Visual Servoing of Presenters in Augmented Virtual Reality TV Studios

Suraj Nair, Thorsten Röder, Giorgio Panin, Alois Knoll, *Member, IEEE*

***Abstract*— This paper presents recent developments in the area of visual tracking methodologies for an applied real-time person localization system, which primary aims to robust and failure-safe robotic camera control. We applied the described methods to virtual-reality TV broadcasting studio environments in Germany in order to close a gap in TV studio automation. The presented approach uses robot camera systems based on industrial robots in order to allow high-precision camera manipulation for virtual TV studios, without limiting the degrees of freedom that a robot manipulator can provide. To take robotic automation in TV studios to a completely new dimension, we have imparted intelligence to the system by tracking the TV presenter in real-time, allowing him or her to move naturally and freely within the TV studio, while maintaining the required scene parameters, such as position in the scene, zoom, focus, etc. according to prior defined scene behaviors. The tracking system itself is distributed and has proven to be scalable to multiple robotic camera systems operating synchronously in real-world studios.**

I. Introduction

Virtual TV studios have gained immense importance in the broadcasting area over the past years, and are becoming the mainstream way of broadcasting in the future. This evolutionary step is based on developments in computer graphics and rendering hardware, that allow to achieve high quality images with reasonable effort. For these reasons, the complexity and quality of HDTV contents in fully virtual sets have seen a new high, providing an impressive experience for educational and documentary movies, as well as for e.g. weather or financial forecast transmissions. The present day technology also allows broadcasters to have virtual objects inside the virtual scene. Although with these advances the acceptance for this technology has increased, the system complexity has also inherently increased. Therefore it is a crucial goal to keep systems maintainable for human operators, and thus in general to hide their complexity.

Besides the nowadays powerful rendering engines and astonishing 3D graphics available, the fundamental quality of a virtual scene depends on the real-time robustness and accuracy of three major system components, namely: 1. The camera tracker, that recovers the absolute 3D position and orientation of the camera (e.g. by using external infrared sensors or odometry), 2. the rendering software itself, and 3. the precision with which the camera is moved w.r.t. translation, rotation, zoom and focus.

Virtual TV studios around the world use a typical camera configuration for motion control, consisting of a pedestal housing a pan tilt unit. These systems have limitations in terms of degrees of freedom, motion smoothness and high costs of the external sensor-based tracker, used to recover the 3D pose of the camera.

Authors Affiliation: Technische Universität München, Fakultät für Informatik.

Address: Boltzmannstrasse 3, 87452 Garching bei München (Germany).
Email: { nair, roeder, panin, knoll }@in.tum.de

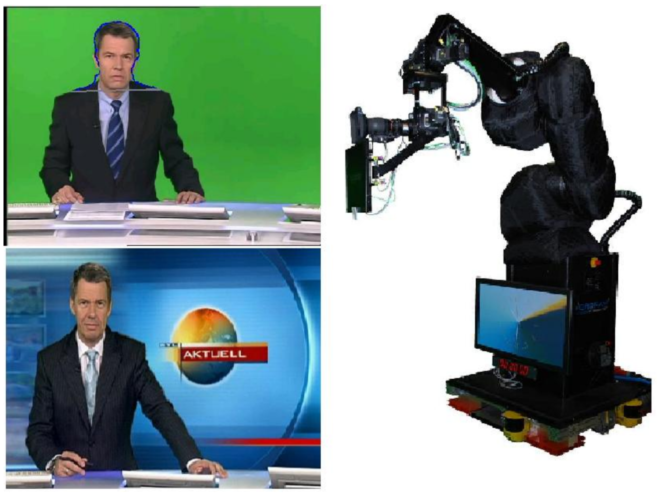
Fig. 1. A virtual set for daily news broadcasting events. The upper and lower left pictures show the virtual studio. The right shows our robot camera system for broadcast automation. (*image courtesy: RTL Television Studio Köln, Germany, and Robotics Technology Leaders GmbH*).

Industrial robot arms can perform precise manipulation of TV cameras with high repeatability in large workspaces, using many degrees of freedom. Moreover, the main advantage of a robotic system is that the 3D camera pose is obtained free of cost through the robot kinematics, thereby eliminating the need for external trackers.

Robotic automation in TV studios can be pushed to a new high by imparting intelligence to the robot system. To this aim, in this paper we propose a vision-based person tracker for visual servoing, integrating multiple visual modalities. The system is able to localize the moderator and keep her/him within the screen while sitting or freely walking inside the studio.

