3D Position based Human Servoing System by Low-Level-Control of 6 DOF Industrial Arm

Suraj Nair, Emmanuel Dean, Alois Knoll,

***Abstract*—In this paper, we present a new vision-based multiple human tracking system. This novel $3D$ visual tracking system is capable of automatically identifying, labeling and tracking multiple humans in real-time even when they occlude each other. Furthermore, the multiple human tracker was implemented in a vision driven robot system for human robot interaction. The distributed system comprises of 4 subsystems: a)Multiple Human Tracking System, b) Robot Control System, c) 3D Visualization System and d) Remote Interface System. The Visual Tracking System performs real-time detection and tracking of humans in $3D$ within a large workspace. The Robot System uses the $3D$ position data of the targets obtained from the vision system to interact with the humans. The visual information is also used to monitor safe interaction within humans and robot. The Robot System is a $6DOF$ Stäubli TX90 industrial arm, controlled in real-time through a low-level interface. A real-time representation of the actual environment is rendered in $3D$ by the 3D Visualization System. The individual subsystems communicate with each other over a common communication engine based on TCP/IP. The complete system can be controlled and monitored through a wireless device.**

I. Introduction

The recent years have seen a rapid progress in computer vision and motion control technology. As a result many applications have evolved in this domain. Visual servoing is one such application which finds many use cases. It combines computer vision to visually track an object or target and robot motion control. The main idea behind visual servoing tasks is to control a robot system using visual information feedback [1, 8]. The target can be an object or a human, while the servoing device can be a simple pan-tilt unit or a complete industrial robot arm [8]. The reliability of the complete systems depends on both, the accuracy of the vision system and the robustness of the control approach. Special attention has been paid to *Human Robot Interaction*, where the vision system must provide the target's pose with high accuracy and the robot uses this data to achieve a specific task [6]. Depending on the properties of the target pose returned by the visual tracker, there are two main categories within visual servoing being, *image based* and *position based* visual servoing. In image based visual servoing the typical configuration is a camera mounted on a industrial robot arm [8]. The target, in this case a Human, is tracked using visual tracking approaches through the single camera image, providing the target's pose mostly in $2D$. This information is used by the robot controller to manipulate the camera position/orientation in order to hold the target within the camera's field-of-view at a defined perspective. This task is simple when there is only one target, but in real scenarios the robot system might need

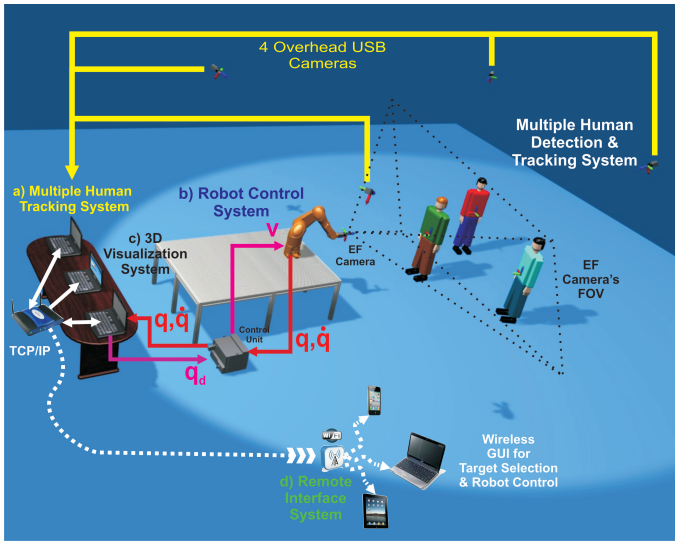

Fig. 1. Complete Robotic Setup. The figure shows the different systems involved in the robotic setup. a) the Multiple Human Tracking System: 4 USB cameras connected to a GNU/Linux OS PC, b) Robot Control System: An industrial robot StäubliTX90 and a CS8C control unit are connected to a GNU/Linux RT OS PC, c)3D Visualization System: OpenGL-based virtual world visualization running on a GNU/Linux PC and d) Remote Interface System: it allows the user to select the targets and control the robot.

to track more than one human at the same time. A single robot mounted camera cannot achieve this within a large workspace due to its limited field of view. The vision system must also handle object identification, labeling, and occlusions between targets in real-time, which is difficult using a single robot mounted camera.

In such circumstances a vision system capable of performing tracking of multiple human targets in real-time over a large workspace in $3D$ at all times is required. This information can be used by the robot system to servo all the targets within a large workspace. This approach is called position-based visual servoing since the targets position is computed in the Cartesian space.

