Bringing Visual Servoing into Real World Applications

Mona Gridseth Camilo Perez Quintero Romeo Tatsambon Fomena Oscar Ramirez Martin Jagersand

University of Alberta

***Abstract*— Substantial work has been published in visual servoing over the years, yet we have seen few real world applications. We discuss four different milestones (human robot interaction (HRI), tracking, multi-view geometry and visual servoing) needed to bring visual servoing into an unstructured human environment. We focus our efforts on HRI and tracking which we think are the link to bring visual servoing towards more practical applications.**

Keywords: Visual Servoing, Human Robot Interaction, Visual Task Specification, Tracking.

I. Introduction

Visual Servoing is a well-known technique used to control a dynamic system using visual feedback [7]. Despite much published research on visual servoing, it has had little penetration in real-world applications. With its roots in engineering, robotics research has focused on topics such as mechatronics design, control and autonomy, while fewer works pay attention to human-robot interfacing and user studies. This results in an increasing gap between expectations of robotics technology and its real world capabilities. Our aim is to bring visual servoing towards more practical and usable implementations. In order to create a successful visual servoing system, four areas need to be addressed: HRI, tracking, multi-view geometry and visual servoing.

We illustrate these four points with a real-world example. Consider a surgical setting where the surgeon wants to make an incision along the midline of the patient (Figure 1). The surgeon draws a line on the patient's body to mark the location of the incision. Either the incision can be performed by a robot arm that follows this line or the surgeon can hold the scalpel collaboratively with the robot arm in a haptic interaction. The surgeon moves the scalpel forward or backward, while the robot constrains the movement to follow the indicated line.

A good interface for HRI is necessary to provide the surgeon with an intuitive way to specify the line that the robot should follow. Furthermore, the robot end-effector as well as either the line on the patient or the end-points of that line must be tracked. This is because the patient can move during the procedure and depending on the placement of the cameras for visual servoing (eye-in-hand and/or eye-to-hand), the end-effector or the surrounding world will move in the images. Multi-view geometry is used to turn the image information from the line the surgeon specified into an error function for visual servoing. Finally visual servoing is used to move the robot arm.

Human Robot Interaction

As illustrated with the above example, HRI is fundamental

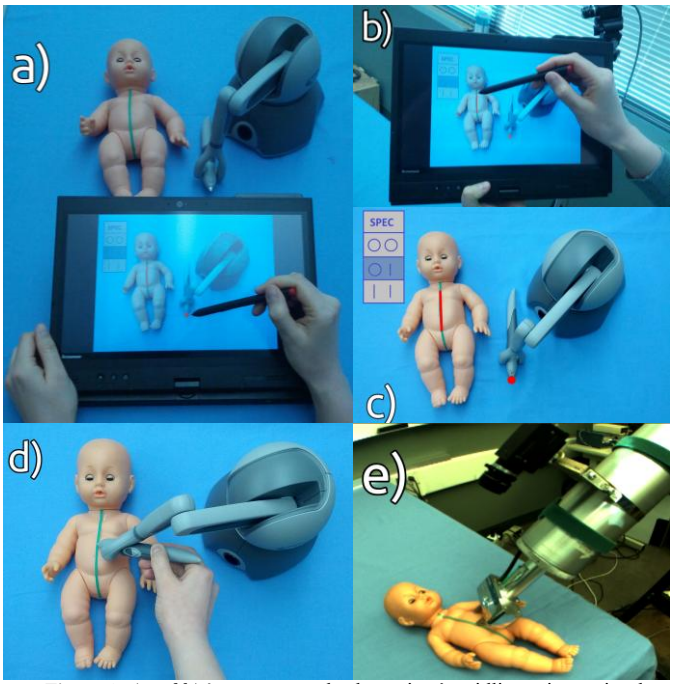

Figure 1. a) and b)A surgeon marks the patient's midline using a visual interface. c) A view of the interface. d) Haptic interaction to perform incision. e) Visual servoing to perform incision.

in any robotics real world application. Even for autonomous routines the user needs an interaction to start and stop them. Today robot arms are deployed and work successfully in controlled environments such as factories, where they can be pre-programmed to complete a set of fixed tasks autonomously and accurately. However, when we want robots to aid us in our everyday lives or in the workplace the challenge becomes greater due to unstructured and dynamic surroundings. Robotics researchers are looking to tackle this problem by letting humans and robots work together to complete tasks [1, 2, 3]. This can be done by keeping the 'human-in-the-loop' [3], which means the human guides the robot without controlling it completely.