Another important feature is the automatic positioning of the moderator during different run-down scenes according to a pre-determined region of interest. For example, when switching from a scene with the moderator in the center to one where visual graphics need to be rendered, it is necessary to hold the moderator on the left (or right) part of the scene. This can be achieved using the tracking results with almost no need for human intervention. In comparison to our previous published work [1] and [2], the system has improved in scalability, distribution and contains new modules for three-dimensional tracking. It has been completely integrated into a commercially distributed system called RoboKam®.

The present paper is organized as follows: Sec. II briefly reviews the related state-of-the-art; Sec. III describes the visual tracker, providing a system overview, the user interface and the tracking methodology. Sec. III-F explains the robot controller. Experimental results are given in Sec. IV, and conclusions including future system developments are finally given in Sec. V.

II. Prior art

To our knowledge, no fully integrated and self-contained vision-driven robot cameraman has been developed for virtual reality (VR) TV studio applications. However, the literature concerning single person or multiple people tracking in video surveillance, mobile robotics and related fields, already counts several well-known examples, that we briefly review here. Although implementations of similar vision systems do exist in the research and scientific domain, successful application in real world scenarios remains very limited.

Multiple people trackers [3], [4], [5], have the common requirement of using a very little and generic offline information concerning the person shape and appearance, while building and refining more precise models (color, edges, background) during the on-line tracking task; this unavoidable limitation is due to the more general context with respect to single-target tracking, for which instead specific models can be built off-line.

Many popular systems for single-target tracking are based on color histogram statistics [6], [7], [8], [9] and employ a pre-defined shape and appearance model throughout the whole task.

In particular, [8] uses a standard particle filter with color histogram likelihood with respect to a reference image of the target, while [7] improves this method by adapting the model on-line to light variations, which however may introduce drift problems in presence of partial occlusions; the same color likelihood is used by the well-known *mean-shift* kernel tracker [9].

The person tracking system [6] employs a complex model of shape and appearance, where color and shape blobs are modeled by multiple Gaussian distributions, with articulated degrees of freedom, thus requiring a complex modeling phase, as well as several parameters specification.

By comparison, in our system the off-line model is kept to a minimum complexity, while at the same time retaining the relevant information concerning the spatial layout of colour statistics.

The main advantage of our tracker is therefore its usability, flexibility and successful integration within the RoboKam® system for broadcast automation. In its complete form, it bridges the gap between scientific research and industrial applications with a minimum required throughput.

III. The vision-based person tracker

In this Section we describe our vision-based system for moderator localization and continuous tracking, providing details regarding its design and implementation.

A. System overview

The system consists of two parts: 1. a single person tracker, operating on each robot, localizes the moderator in 2D position (x,y), scale h and rotation θ within the field of view of the TV camera, allowing the robot to hold the moderator in a desired region of the scene, with the desired zoom and focus.

The target is modeled by representing the head and shoulder silhouette, in order to provide a stable scale output, which is not possible by using only color statistics lacking direct spatial information, 2. a supplementary overhead stereo tracker localizes the target over the entire studio floor w.r.t. 3D translation (x,y,z). Although the 3D tracker makes the overall architecture more complex, it serves very important features such as:

- Initialize each robot camera, so that they can bring the target in the respective fields of view independently of their initial positions
- Re-initialize the local trackers in case of a target loss, to recover their 2D locations
- Initialize zoom and focus control of each robot camera
- Handle interactions of the person with virtually rendered objects, such as occlusion of the object when the moderator moves around it

The target is modeled as a fixed omega-shape for the person tracker, while for the overhead tracker it is represented by a 3D box, along with a frontal picture of the person appearance.

Figure 2 gives an overview of the complete system. Each tracker is integrated into the robotic camera system through a modular middleware called COSROBE, developed for communication and configuration of studio devices.

It is possible to have more than one robot camera in the same studio, although there exists only a single overhead tracker. This tracker uses ceiling mounted firewire cameras in a stereo configuration, to compute the 3D pose of the moderator. These cameras are calibrated with respect to a common *world* frame, with individual intrinsic and extrinsic parameters.

For the 2D person tracker we choose a Kalman filter [10], running on the output of a contour tracker known as contracting curve density *CCD* algorithm [11], [12], based on separation of local color statistics. In an object tracking context, separation takes place between the object and the background regions, across the screen contour projected from the shape model onto a given camera view, under a predicted pose hypothesis.

The overhead tracker uses a sampling-importance-resampling particle filter [13] working on color histograms, providing a joint likelihood of the 3D target pose. We choose a particle filter for the overhead tracker, over a more conventional Kalman filter, techniques because the overhead tracker has to be highly robust when dealing with multi-modal likelihoods, due to a high probability of having a cluttered background. This way it can support the 2D person tracker system in cases of loss detection and re-initialization.