In this article we present a novel position-based visual servoing system for multiple human targets over a large workspace. It uses a vision based $3D$ multiple human tracking system using externally mounted cameras and a 6 DOF industrial robot arm in order to visually servo the targets such as they are all visible in the field-of-view of the camera mounted in the robot end-effector. The vision based system uses ceiling mounted cameras in a stereo setup to track each human target in $3D$. The robot system uses information of the

targets positions to control the robot mounted camera such as all targets are visible in its field-of-view with a desired perspective. The robot mounted camera is not used by the vision system. The visual tracking system can react to new targets even if they are not in the field-of-view of the robot mounted camera. The system can also servo selected targets through the remote interface system, see Fig. 1.

II. Prior Art

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.

Multiple people trackers [7, 14, 10], have the common requirement of using a very little and generic off-line information concerning the person's 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 [16, 12, 13, 2] and employ a predefined shape and appearance model throughout the whole task.

In particular, [13] uses a standard particle filter with color histogram likelihood with respect to a reference image of the target, while [12] 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 [2].

The person tracking system [16] 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.

The work presented in [5], uses a template based approach. This method uses about 4,500 templates to match pedestrians in images. The Chamfer distance measure is used for similarity measure.

[3] combines target occupancy in the ground plane with color and motion models to track people in continuous video sequences. This approach requires heuristics to rank the individual targets to avoid confusing them with another.

III. System Overview

In our setup, we use 4 USB cameras mounted on the ceiling, sharing a common viewing region. They are calibrated for both intrinsic and extrinsic parameters with respect to a global origin. These cameras stream images of the tracking area at a rate of $25 - 35\ fps$. This cameras are integrated in a robotic setup. This setup is integrated with 4 different systems, a) the **Vision Tracking System**, b) the **Robot Control System**, c) the **3D Visualization System** and d) the **GUI Command Station**, see Fig. 1. The communication between all the systems is using TCP/IP protocol in a local network.

The organization of this article is as follows; in section IV, the *Human Tracking System* is described, followed by section V describing the *Robot Control System* and the trajectory planner. Section VI introduces the OpenGL based *3D Visualization Environment* and section VII describes the *Remote Interface*. The *Experimental Validation* of the complete system is illustrated in section VIII. Finally, in section IX the *Conclusions and Future Work* are given.

IV. Human Tracking System

The human tracking system automatically localizes and tracks humans in real-time within a desired working area. When a new target enters the tracking area the system identifies and adds him/her to a target list to be tracked. If a target leaves the tracking area, it is erased from the list of targets being tracked.

The target is modeled as a $3D$ rectangular box approximating to the dimensions of a human. The tracker holds a $3D$ state-space representation of the target's pose, given by a translation (x, y, z) within the observed workspace.

The tracker uses a bank of sampling-importance-resampling particle filters [9] working on a $3D$ motion model and an appearance model based on joint probability color histograms. Each target is associated with a unique particle filter. We choose a particle filter for the tracker over the more conventional Kalman Filtering [15] techniques because the tracker needs to be highly robust in dealing with multi-modal likelihoods due to cluttered background. The particle filter provides the sequential prediction and update of the respective $3D\ state$ $s = (x, y, z)$.

Particle filters are usually computationally intensive. A bank of particle filters increase computation cost with every new target. In order to achieve real-time performance, we maintain a global particle set and distribute it evenly among the bank of particle filters. Hence, if a new target enters the tracking area, the system instantiates a new particle filter and redistributes the global particle set evenly over the updated filter bank keeping the computation cost constant. This is feasible because, when the number of targets increase in the workspace their mobility reduces and the number of particles needed to track a target can be reduced. We rely on the particle filter bank approach over the the more conventional $MCMC$ filter [11] for handling multiple targets since the later maintains a global motion model, where as a bank of particle filters allows us to learn and control the motion model of each target individually.

Tracking multiple targets requires occlusion handling between targets in each camera view. This is important since when a target occludes another target in a camera view, that particular camera should be excluded during the likelihood computation for the targets which are being occluded. The reason being that, in the observation the region sampled will contain measurement data only for the target which occludes the other targets. However for successfully obtaining the $3D$

