)).

### F. Summary

The key part of the proposed visual servo control scheme is an on-line trajectory generator that generates motion trajectories from arbitrary states of motion within one low-level control cycle. This module is fed by an image processing algorithm and can be either considered a state-feedback controller or a filter, which can explicitly interpret kinematic motion constraints $\mathbf{B}_i$ of robot systems (cf. eqn. (3)). How the new control scheme behave on a physical robot system, and how it robustly reacts to sensor failures, is described in the next section.

## V. Real-World Experimental Results

The overall goal of the experiments was to demonstrate the efficacy of the control scheme in producing safe and smooth manipulator motion with noisy and erratic visual tracking measurements. Toward this goal, we validated the control method by having the manipulator end-effector follow a ball held by a human who moved the ball in any direction and

in an erratic manner. The ball was observed by a stereo camera system not moving with the manipulator, and the vision system did not measure the manipulator configuration or end-effector pose. As such, this experiment seemingly does not fit the mold of a canonical visual servo application where a loop is closed around error that is directly measured by the visual system. However, this new control scheme is directly extensible to an eye-in-hand vision system, and with an eye-in-hand vision system this does fit the canonical visual servo application. The only difference in the control scheme would reside in the front-end visual processing system. We designed our experiments to verify the control scheme while avoiding the added complexity of mounting an eye-in-hand visual system, as this vision technology is well understood and thoroughly researched.

A *KUKA Light-Weight Robot IV* [31], [32] was controlled through the *Fast Research Interface* [33], [34] at a rate of 1 KHz. The focus of our experiments was on the control scheme of Fig. 2, and the vision system was implemented with focus on simplicity and overall computational efficiency. We accepted high noise and inaccuracies in the estimated values of $\mathbf{M}_i^{trgt\diamond}$ for increased computational speed and ease of implementation. The time lag from image capture to manipulator reaction depends on the kinematic motion constraints $\mathbf{B}_i$, that is, high values of $\mathbf{B}_i$ lead to lower lags and less smooth motions, and vice versa.

The vision hardware consisted of a pair of *IDS UI-5220SE-C* camera systems [35] running at a frame rate of ≈60 Hz. Stereo calibration was accomplished by standard Bouguet code [36], and was left coarse. Further, the stereo cameras were calibrated with respect to the manipulator base frame by estimating the rotation matrix from the camera frame to the robot base frame. This estimation was accomplished by a linear least squares solution of the overdetermined system formed from measurements of the manipulator Cartesian end-effector position in several configurations, as measured from two sources: *(1)* derived from manipulator encoders, and *(2)* measured by the vision system.

Our tracked target was a small, solid color ball. For simplicity, we tracked Cartesian position solely, with no consideration of target orientation. We tracked the target by segmenting the original images via a standard backprojection method using a saved color histogram of the ball [37]. The segmented image was further eroded in order to cleanly segment the target from its surroundings, and the ball center was crudely estimated as the mass center of the nonzero pixels in the eroded image. The left image of Fig. 3 shows an eroded image and is indicative of the level of shape distortion that the ball image underwent in the segmentation and erosion processes. This distortion induced small errors in the estimated ball center in pixel space, which led to larger errors after stereo triangulation. The right image of Fig. 3 shows the target estimate of the ball position in red over the (stereo) rectified color image.

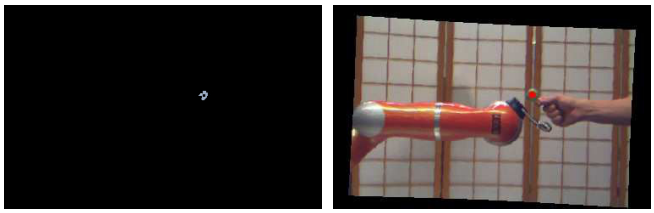

Fig. 3. Images from vision system. **Left:** Segmented and eroded image. **Right:** Rectified image with target position estimate overlayed in red.

The target position estimate was noisy, especially in the camera z-direction (stereo depth direction). The source of this noise can be directly attributed to the coarse stereo calibration, the error from the least-squares estimate of the camera frame to robot frame rotation matrix, the error from the simple segmentation method, and the error due to erosion and the simplistic mass center estimate. However, this noise was a feature for our experiments, as it more strongly demonstrated the robustness and safety of the trajectory generation system given noisy target measurements.

The target velocity was estimated from the measured target positions. This was accomplished via a simple low-pass filtered difference calculation. The velocity estimates were noisy, and we did not try to reduce this noise during our experiments. Instead, we relied on the strength of the proposed control framework to guarantee safe and smooth motions.