One commonly used technique in robotics is 'teach-by-showing' [7] (Figure 2). The robot is manually moved to its goal position and that location is recorded. Afterwards the robot can perform the same task autonomously. This works for repetitive tasks in a controlled environment, but not for semi-autonomous control. In the particular case of visual servoing, tasks can be visually specified in images. A task can be defined as the objective of bringing the end-effector of the robot arm to its target in the work space [4]. For example a point-to-point

task aligns a point on the robot end-effector (or an object the end-effector is holding) with a point in the workspace, whereas a point-to-line task brings a point on the end-effector to a line in the workspace (Figure 2). The formal relationship between feature alignments in 2D image frames and the 3D world has been explored theoretically [4, 5, 6], but not seen much use otherwise. We will exploit these visual specifications in the HRI we present in Section II. By allowing the user to specify tasks visually we can keep the 'human-in-the-loop' interaction.

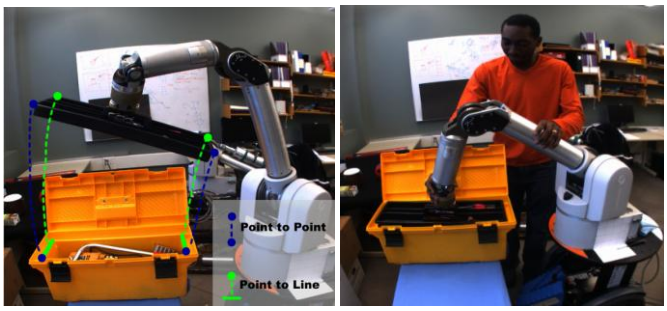

Figure 2. The left image shows visual tasks (point-to-point and point-to-line) needed to perform visual servoing to move the tray into the box. The image on the right illustrates 'teach-by-showing' for task specification.

Tracking

Tracking is essential in visual servoing as it provides the location of the target object and the robot end-effector. In particular for visual task specification we may need to track several points in the image frame. In the case of a point-to-line task we need to track a point on the end-effector and two points on the line in the workspace. Alternatively we can combine point tracking and line tracking. In many visual servoing papers tracking is done simply by extracting the position of black and white markers on the robot and target object. This approach will not work in a real-world application since we will not have access to marked objects. Familiar types of tracking algorithms include segmentation-based tracking, feature-based tracking and registration-based tracking [10, 11, 12]. Any tracking algorithm used for visual servoing needs to have good accuracy, be able to handle large camera motion and it must be robust.

Multi-View Geometry

From the user interface we get the tasks that the user has specified and tracking provides the location of the relevant points. In addition to specifying a task, we also need to verify whether the task has indeed been accomplished by the robot arm after visual servoing. Both the specification and verification of tasks is based on data from 2D images. We need a way to decide whether convergence to the goal in the image space actually means that we have reached the target in the robot work space. Dodds [6] shows that there is visual ambiguity in 2D images as illustrated in Figure 3. Although the two camera views show the probe touching the wire, this does not actually happen.

The "teach-by-showing" technique mentioned earlier is also relevant to task specification. The robot arm is moved to the goal position and the corresponding image features are recorded. We know the task is completed when the error between the current image features and the recorded ones has reached zero. However this approach does not eliminate the problem with image ambiguity.

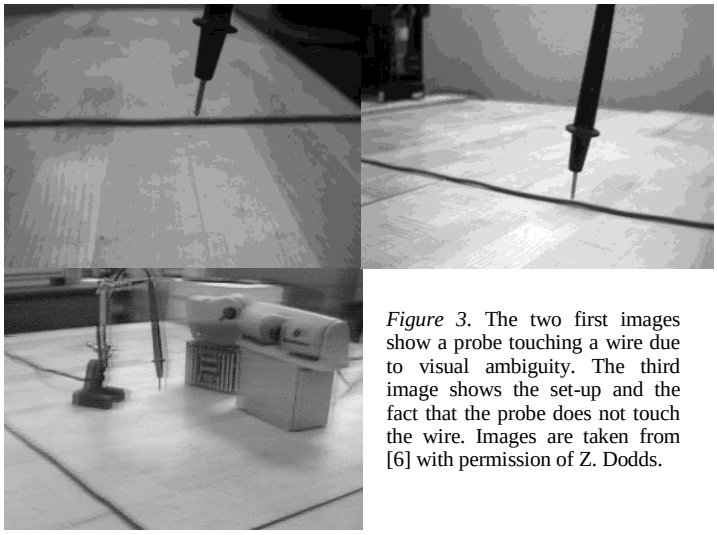

Figure 3. The two first images show a probe touching a wire due to visual ambiguity. The third image shows the set-up and the fact that the probe does not touch the wire. Images are taken from [6] with permission of Z. Dodds.