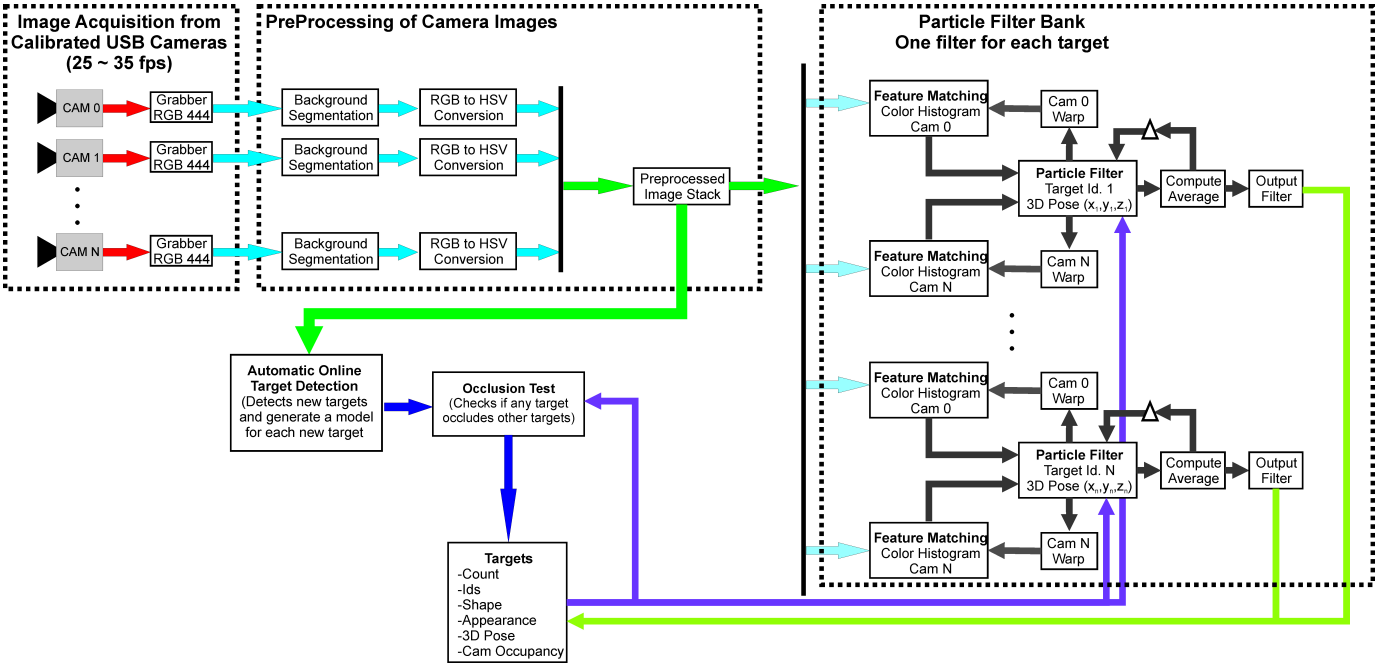

Fig. 2. The figure illustrates the block diagram of the tracking system. It consists of the image acquisition module, target detection module, occlusion test module and the particle filter bank.

pose of each target, it is necessary that the features be visible in at least 2 or more camera views. Our system handles occlusions between targets in real-time using a occlusion query module. It is also used during the target detection phase, which is important because when the system models the target the appearance information should be sampled only from the camera views in which the target is visible. This module will be discussed in greater detail in the sections to come.

Fig. 2 describes the complete pipeline of the tracking system. Each module is discussed in detail in the subsections below.

A. *Image Acquisition*

The sensors used are 4 USB cameras capable of streaming raw RGB 444 images of resolution (752×480) at the rate of $25 - 35 fps$. The cameras are arranged such that they observe a common tracking area with good overlap. Each camera is calibrated for its intrinsic and extrinsic parameters with respect to a global origin on the tracking area floor. The 4 cameras are connected to a single PC with 2 dedicated USB controllers. The cameras operate in streaming mode where images are written into the memory continuously. When a request arrives for an image update, the latest image from the camera buffer is returned to the tracker.

B. *Pre-processing of sensor images*

The sensor images obtained Img_i from the image acquisition system, where the index i corresponds to the USB camera index, undergo a two stage pre-processing. In the first step background segmentation is performed on each camera image using a static background model. The background segmented image from each camera is then converted from RGB to HSV for the color-based likelihood, where the index i corresponds to the USB camera index.

$$Ibg_i = bgSub(Img_i) \tag{1}$$

$$z_i = rgb2hsv(Ibg_i) \tag{2}$$

The pre-processed images are available at both stages since the on line target detection module only requires the background segmented image while the tracker requires the pre-processed image resulting after both stages are performed.

C. *On Line Target Detection*

This module automatically detects targets when they enter the tracking area by performing a scan along the tracking floor area using the $3D$ box target model. At each location in the scan the probability of a possible target detection is computed using the background segmented image. The number of foreground pixels are computed within the $2D$ region obtained by warping the $3D$ *pose* of the target model on the respective camera images. Regions occupied by existing targets are not considered. If foreground occupancy of at least 70% is observed in each camera view then a target is registered with an initial $3D$ *pose* of the particular scan location. A occlusion test if performed at the target location in order to identify the cameras in which the target is completely visible. Using this information the shape and appearance model of the target is generated. The shape consist of a $3D$ rectangular cube with dimension of a normal sized human $(0.2m \times 0.3m \times 1.8m)$. The appearance model consists of $2D$ histograms of the targets in the HSV color space. Only cameras in which the target is visible are selected. If the target

is occluded in a camera view, the appearance model in that view is suspended until the target is visible in that camera view. In order to register a target, we require that it be visible in at least 2 camera views. The target data consists of:

- unique $Target\ ID$
- initial $3D$ $pose$
- shape data
- appearance data
- occupancy information depending on occlusion test
- current $3D$ $pose$

D. Occlusion Testing

This module determines if a target is occluded by any other targets. This is very important during target detection and tracking since we use $2D$ regions in the camera views obtained by warping the $3D$ $pose$ of the hypothesis under consideration. When a target occludes other targets in a camera view, the warped $2D$ regions are overlapped making the appearance data visible only for the target which occludes the other target. In such situations, the information from these $2D$ regions should not be sampled for the targets which are occluded. The occlusion test provides information of such occlusions.