### A. Motion Tracking Results

Figure 4 shows the visual servo control behavior of the proposed robot motion control scheme of Fig. 2 over a period of 25 s. The top diagram shows the position signals ${}_xP_i^{trgt}$, ${}_yP_i^{trgt}$, and ${}_zP_i^{trgt}$ (dashed lines), which are generated by the previously described image processing algorithm, as well as the position signals ${}_xP_i$, ${}_yP_i$, and ${}_zP_i$ (solid lines) that are computed by the online trajectory generator. Corresponding to this diagram, the three bottom diagrams show the velocity signals ${}_xV_i^{trgt}$, ${}_yV_i^{trgt}$, and ${}_yV_i^{trgt}$ of the image processing algorithm (dashed lines) and ${}_xV_i$, ${}_yV_i$, and ${}_zV_i$ of the trajectory generator (solid lines). To present a more detailed view of the achieved trajectories, Fig. 5 shows the position, velocity, and acceleration progressions of the time interval from 4.5 to 5 seconds. In addition to the twelve position and velocity signals, ${}_xA_i$, ${}_yA_i$, and ${}_zA_i$ are shown in this figure. One can clearly recognize, that the motion is always jerk-limited, and that $T^{vision}$ is much greater than $T^{cycle}$, as the values of $\mathbf{M}_i^{trgt}$ are only updated at discrete time instants.

### B. Reactions to Sensor Failures

Finally, we show the robust behavior against obstacles covering the camera or camera failures. Figure 6 shows the switching behavior from $\mathbf{M}_i^{trgt\diamond}$ to $\mathbf{M}_i^{trgt*}$ (cf. eqns. (6) and (7)). At $t = 6.515\,s$ an obstacle is detected in front of the stereo vision system, such that $\vartheta_i$ and $\sigma_i$ switch from 0 to 1, and the predefined safe state of motion $\mathbf{M}_i^{trgt*}$ is used as the desired target state of motion for the on-line trajectory generation algorithm. One can clearly recognize that the

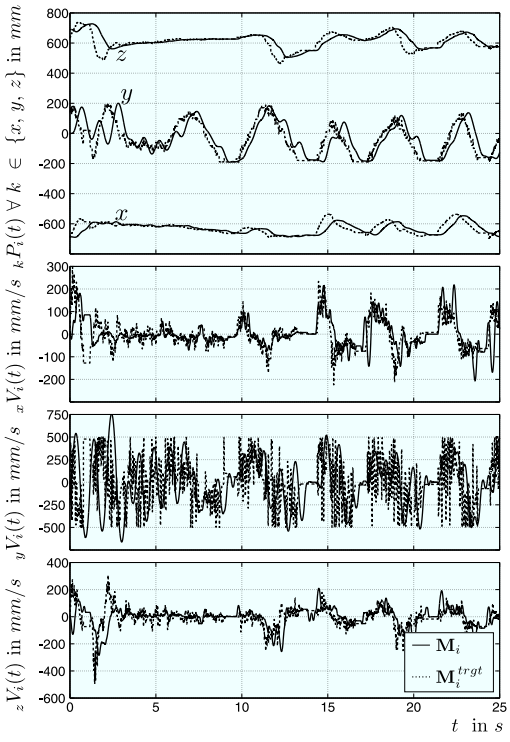


Fig. 4. Position and velocity progression over a period of 25 s. The progressions within the time interval from 4.5 s to 5.0 s is shown in detail in Fig. 5.

motion remains jerk-limited despite this abrupt switching procedure, and that the safe position

$$\vec{P}_{6515}^{\,trgt*} \;=\; (-449.718,\, 0.000,\, 531.362)\ \text{mm}$$

is reached at $t = 7.309\,s$.

It should be noted that we chose a strategy of returning to a safe position upon sensor failure. It is easy to envision this strategy as a valid one (out of many) for applications where humans are in or near the manipulator workspace. Upon sensor dropout, we would want the robot to return to a safe position or state free of collision. Conversely, other strategies can be developed (e.g., switching to other closed-loop controllers or a simple stop of motion).