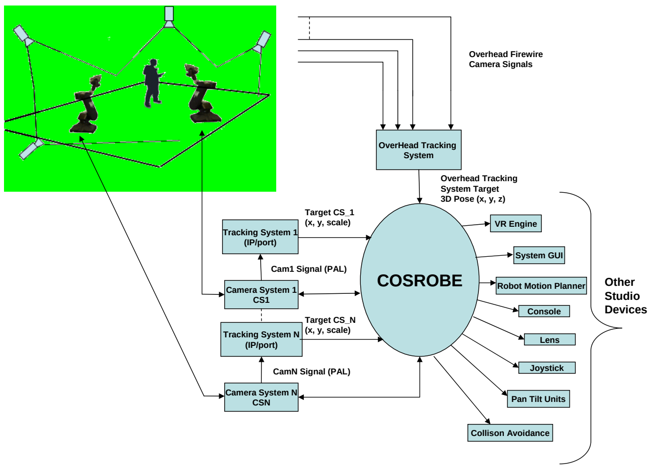

Fig. 2. Block diagram of the system architecture. The middleware COSROBE integrates multiple robotic camera devices and tracking systems. An overhead stereo camera is used to initialize the system at startup and in cases of target loss. Since each camera device is calibrated to a given world point, tracking information can be fused. The middleware also connects to the virtual reality engine and system configuration modules, like. e.g. the operator GUI or the motion planning module.

Concerning computational resources, the software for each camera system runs on a separate PC and obtains the TV camera picture through a frame grabber. The overhead tracker uses stereo FireWire cameras with wide angle lenses for covering the complete studio floor. Currently the overhead tracker runs on a single PC, to which two FireWire cameras are connected.

B. Graphical user interface

The tracking system can be easily controlled by the user using an intuitive graphical user interface (Fig. 3). The GUI provides functions such as start/stop tracking, and automatic target re-detection. The latter is done by frontal face detection [14].

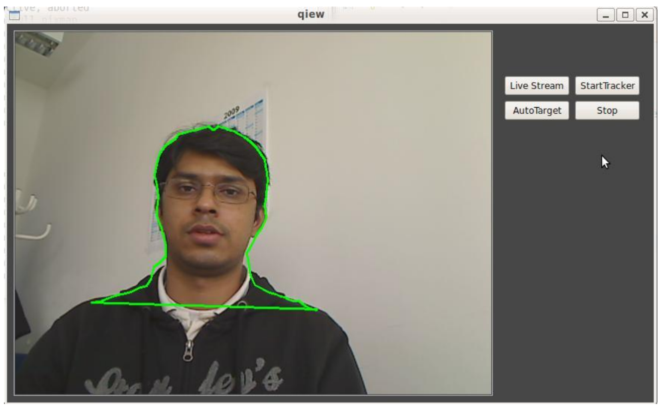

Fig. 3. Graphical user interface

C. The 2D person tracker

Our trackers are designed and implemented following a recently developed general-purpose framework [15], following a *tracking pipeline* concept. Fig. 4 describes the pipelines for the two trackers of the previous Section.

Each 2D tracker holds a state-space representation of the model pose, given by a planar roto-translation and scale in the image plane. The Kalman filter provides the sequential prediction and update of the respective 2D state $s = (x, y, h, \theta)$.

*1) **Pre-processing**:* Sensor data are obtained from the TV camera in a raw PAL format, through a frame grabber. In CCD, no pre-processing is done, and color pixels are directly used by the feature matching module, in order to collect local statistics and optimize the pose. Therefore, the pre-processing function merely copies the input image to a local storage for the other modules.

*2) **Tracker prediction**:* The Kalman filter generates a prior state hypothesis s_t^- from the previous state (s_{t-1}) by applying a Brownian motion model

$$s_t^- = s_{t-1} + w_t; \tag{1}$$

with w a white Gaussian noise sequence. Although very simple, this model suits our needs very well, providing a Gaussian distribution around the current state which helps keeping track of the target when it is not moving, as well as when it exhibits motion at a reasonable speed. If the target moves very fast, then we would need more adequate models such as constant velocity, but this situation normally does not arise in TV studio environments.

*3) **CCD Likelihood for Feature Level Matching**:* In the original CCD algorithm [11], local areas for collecting color statistics are given by regions around each contour sample position, on each side of the contour. In order to simplify the computation, as also suggested in [12], we first sample points along the respective normals, separately collect the statistics, and afterwards *blur* each statistics with the neighboring ones (Fig. 5). This is fully equivalent to the initial process, but computationally more convenient.