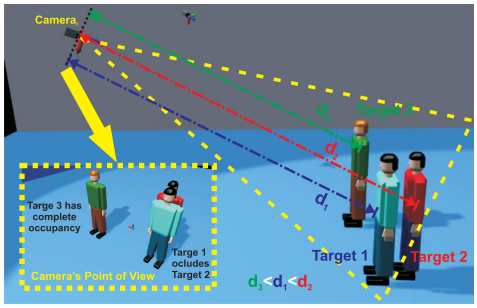

Fig. 3. The figure illustrates the occlusion test system. The left part of the figure shows a scene in a camera view with 3 targets, where target 1 occludes target 2. The right part of the figure shows how the occlusion is detected by rendering the targets with respect to their distance from the camera The target closes to the camera is rendered first.

Fig. 3 illustrates the occlusion test system. This system considers all the targets and computes their occupancies in each camera image. For each camera view the euclidean distance from the camera to each target is computed. The target which is farthest from the camera is rendered first on the camera image. This is followed by the remaining targets, where the closest target is rendered last. Once all targets are rendered an overlap test is conducted. If a target occludes another target it will overlap the rendered region of that target. For a target, if more than 70% of the rendered region remains non-overlapped, then its considered to have a good occupancy in the current view of this particular camera. Thereby, while tracking the filter associated with a particular target considers only the camera views in which the target is not occluded. For a target to be tracked it is required to be visible in at least 2 or more camera views.

E. Tracker

The tracker has the primary goal of keeping a track of all the targets in real-time, once they have been registered by the target detection system. In order to do this, the tracker uses a bank of Sampling-Importance-Resampling based Particle filters[9]. Each target is associated with its own particle filter. Each filter uses a Brownian motion model and a $3D$ translation state. The visual modality used is $2D$ color histograms and the likelihood estimation is performed by computing a distance measure between the histogram sampled from the current hypothesis and the reference histograms. For each hypothesis, the likelihood is computed for each camera view, in turn computing an average likelihood. Camera views in which targets are not visible are dropped during the likelihood computation for the respective targets.

Particle filters are computationally expensive and hence in order to obtain real-time performance from a bank of particle filters, we maintain a common global particle count which is distributed evenly among the filters. This distribution depends on the number of targets. When a target is added or removed from the target list, the number of particles allocated to each filter is updated. Hence, if N_p is the global particle count and n_p is the number of particles allocated to each filter, then

$$n_p = \frac{N_p}{N} \tag{3}$$

where, N is the number of targets. This approach is well suited since as the number of targets increase within the tracking area, their mobility reduces and hence the number of particles needed to track them can be reduced. The following subsections provide detailed explanation of the functioning of the particle filter.

- **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.

$$s_t^i = s_{t-1}^i + v_t^i \tag{4}$$

  with v a zero-mean Gaussian white noise of pre-defined covariance in the (x, y, z) state variables. The motion model can be controlled during the course of tracking by learning the motion of the target. Deterministic re-sampling strategy over the previous weights w_{t-1}^i is also employed.
  For each generated hypothesis, the tracker asks for computation of the likelihood values $P(z^{col}|s^i)_n$ after projecting every hypothesis on to each camera image.
- **Color likelihood** The object model defining the targets shape is projected on the pre-processed image of each camera image at the predicted hypothesis s_t^i using the intrinsic and extrinsic parameters of the respective cameras. The underlying H and S color pixels are collected in the respective 2D histogram $q\left(s_t^i\right)$, that is compared with the reference one q^* through the Bhattacharyya coefficient

[13]

$$B_m\left(q_i\left(s\right),q_i^*\right)=\left[1-\sum_N\sqrt{q_i^*\left(n\right)q_i\left(s,n\right)}\right]^{\frac{1}{2}} \quad (5)$$

where the sum is performed over the $(bin \times bin)$ histogram bins (in the current implementation, $bin = 10$). The computation is done for each camera where c represents the camera id $(c = 1...M)$.
The color likelihood is then evaluated under a Gaussian model in the overall residual

$$P(z^{col}|s_t^i) \propto exp(-\prod_M(B_i^2/\lambda)) \quad (6)$$

with given covariance λ.

- **Computing the estimated state**
  The average state $\overline{s}_t$

$$\overline{s}_t=\frac{1}{N}\sum_i w_t^i s_t^i \quad (7)$$

is computed and the three components $(\overline{x},\overline{y},\overline{z})$ are returned. In order to reduce the jitter in the output, the average pose is smoothed using an exponential filter.