## VI. Future Work

As discussed in Sec. IV-D, physical and/or artificial workspace limitations can only be robustly maintained by the control scheme of Fig. 2 if $\vec{V}_i^{\,trgt} = \vec{0}\ \forall\ i\ \in\ \mathbb{Z}$. Otherwise, the workspace limits $\vec{P}_i^{\,min}$ and $\vec{P}_i^{\,max}$ can be overshot. To bypass this problem, $\vec{V}_i^{\,trgt}$ has to be changed to an adapted value $\vec{V}_i^{\,\underline{trgt}}$. An extension has to be derived, such that the adaptation of $\vec{V}_i^{\,\underline{trgt}}$ does not only prevent from overshootings, but also from leaving the workspace. To achieve this, a function $f$

$$\vec{V}_i^{\,\underline{trgt}} \;=\; f\left(\vec{P}_i^{\,min},\, \vec{P}_i^{\,max},\, \mathbf{M}_i,\, \mathbf{B}_i\right) \qquad (9)$$

has to be developed. In this equation, the matrix $\mathbf{M}_i$ represents the state of motion at the instant $T_i$, and the matrix $\mathbf{B}_i$ contains the kinematic motion constraints at $T_i$. To achieve this, the desired function $f$ of eqn. (9) requires the property that $\vec{V}_i^{\,\underline{trgt}}$ is lower the smaller the distance between $\vec{P}_i^{\,trgt}$ and $\vec{P}_i^{\,min}$ or $\vec{P}_i^{\,trgt}$ and $\vec{P}_i^{\,max}$ is. If $\vec{P}_i^{\,trgt}$ equals either $\vec{P}_i^{\,min}$ or $\vec{P}_i^{\,max}$, $\vec{V}_i^{\,\underline{trgt}}$ must be zero.

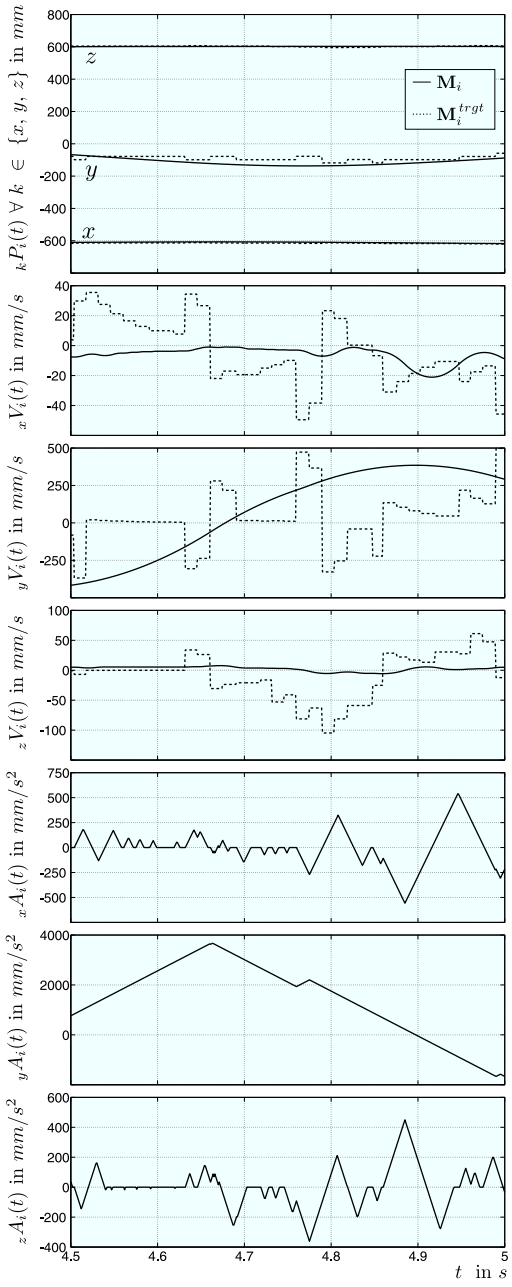


Fig. 5. Position, velocity, and acceleration progression of the on-line generated trajectory. This time interval corresponds to the visual servo control result shown in Fig. 4

## VII. Conclusion

A new visual servo control control scheme for robot arms based on an on-line trajectory generation algorithm has been proposed. It could be shown that the scheme behaves very robustly against strong noise, bad image quality, and misbehavior of image processing algorithms. Furthermore, the system can safely and instantaneously react to camera