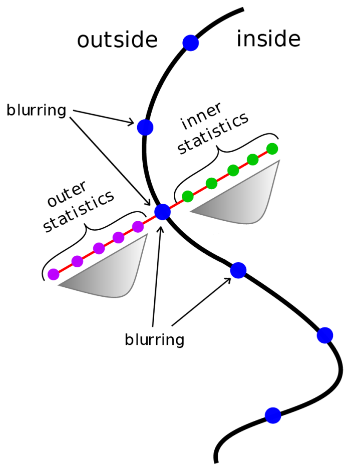

Fig. 5. The CCD algorithm tries to maximize the separation of color statistics between two image regions. The algorithm first samples pixels along the normals for collecting local color statistics.

From each contour position h_i, foreground and background

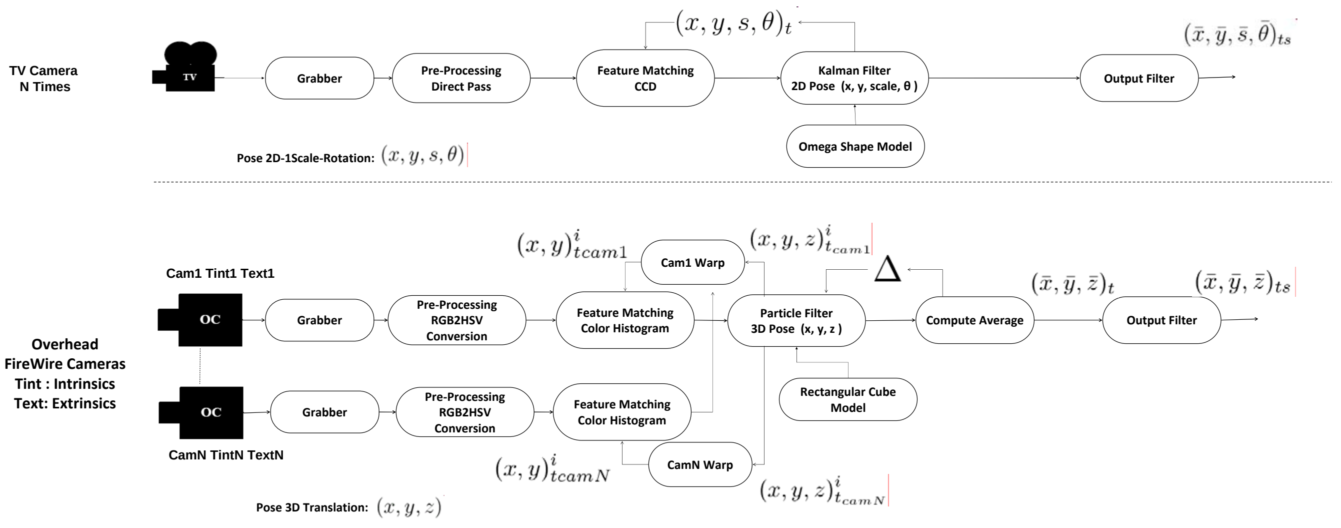

Fig. 4. Person tracking pipelines. The top pipeline estimates a planar pose with 4 degrees of freedom (roto-translation and scale). A Kalman filter incorporates a feature-based contour matching modality, using an omega shape model. The pipeline on the bottom describes the overhead tracker: here, a stereo setup is used in order to estimate the 3D position of the target. After capturing the image frames, they are converted into an HSV color space. A SIR particle filter is used for Bayesian prediction and correction. As an objective function we use 2D color statistics within a 3D box model, that gets simultaneously projected and evaluated on each camera.

color pixels are collected along the normals n_i up to a distance L, and local statistics up to the 2nd order are estimated

$$\begin{aligned} v_i^{0,B/F} &= \sum_{d=1}^{D} w_{id} \\ v_i^{1,B/F} &= \sum_{d=1}^{D} w_{id} I(h_i \pm L\bar{d}n_i) \\ v_i^{2,B/F} &= \sum_{d=1}^{D} w_{id} I(h_i \pm L\bar{d}n_i) I(h_i \pm L\bar{d}n_i)^T \end{aligned} \quad (2)$$

with $\bar{d} \equiv d/D$ the normalized contour distance, where the $\pm$ sign is referred to the respective side, and image values I are 3-channel RGB. The local weights w_{id} decay exponentially with the normalized distance, thus giving a higher confidence to observed colors near the contour. For a more detailed explanation please refer to [12].

Single-line statistics are afterwards *blurred* along the contour, providing statistics distributed on local areas

$$\tilde{v}_i^{o,B/F} = \sum_j \exp(-\lambda\,|i-j|) v_j^{o,B/F};\ o = 0,1,2 \quad (3)$$

and finally normalized

$$\begin{aligned} \bar{I}_i^{B/F} &= \frac{\tilde{v}_i^{1,B/F}}{\tilde{v}_i^{0,B/F}} \\ \bar{R}_i^{B/F} &= \frac{\tilde{v}_i^{2,B/F}}{\tilde{v}_i^{0,B/F}} \end{aligned}$$

in order to provide the two-sided, local RGB means $\bar{I}$ and (3×3) covariance matrices $\bar{R}$.