F. *Graphical User Interface*

The tracking system can be effortlessly controlled by the user using an intuitive graphical user interface as shown in fig. 4. Although the system can be operated automatically, the GUI provides useful functions such as start, stop, background training, etc. The complete tracking scene from all the camera can be visualized by the GUI. The scene rendering is done using widgets with capability of rendering into OpenGL contexts.

Fig. 4. Graphical user interface for controlling the system and visualizing the tracking results

In order to validate the **Multiple Human Tracking System**, we implemented a complete Human-Robot-Interaction scenario, where the position of each target is needed to achieve a specific task. The control flow of the vision-based robotic system is illustrated in Fig. 5, where the *Vision Tracking System* get the position of each target and sends them to the *Robot Control System*. This module computes the joint position reference vector and sends it to the control unit, which in turn feeds back the current joint positions of the robot. The *Robot Control Unit* updates the *3D Visualization* environment using the real joint positions. Each system is explained in the following sections.

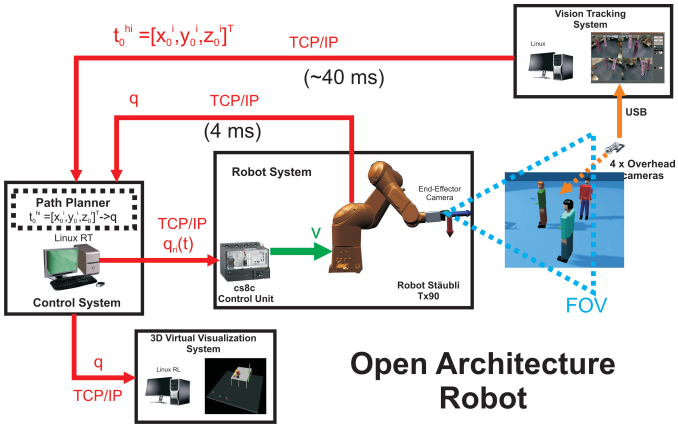

Fig. 5. Human Tracking System Block Diagram

V. Robot Control System

The robotic system comprises of a Stäubli TX90 industrial robot arm, a CS8C control unit and a Workstation running on GNU/Linux OS with real-time extension, see Fig. 5. The data communication between the PC and the control unit is in a local network based on TCP/IP. In order to open the architecture of the industrial robot a C++ library based on Stäubli LLI was written [4]. This library allows the user to command the robot joint positions or the torque values for each motor drive. In this article, the joint positions was used to command the robot. This joint positions $q_r\left(t\right) \in \Re^n$ are obtained using the next trajectory planning.

A. *Trajectory Planning*

The **Human Tracking System** provides the position of each target $t_0^{h_i}=[x_0^i,y_0^i,z_0^i]^T$ with $i=1,2,...,m$ where m is the total number of targets, see Fig. 5. The trajectory planner uses this data to generate the desired joint positions $q_d \in \Re^n$. Given the kinematic decoupling properties of the Stäubli robot, the task can be divided in two phases, the first being the *Pan and Tilt* control of the eye in hand camera, and the second is to set the camera's *Field of View* (**FOV**).

B. *Pan and Tilt*

Using the Denavith-Hartenberg convention, the *pose* of the robot's wrist is given by, $T_0^3=\begin{bmatrix} R_0^3 & t_0^3 \\ 0^{1\times3} & 1 \end{bmatrix}$, where $t_0^3=\left[x_0^3,y_0^3,z_0^3\right]^T \in \Re^{3\times1}$, and $R_0^3=\left[X_0^3,Y_0^3,Z_0^3\right] \in SO\left(3\right)$ represent the position and orientation of the wrist wrt the world coordinate frame. In the same manner, T_0^6, T_0^c and $T_0^{h_i}$ are the *pose* of the robot's end-effector , the eye in hand camera and the target i, respectively.

In order to calculate the orientation of the camera T_0^c, the average target position is needed.

$$\widetilde{P}=\frac{1}{m}\sum_{i=1}^{m}t_0^{h_i}. \quad (8)$$

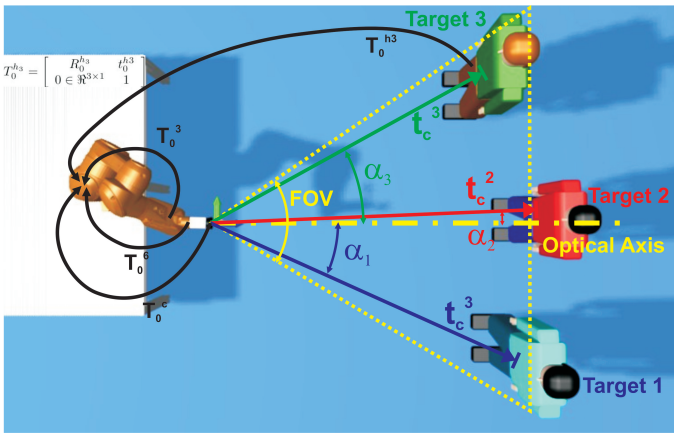

Fig. 6. Camera's Field of View and the Eye in Hand camera.