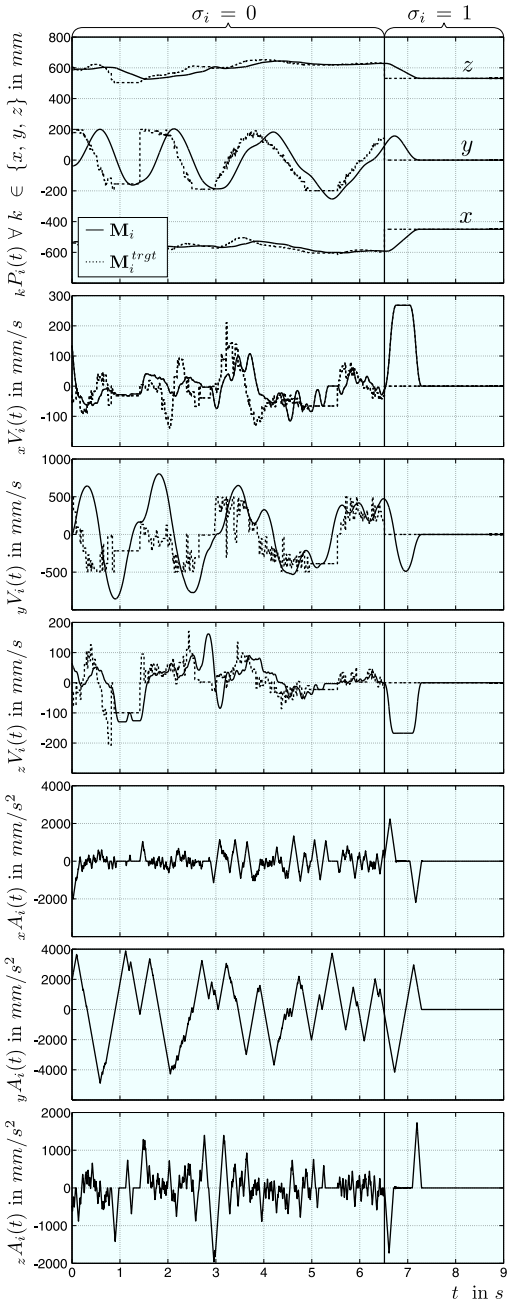


Fig. 6. Position, velocity, and acceleration progression during a switching procedure from $\mathbf{M}_i^{trgt\diamond}$ to $\mathbf{M}_i^{trgt*}$.

failures and obstacles covering the vision system. The on-line trajectory generator acts as a filter that can take the kinematic motion constraints of the robot explicitly into account, such that a jerk-limited trajectory can be guaranteed in any case. The new scheme is of a very simple nature, such that it can be embedded in many existing robot motion control schemes.

## APPENDIX

All experiments of this paper made use of the OTG Framework that has become part of the Reflexxes Motion Libraries [3], which can be downloaded from [38].

## ACKNOWLEDGEMENTS

Both authors would like to express their appreciation to *Professor Oussama Khatib*, who is currently hosting them at the *Stanford Artificial Intelligence Laboratory*. Moreover, the first author is deeply indebted to the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) that is currently funding him. Finally, we would like to thank *QNX Software Systems* and *Wind River Systems* for providing free software licenses.

## REFERENCES

[1] T. Kröger and F. M. Wahl. On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, 26(1):94–111, February 2010.

[2] T. Kröger. *On-Line Trajectory Generation in Robotic Systems*, volume 58 of *Springer Tracts in Advanced Robotics*. Springer, Berlin, Heidelberg, Germany, first edition, January 2010.

[3] T. Kröger. Opening the door to new sensor-based robot applications — The Reflexxes Motion Libraries. In *Proc. of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[4] D. A. Forsyth and J. Ponce. *Computer Vision A Modern Approach*. Pearson Education, 2003.

[5] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry. *An Invitation to 3-D Vision*. Springer, New York, NY, USA, 2003.

[6] F. M. Wahl. *Digital Image Signal Processing*. Artech House, Boston, MA, USA, first edition, 1987.

[7] F. Chaumette and S. A. Hutchinson. Visual servo control. Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 4(13):82–90, December 2006.

[8] F. Chaumette and S. A. Hutchinson. Visual servo control. Part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, 1(14):109–118, March 2007.

[9] F. Chaumette and S. A. Hutchinson. Visual servoing and visual tracking. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 24, pages 563–583. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[10] L. E. Weiss, A. C. Sanderson, and C.P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5):404–417, 1987.

[11] J. T. Feddema and O. R. Mitchell. Vision-guided servoing with feature-based trajectory generation. *IEEE Trans. on Robotics and Automation*, 5(5):691–700, 1989.

[12] N. R. Gans. *Hybrid Switched System Visual Servo Control*. PhD thesis, Department of General Engineering, University of Illinois at Urbana-Champaign, 2005.

[13] N. R. Gans and S. A. Hutchinson. Stable visual servoing through hybrid switched-system control. *IEEE Trans. on Robotics*, 23(3):530–540, June 2007.