The second step involves computing the residuals and Jacobian matrices for the Gauss-Newton pose update. For this purpose, observed pixel colors $I(h_i + L\bar{d}n_i)$ with $\bar{d} = -1, ..., 1$, are classified according to the collected statistics (4), under a fuzzy membership rule $a(x)$ to the foreground region

$$a(\bar{d}) = \frac{1}{2}\left[erf\left(\frac{\bar{d}}{\sqrt{2}\sigma}\right) + 1\right] \quad (4)$$

which becomes a sharp $\{0,1\}$ assignment for $\sigma \to 0$; pixel classification is then accomplished by mixing the two statistics accordingly

$$\begin{aligned} \hat{I}_{id} &= a(\bar{d})\bar{I}_i^F + (1 - a(\bar{d}))\bar{I}_i^B \\ \hat{R}_{id} &= a(\bar{d})\bar{R}_i^F + (1 - a(\bar{d}))\bar{R}_i^B \end{aligned} \quad (5)$$

and color residuals are given by

$$E_{id} = I(h_i + L\bar{d}n_i) - \hat{I}_{id} \quad (6)$$

with covariances $\hat{R}_{id}$.

Finally, the $(3 \times n)$ derivatives of E_{id} can be computed by differentiating (5) and (4) with respect to the pose parameters[1]

$$J_{id} = \frac{\partial \hat{I}_{id}}{\partial s} = \frac{1}{L}(\bar{I}_i^F - \bar{I}_i^B)\frac{\partial a}{\partial \bar{d}}\left(n_i^T \frac{\partial h_i}{\partial s}\right) \quad (7)$$

which are stacked together in a global Jacobian matrix $\mathbf{J}_{ccd}$.

The state is then updated using a Gauss-Newton step

$$\begin{aligned} s &= s + \Delta s \\ \Delta s &= J_{ccd}^{+} E_{ccd} \end{aligned} \quad (8)$$

[1]As in [12], we neglect the dependence of R_{id} on s while computing the Jacobian matrices.

using the stacked Jacobian and residual vector. After each iteration, the measurement covariance is reduced with exponential decay, providing a robust multi-resolution convergence to the locally optimal pose

$$Z_{ccd} = s^* \tag{9}$$

which is used as a measurement Z for the respective Kalman filter.

D. 3D overhead tracker pipeline

The overhead tracker holds a state-space representation of the 3D model pose, given by a translation (x,y,z) of the box model with respect to the reference system of the stereo setup. The particle filter provides the sequential prediction and update of the respective state $s = (x,y,z)$.

*1) **Pre-processing**:* The sensor data for the person tracker is obtained from each FireWire camera in the raw RGB-444 format. The image from each camera is pre-processed by performing RGB to HSV color conversion $z_c^{col}; c = 1,\dots,C$ for the color-based likelihood, where the index c corresponds to the FireWire camera, and C is the total number of cameras. An optional background subtraction step is possible before color conversion, to further increase robustness in suitable studio setups.

*2) **Tracker prediction**:* The particle filter generates several prior state hypotheses s_t^i from the previous distribution $(s^i, w^i)_{t-1}$ through a Brownian motion model

$$s_t^i = s_{t-1}^i + w_t^i \tag{10}$$

with w a zero-mean Gaussian noise of pre-defined covariance in the (x,y,z) state variables. Deterministic resampling over the previous weights w_{t-1}^i is employed at each frame.

*3) **Color likelihood**:* For each generated hypothesis, the tracker asks for a computation of the likelihood values $P(z_c^{col}|s^i)$ after projecting the hypothesis onto each camera view.

The object model defining the person shape is projected onto the HSV image of each camera at the predicted hypothesis s_t^i, using the respective intrinsic and extrinsic parameters. The underlying H and S color values are collected in the respective 2D histogram $q_c\left(s_t^i\right)$, that is compared with the reference one q_c^* through the Bhattacharyya coefficient [8]

$$B\left(q_c\left(s\right), q_c^*\right) = \left[1 - \sum_N \sqrt{q_c^*\left(n\right) q_c\left(s,n\right)}\right]^{\frac{1}{2}} \tag{11}$$

where the sum is performed over the $(bin \times bin)$ histogram bins (in the current implementation, $bin = 10$). The computation is done separately for each camera c.

The overall likelihood is then evaluated under independent Gaussian models in the overall residual

$$P(z^{col}|s_t^i) \propto exp(-\prod_c (B_c^2/\lambda)) \tag{12}$$

with a given covariance λ.