Then, the position error vector between t_0^3 and $\widetilde{P}$ is computed

$$\Delta P = t_0^3 - \widetilde{P}. \tag{9}$$

The vector in eq.(9) will define the orientation of the camera using its direction cosines in a general rotation matrix,

$$R_0^c = R_x(\alpha) R_y(\beta) R_z(\gamma) \in SO(3). \tag{10}$$

Where $R_k(\theta)$ is the basic rotation matrix around the k axis through an angle θ.

The robot's direct kinematics can be used to define the camera's orientation in terms of the *Wrist's Orientation* R_0^3 and the *End Effector Orientation* R_3^6,

$$\begin{aligned} R_0^c &= R_0^3(q_1, q_2, q_3) R_3^6(q_{d_4}, q_{d_5}, q_{d_6}) \\ R_3^6 &= R_0^3(q_1, q_2, q_3)^T R_0^c \end{aligned} \tag{11}$$

The solution of eq.(11) generates the desired $q_{d_4}, q_{d_5}, q_{d_6}$, and depends on q_1, q_2, q_3. Now, to compute these first joints, the desired wrist position t_0^3 must be provided, which depends on the camera's field of view.

C. Camera's FOV

For this task, two facts affect the position of the wrist: a) the optical axis normal to the targets[1] and b) the camera's FOV. For the first part, the solution of $q_{d_1}, q_{d_2}, q_{d_3}$ is derived from eq.(10) with $R_0^3 = R_0^c$. This motion is implemented only when $q_{d_5} > q_{5_{max}}$, where $q_{5_{max}}$ is defined by the user. In the second part, the camera's FOV must be fixed. Fig. 6 shows the relation of the camera's FOV and the angle of each target, where $t_c^i = t_0^c - t_0^{h_i} = [x_i, y_i, z_i]^T$ represents the position vector of the target i wrt the camera frame. α_i is the angle between t_c^i and the *optical axis* given by Z_0^c.

Then, the optimal wrist position $\left\{t_0^3 \in \Re^3 | \alpha_i < \frac{FOV}{2}, \forall i = 1, 2, ..m\right\}$ must be computed. This solution is similar to calculating $\alpha_{max} < \frac{FOV}{2}$ with $\alpha_{max} = max(\alpha_i)$.
Then, to find the solution the next steps must be followed:

[1]The idea is to keep the eye in hand camera in front of the targets.

1) Compute $\alpha_i = a\cos(|z_0^c| \cdot |t_c^i|)\ \forall i = 1, 2, ..., m,$.
2) if $\alpha_i > \frac{FOV}{2}$, then:
   a) compute distance from camera to target i,
   $$d_i = \|t_c^i\|_2^2,$$
   b) compute minimum distance,
   $$d_{\min} = d_i \cos\left(\frac{FOV}{2}\right), \tag{12}$$
   c) compute the projection of target i over the *optical axis* Z_0^c,
   $$d_{z_i} = d_i \cos(\alpha_i), \tag{13}$$
   d) compute error distance,
   $$\Delta d = d_{\min} - d_{z_i}, \tag{14}$$
   e) then,
   $$t_0^3 = t_0^3 - \Delta d Z_0^c, \tag{15}$$
3) if $t_0^3 > t_{\min}$, $t_0^3 = t_{\min}$, where $t_{\min}$ is a safety threshold, to avoid collisions with the robot body.
4) Finally, use t_0^3 and the *Inverse Kinematics* to obtain $q_{d_1}, q_{d_2}, q_{d_3}$.

D. Path Planning

Once the desired $q_{d_i}, i = 1, 2, ..., n$ have been set, the trajectory from the current position $q_0 \in \Re^n$ to the desired position q_d must be established. In this case, a 5th order polynomial function has been used,

$$q_{r_i}(t) = (f_{5_i}(t)(q_i - q_{0_i})) + q_{0_i} \tag{16}$$

$$f_{5_i}(t) = a_1\left(\frac{t - t_0}{t_{fi} - t_0}\right)^3 + a_2\left(\frac{t - t_0}{t_{f_i} - t_0}\right)^4 + a_3\left(\frac{t - t_0}{t_{f_i} - t_0}\right)^5 \tag{17}$$

where $t \in \Re$ is the current time, $t_0 \in \Re$ is the initial time and the final time is $\{t_{f_i} \in \Re | \ddot{q}_{d_i}(t) < a_{\max}, \forall i = 1, 2, ..., n\}$ with a_{max} as the maximum joint acceleration defined by the user. This q_{r_i} is transmitted to the control unit in real time, see Fig. (5).