[14] L. Deng, F. Janabi-Sharifi, and W. J. Wilson. Hybrid motion control and planning strategies for visual servoing. *IEEE Trans. on Industrial Electronics*, 52(4):1024–1040, August 2005.

[15] O. Tahri, Y. Mezouar, F. Chaumette, and P. Corke. Decoupled image-based visual servoing for cameras obeying the unified projection model. *IEEE Trans. on Robotics*, 26(4):684–697, August 2010.

[16] Y. Wang, H. Lang, and C. W. de Silva. A hybrid visual servo controller for robust grasping by wheeled mobile robots. *IEEE/ASME Trans. on Mechatronics*, 15(5):757–769, October 2010.

[17] J. Jeong and S. Lee. Outlier elimination method for robust visual servo control in complex environment. In *Proc. of the IEEE International Conference on Robotics and Biomimetics*, pages 938–943, Tianjin, China, December 2010.

[18] G. Chesi. Visual servoing path planning via homogeneous forms and LMI optimizations. *IEEE Trans. on Robotics*, 25(2):281–291, April 2009.

[19] X. Broquère, D. Sidobre, and I. Herrera-Aguilar. Soft motion trajectory planner for service manipulator robot. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2808–2813, Nice, France, September 2008.

[20] X. Broquère, D. Sidobre, and K. Nguyen. From motion planning to trajectory control with bounded jerk for service manipulator robots. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4505–4510, Anchorage, AK, USA, May 2010.

[21] S. Liu. An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators. In *Proc. of the seventh International Workshop on Advanced Motion Control*, pages 365–370, Maribor, Slovenia, July 2002.

[22] R. Haschke, E. Weitnauer, and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3248–3253, Nice, France, September 2008.

[23] K. Ahn, W. K. Chung, and Y. Yourn. Arbitrary states polynomial-like trajectory (ASPOT) generation. In *Proc. of the 30th Annual Conference of IEEE Industrial Electronics Society*, volume 1, pages 123–128, Busan, South Korea, November 2004.

[24] R. H. Castain and R. P. Paul. An on-line dynamic trajectory generator. *The International Journal of Robotics Research*, 3(1):68–72, March 1984.

[25] X. Broquère. Homepage — Movies, `http://homepages.laas.fr/xbroquer/media2.php` (accessed: Mar. 6, 2011). Internet, 2011.

[26] K. Waldron and J. Schmiedeler. Kinematics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 1, pages 9–33. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[27] W. Chung, L.-C. Fu, and S.-H. Hsu. Motion control. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 6, pages 133–159. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[28] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer, New York, NY, USA, first edition, 2007.

[29] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, RA-3(1):43–53, February 1987.

[30] L. Sentis, J. Park, and O. Khatib. Compliant control of multi-contact and center of mass behaviors in humanoid robots. *IEEE Trans. on Robotics*, 26(3):483–501, June 2010.

[31] R. Bischoff, J. Kurth, G. Schreiber, R. Köppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR lightweight robot arm — A new reference platform for robotics research and manufacturing. In *Proc. of the Joint Conference of ISR 2010 (41st Internationel Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, Munich, Germany, June 2010. VDE Verlag.

[32] KUKA Laboratories GmbH, Zugspitzstraße 140, D-86165 Augsburg, Germany. Homepage. `http://www.kuka-labs.com/en` (accessed: Aug. 22, 2011). Internet, 2011.

[33] G. Schreiber, A. Stemmer, and R. Bischoff. The fast research interface for the KUKA lightweight robot. In *Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications — How to Modify and Enhance Commercial Controllers at the IEEE International Conference on Robotics and Automation*, pages 15–21, Anchorage, AK, USA, May 2010.

[34] T. Kröger. Documentation of the fast research interface library, version 1.0, `http://cs.stanford.edu/people/tkr/fri/html` (accessed: Feb. 9, 2012). Internet, November 2011.

[35] IDS Imaging Development Systems GmbH, Dimbacher Straße 6–8, D-74182 Obersulm, Germany. Homepage. `http://www.ids-imaging.com` (accessed: Mar. 28, 2011). Internet, 2011.

[36] J.-Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc` (accessed: Mar. 28, 2011). Internet, 2011.

[37] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Sebastopol, CA, USA, 2008.

[38] Reflexxes GmbH, Sandknöll 7, D-24805 Hamdorf, Germany. Hompepage. `http://www.reflexxes.com` (accessed: Feb. 6, 2012). Internet, 2012.