*4) **Computing the estimated state**:* The average state $\bar{s}_t$

$$\bar{s}_t = \frac{1}{N}\sum_i w_t^i s_t^i \tag{13}$$

is computed and the three components $(\bar{x},\bar{y},\bar{z})$ are returned to the robot controller.

E. Loss detection and handover between trackers

One of the most important features of our system is the possibility to automatically detect a track loss when the person leaves the scene or gets occluded, and to re-initialize the system in such situations, using the overhead tracker.

In principle there are two main techniques to determine a possible target loss, 1) based on a covariance test, 2) based on a likelihood test. A covariance test would be independent from the actual likelihood values, but it may fail to detect loss when the hypotheses concentrate on a small peak (false positive), which has a low covariance as well. This is very undesirable in a TV studio application, where the only target that should be detected is the moderator, and never other people or objects which the robot could drift to.

On the other hand, the likelihood test is dependent on the likelihood values, and may detect too often a loss (false negatives), for example in presence of light variations. However, in a TV studio light conditions are strongly controlled, and an occasional false negative is acceptable, as long as the re-initialization is successful.

Therefore, we employ the likelihood test on the estimated state $\bar{s}_t$ from both trackers, and declare a loss whenever $P(z|\bar{s}_t)$ decreases below a minimum value P_{min}. This threshold is set as a percentage (e.g. $\leq 10\%$) of a reference value P_{ref}, initially given by the observed maximum likelihood.

In order to provide adaptivity to variable postures, e.g. when the moderator turns on a side, as well as light or shading variations, P_{ref} itself is slowly adapted if the last average likelihood $P(z|\bar{s}_{t-1})$ is comparable to P_{ref} (e.g. $\geq 60\%$). When a target loss occurs, the tracker is automatically re-initialized.

F. Robot controller

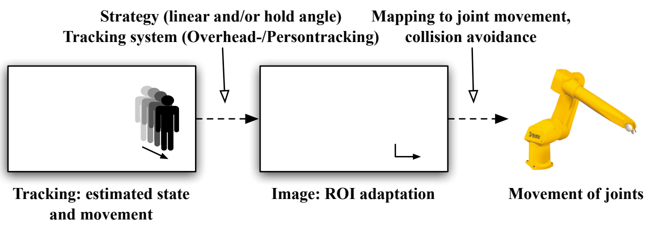
Strategy (linear and/or hold angle)
Tracking system (Overhead-/Persontracking)
Mapping to joint movement, collision avoidance
Tracking: estimated state and movement
Image: ROI adaptation
Movement of joints

Fig. 7. Robot control methodology

In order to keep the target in a predefined ROI it is necessary to generate the relative motion parameters for the corresponding robot system. This is achieved by using 2D pose information from the local tracker and convert it to 3D motion commands in the manipulator space. For this conversion, additional parameters have to be considered, namely:

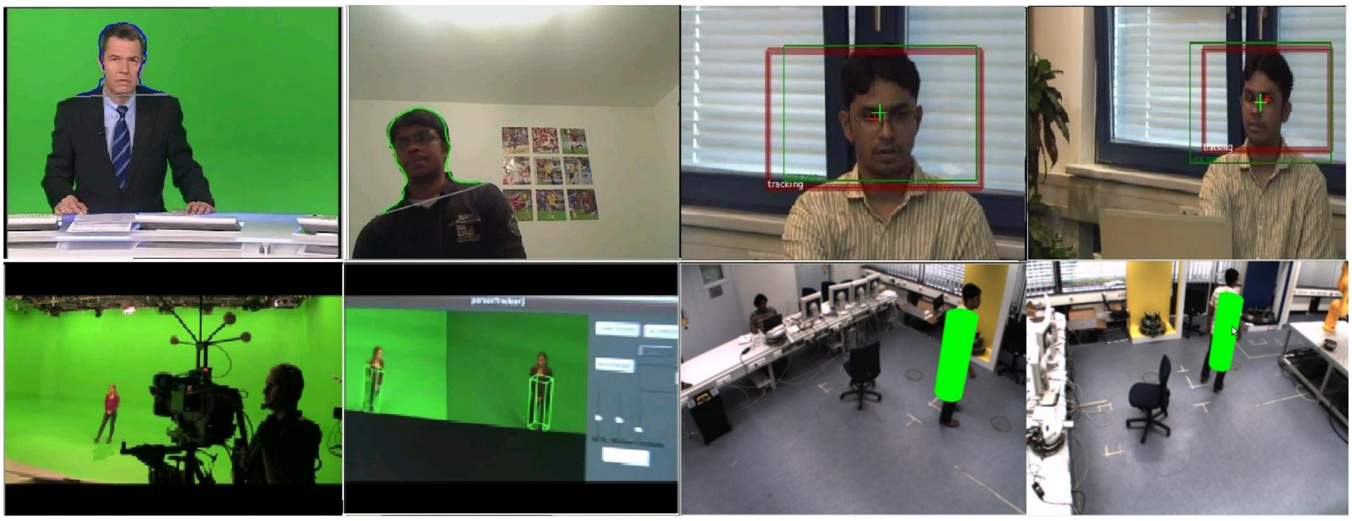

Fig. 6. Experimental results Upper row: 2D contour tracking, both in a TV news studio with a green wall background and constant lighting, as well as on a cluttered background. Bottom row: overhead stereo tracker, localizing the moderator in 3D translation.

- Region of interest (ROI): it is the desired area in the image space of each camera system where the target should be held, e.g. in weather broadcasts the target appears usually on the right side of the scene.
- Balance speed: the speed of the robot is specified for X, Y, pan and tilt, as an absolute percentage ranging from $[-100\%; 0\%; 100\%]$ of the maximum values. This speed is used by the camera system in order to get the target back into the ROI. The actual speed also depends on the distance between the target and the ROI. The effective speed is then proportional to the calculated distance, providing smooth motion properties.

As shown Fig. 7, 3D Cartesian control is computed out of the X, Y, pan, tilt speeds. We propose two different operation modes for the joint control:

- Normal mode: The movement of the robot is limited to a linear motion in the X and Y direction, and angular motion for Pan and Tilt in order to get the target back into the desired ROI.
- Hold-angle mode: The movement of the robot is done in 2 phases: In the 1st phase, the robot uses only Pan and Tilt to bring the target back into the ROI, and in the 2nd phase it uses linear X and Y motion to compensate and hold a predefined camera viewing angle.

The robot in used in the studio is a Stäubli RX-160L with modified kinematics. In laboratory experimental setups we use the Stäubli RX-90. The RX-160L has 6 joints, but we replaced joints 4, 5 and 6 with a specialized tilt-pan-tilt configuration, to improve capabilities of camera motion control, as illustrated in Fig. 1. This robot comes with the CS-8 Stäubli controller, so we bypass the kinematics computation provided by the controller and instead compute the kinematics and motion trajectories externally, and send this information directly to the low level controller through the respective interface. The controller runs a real-time extension of LINUX. For the PID control, we instead rely on the CS-8 controller shipped by the manufacturer.

In a TV Studio setup, production is done on a scene-by-scene basis, with respect to the *run-down*. The robot system should react intuitively to a switch, from one scene to the other. This is achieved automatically, with almost no need for human intervention. Different scenes require the moderator to appear in different regions, and with different zoom and focus settings. This information can be combined with the run-down information, in order to enable a completely automatic switch of the camera position and zoom/focus, holding the moderator within the current region of interest.

IV. Experiments and Results

The system has been evaluated in real TV studios, for virtual reality productions. For this purpose, standard Desktop PCs with $2.4GHz$ Intel Pentium IV and standard graphics hardware, have been used to realize each camera tracker and the overhead stereo tracker, all running on Linux operating system.

Both trackers run in real-time, with approximately $15-20fps$, which we found more than sufficient for the robot controller, that requests visual feedback every $200ms$. Image resolution is 640×480 pixels. Fig. 8 illustrates a lab robot with a pan-tilt unit and a TV camera: the robot camera follows the person while moving and interacting with other people in the scene.

Fig. 6 show some experimental results of the 2D person tracker, and also illustrates the 3D overhead tracker. Again, the system keeps good track of the person during the whole sequence. The accompanying video file demonstrates performances in different scenarios, including automatic scene switching during run-down.

Although the tracker perform in real-time, it communicates to the robot controller through a common middleware, with TCP-IP sockets, that introduce some delay in the motion control. The robot controller send commands to the robot with a cycle time of 4msec, at the same time requesting

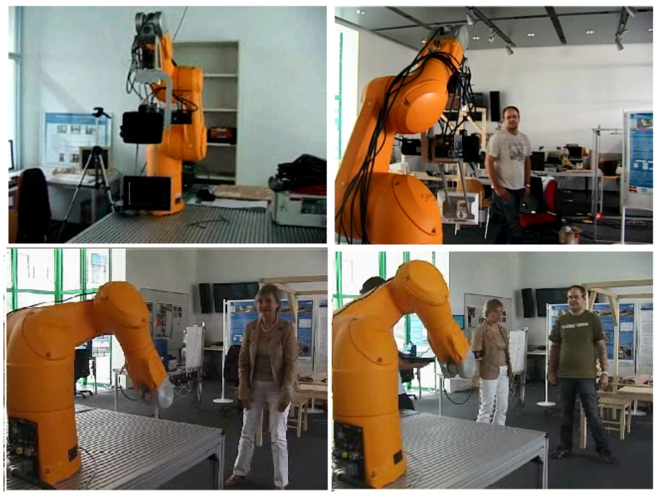

Fig. 8. In-house testbed: example sequence with the robot controller in action. Here the control of the robot arm as well as the pan-tilt unit, have to be jointly performed, to provide smooth jitter-free trajectories.