VI. 3D Visualization System

This module performs $OpenGL$ based real-time rendering of the workspace in $3D$. The scene is constructed using $3D$ models of different objects occupying the scene such as the robot, controller box, table and the human target locations. The system updates the configuration of the robot arm and the positions of the human in real-time. Fig. (7) illustrates this module.

VII. Remote Interface System

Each individual system can be controlled through a remote device supporting $WiFi$ interface. Therefore devices such as iPhones, Tablet PCs, Net-books. etc can be used to remotely control and monitor the entire system. The remote interface exchanges data with each subsystem through TCP/IP.

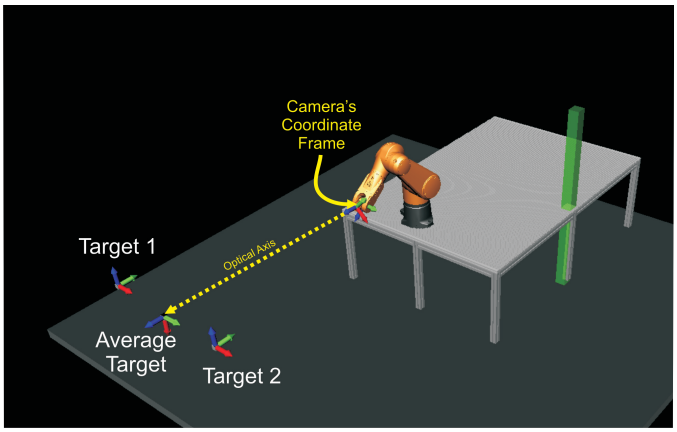

Fig. 7. *OpenGL* based virtual visualization

VIII. Experiments and Test

To test the performance of the vision system we need precise ground truth data of the target's position. In order to achieve this, we designed a $3D$ scene of the complete workspace as illustrated in Fig. (1). In this scene, the motion of two human targets were simulated in order to obtain the ground truth data of their positions in each frame with respect to the global origin. The whole animation was captured into video sequences from the perspective of the 4 cameras used by the vision tracking system. These sequences were used to test the tracker and validate its performance and accuracy as illustrated in Fig (8).

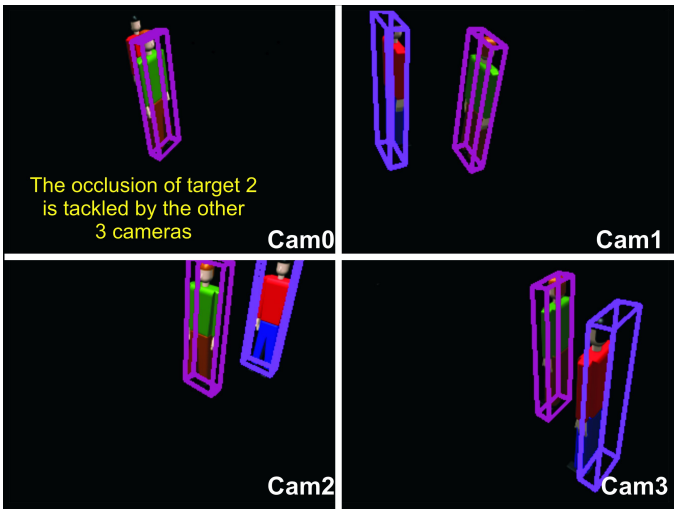

Fig. 8. Tracking results on video sequences obtained from the $3D$ animation of the workspace where the motion of 2 human targets is animated.

Fig (9) illustrates the accuracy of the tracking results with respect to the ground truth. The pose of the target is observed in X and Y while the displacements in Z remains fairly constant as the targets are moving on a horizontal floor. The variance of tracking result was approximately $8cm$ in X and $1.1cm$ in Y for both targets. The error is higher in the X direction since the motion of the targets was more in the X direction in the video sequence.

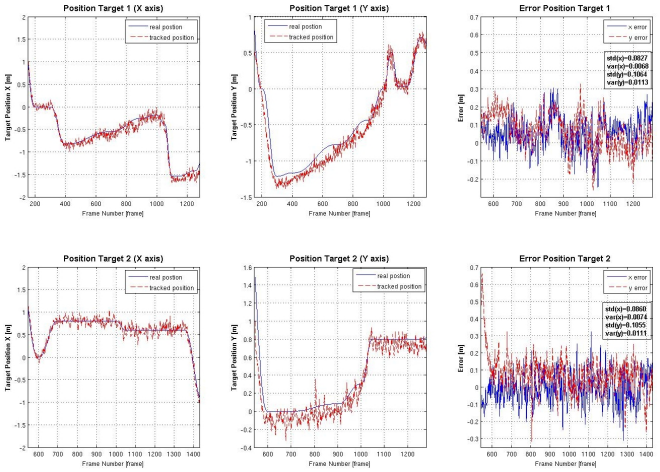

Fig. 9. Tracking Results with respect to the ground truth using the animated scene. It shows the tracking results for the two target in X and Y along with the standard deviation and variance of the result.