data from the tracker every 100msec; the tracker generates data at 15-25 fps, thereby fulfilling the requests from the robot controller. There is indeed some delay (approximately 200msec) when the robot reacts to fast movements of the moderator, but in TV studio environments, and especially in virtual sets for new production, this situation is very rare. Most of the motion control takes place during scene switching, where a small delay can be tolerated during production.

V. CONCLUSIONS AND FUTURE WORKS

A. *Conclusions and future work*

We presented a distributed and scalable person tracking system for visual servoing in Virtual-Reality TV applications. In particular, we improved robustness and usability of the current system with respect to our previous work [1], [2] in many respects.

The use of CCD algorithm for 2D tracking made possible a very robust localization of the person with a stable scale estimate, needed for zoom control as illustrated in the top-right of Fig. 6. The new stereo overhead tracker allows 3D pose estimation, thus considerably improving performances of loss detection. Moreover, the system has been successfully integrated into a real-world robot controller, being used live in TV studios.

Our future developments will focus on scaling the overhead system to cover lager floor areas using camera grids. New visual modalities are also planned to be integrated, along with a fuzzy fusion module, to select the more reliable modality according to the scene conditions.

VI. ACKNOWLEDGMENTS

The authors wish to express their acknowledgments to the RTL Television Studio Köln, Germany, for providing the environment, the pictures and live video sequences for our experiments.

REFERENCES

[1] S. Nair, G. Panin, M. Wojtczyk, C. Lenz, T. Friedelhuber, and A. Knoll, “A multi-camera person tracking system for robotic applications in virtual reality tv studio,” in *Proceedings of the 17th IEEE/RSJ International Conference on Intelligent Robots and Systems 2008*. IEEE, Sep. 2008.

[2] S. Nair, G. Panin, T. Röder, T. Friedelhuber, and A. Knoll, “A distributed and scalable person tracking system for robotic visual servoing with 8 dof in virtual reality tv studio automation,” in *Proceedings of the 6th International Symposium on Mechatronics and its Applications (ISMA09).* IEEE, Mar. 2009.

[3] I. Haritaoglu, D. Harwood, and L. S. Davis, “W4: A real time system for detecting and tracking people,” in *CVPR ’98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 1998, p. 962.

[4] N. T. Siebel and S. J. Maybank, “Fusion of multiple tracking algorithms for robust people tracking,” in *ECCV ’02: Proceedings of the 7th European Conference on Computer Vision-Part IV*. London, UK: Springer-Verlag, 2002, pp. 373–387.

[5] M. Isard and J. MacCormick, “Bramble: A bayesian multiple-blob tracker,” in *ICCV*, 2001, pp. 34–41.

[6] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Real-time tracking of the human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[7] K. Nummiaro, E. Koller-Meier, and L. J. V. Gool, “An adaptive color-based particle filter,” *Image Vision Comput.*, vol. 21, no. 1, pp. 99–110, 2003.

[8] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, “Color-based probabilistic tracking,” in *ECCV ’02: Proceedings of the 7th European Conference on Computer Vision-Part I*. London, UK: Springer-Verlag, 2002, pp. 661–675.

[9] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.

[10] G. Welch and G. Bishop, “An introduction to the kalman filter,” Tech. Rep., 2004.

[11] R. Hanek and M. Beetz, “The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria,” *Int. J. Comput. Vision*, vol. 59, no. 3, pp. 233–258, 2004.

[12] G. Panin, A. Ladikos, and A. Knoll, “An efficient and robust real-time contour tracking system,” in *ICVS*, 2006, p. 44.

[13] M. Isard and A. Blake, “Condensation – conditional density propagation for visual tracking,” *International Journal of Computer Vision (IJCV)*, vol. 29, no. 1, pp. 5–28, 1998.

[14] P. A. Viola and M. J. Jones, “Robust real-time face detection.” in *ICCV*, 2001, p. 747.

[15] G. Panin, C. Lenz, S. Nair, E. Roth, M. Wojtczyk, T. Friedlhuber, and A. Knoll, “A unifying software architecture for model-based visual tracking,” in *IS&T/SPIE 20th Annual Symposium of Electronic Imaging*, San Jose, CA, January 2008.