Fig. 10 demonstrates the results obtained in the real-world scenario. The tracker tracks 2 targets simultaneously in real-time and the robot arm servos both targets. Later the operator disables target 2 such that the robot arms servoyes only target 1 followed by target 1 being disabled and target 2 being enabled. Later both targets are enabled and it is observed that target 1 gets closer than the safety limit of the robot which is detected by the tracking system and a signal is sent to the robot controller so that it goes to a safe parking position and thereafter all systems are shutdown.

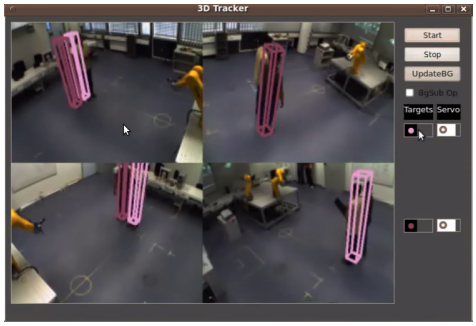

Fig. 11. The figure illustrates how the system detects occlusion between targets.

Fig. 11 shows how the system handles occlusion of targets by other targets. It can be seen that in camera 1 (top right), target 1 is occluded by target 2. Hence, during the likelihood computation for the filter associated to target 1, camera 1 is not considered. Similarly in camera 3 (bottom right), target 2 is occluded by target 1 and therefore camera 3 is not considered in the likelihood computation for the filter associated to target 2.

The visual tracking system runs at a speed of approximately $15fps$ on a Intel Core i7 desktop PC. A complete video

Fig. 10. The figure illustrates the test results. The 4 clustered images represent the tracking system results along with an additional robot mounted camera output. **Top Left:** Two targets are tracked and servoyed by the robot. **Top Right:** Only target 1 is enabled to be servoyed by the robot. **Bottom Left:** Only target 2 is enabled to be servoyed by the robot. **Bottom Right:** Target 1 gets closer than the safety limit of the robot and robot goes to park position and all systems are shutdown.

demonstration is available at http://www.youtube.com/watch?v=4mGXupIY-xU

IX. Conclusion and Future Work

In this article we have presented a novel real-time human tracking system which exhibit high accuracy. This accuracy allows to use this tracking system within a Human Robot Interaction scenario. In order to validate the human tracker a complete robotic system was implemented. The results obtained in the experiments shows the reliability of the tracker and its potential applications. The human tracking system can handle multiple targets and occlusion with each other using only camera information. The visual information obtained from the 4 cameras is used to compute the desired joint positions of the industrial robot.

We plan to continue the developing the system further by implementing fusion of multiple visual cues in the vision system in order to improve the robustness. In order to improve the performance we intend to map computation intensive parts of the system to the GPU. We also intend to implement different applications in order to validate the further use cases of the system.

References

[1] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *Robotics Automation Magazine, IEEE*, 13(4):82–90, 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.250573.

[2] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, 2003.

[3] Jerome Berclaz Francois, Jérôme Berclaz, François Fleuret, and Pascal Fua. Robust people tracking with global trajectory optimization. In *In Conference on Computer Vision and Pattern Recognition*, pages 744–750, 2006.

[4] Thomas Friedlhuber, Kai Klimke, Markus Rickert, and Alois Knoll. Echtzeitsteuerung eines stäubli industrieroboters über tcp/ip. Technical report, Technische Universitaet Muenchen, March 2007.

[5] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *Proc. of European Conference on Computer Vision*, pages 37–49, Dublin, Ireland, 2000.

[6] Michael Goodrich and Alan Schultz. Human–robot interaction: A survey. *Foundations and Trends in Human–Computer Interaction*, 1(3):203–275, 2007.

[7] 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*, page 962, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8497-6.

[8] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on*, 12(5):651–670, October 1996. ISSN 1042-296X. doi: 10.1109/70.538972.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision (IJCV)*, 29(1):5–28, 1998.

[10] Michael Isard and John MacCormick. Bramble: A bayesian multiple-blob tracker. In *ICCV*, pages 34–41, 2001.

[11] Zia Khan. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(11):1805–1918, 2005. Member-Tucker Balch and Member-Frank Dellaert.

[12] Katja Nummiaro, Esther Koller-Meier, and Luc J. Van Gool. An adaptive color-based particle filter. *Image*

Vision Comput., 21(1):99–110, 2003.

[13] Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 661–675, London, UK, 2002. Springer-Verlag.

[14] Nils T. Siebel and Stephen 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*, pages 373–387, London, UK, 2002. Springer-Verlag.

[15] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, 2004.

[16] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.