Depending on the level of calibration there exist tasks that can be unambiguously specified and verified using two 2D image views, they are said to be decidable tasks [4, 5]. One example is a point-to-point task, which is decidable for uncalibrated systems. Furthermore Dodds *et al.* [5] show that there exist operators that preserve task decidability. Decidable tasks can therefore be composed into more complex tasks that are still decidable. That is, we can decompose high-level actions into a string of simpler tasks that we know are decidable.

Visual Servoing

We use visual servoing to move the end-effector of a robot arm to its target in the workspace. The inputs to visual servoing are images from two or more cameras. From these images we collect features that are used to create an error function. As we will see in Section II, when working with visual tasks this error function is constructed using the specified task. Visual servoing is accomplished by driving the error function to zero. One possible approach is to compute the error dynamics differential equation and use the result to create a control law. Depending on the level of system calibration, different types of visual servoing exist. We focus on image based visual servoing that uses geometric transforms going directly from projective image space to the robot motor space [8].

Building on results from visual servoing and multi-view geometry we will focus on developing intuitive HRI to facilitate effortless control of and interaction with a robot arm. We will also discuss in more detail a tracking approach that we have explored.

II. COMPUTER VISION FOR HRI

A. *Interfaces and Input Types*

An important milestone for bringing visual servoing to real world applications is creating a system for effortless and intuitive HRI. This means we need to match the right interface

with different types of users and applications. Part of our research consists of developing, classifying and testing interfaces. In this section we describe several cases where visual input is used to specify tasks for the robot to carry out. Following we consider interaction using (1) a touch screen, (2) a joystick and (3) pointing.

Going back to the surgical setting from Figure 1, the surgeon has to specify the midline on the patient. A touch screen tablet together with a stylus provides easy interaction with the visual servoing images. First the surgeon chooses the type of task from a drop down menu. In our example this is a point-to-line task. Next the surgeon clicks on the end-effector to indicate a point, and then on two more points in the workspace to mark the patient's midline. A red point and line in the image shows the selection. Once told to start, the robot holding the scalpel moves along the specified line and makes the incision. Alternatively, using haptic interaction, the robot and the surgeon hold the scalpel together. The surgeon moves the scalpel forward to make the incision, while the robot implements forces to make sure the incision does not deviate from the midline.

Robot arms can be of great assistance to wheelchair users with low arm function. A light-weight robot arm such as the Kinova JACO arm [9] can be attached to a wheelchair and used to grasp and manipulate objects. For many wheelchair users a touch tablet and stylus is not a feasible mode of interaction because they may suffer from muscular spasms or hand tremors. A better approach would be to use a joystick to control a cursor on the screen. This way the user can specify visual tasks as described before. When picking up an object the user would first specify a point-to-point alignment to move the robot arm close to the desired target. In order to grasp the object from the right angle the gripper would need to be re-oriented and re-aligned. This fine-tuning of the action can be done by specifying one or more additional visual tasks. A detailed example follows in the next section.

The final mode of interaction is pointing [10] (Figure 4). The user points towards an object he/she wants the robot arm to manipulate. This takes advantage of a natural interaction that humans use all the time. After the user has selected an object the robot arm would move there and grasp it. Furthermore the user can point to a location where the object should be put down. In the two previous applications the user interface gives clear feedback to the user by drawing points and lines on the screen. If the specification appears incorrect, he/she can clear the selection and try again. With a pointing interface, feedback is not as straight forward. We would let the robot point towards the object closest to the location chosen by the user. The user can then confirm or decline this selection with a gesture.

A system for pointing and gesture recognition can be set up in two ways. In the first approach the user is observed with two cameras. The line extended from the users arm to the object he/she is pointing to can be projected to lines in the two images. The object location in the left and right images is combined to perform visual servoing for the robot arm. Another possibility is to make use of 3D data replacing the two cameras by a Microsoft Kinect sensor. The 3D point that the user indicates is transformed to the robot base frame. Now the robot end-effector can move to this point after an inverse kinematics calculation.

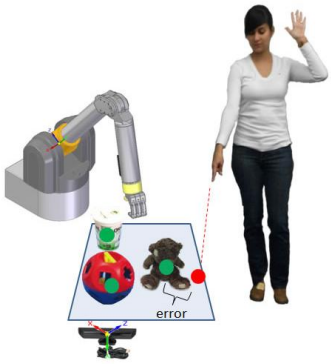

Figure 4. Illustration of an interface where the user points to objects for the robot arm to grasp. The pointing location is found using a Microsoft Kinect Sensor. There will be some error between where the user points and the location of the target object.

B. Composition of Tasks

We want to take high-level actions for the robot to perform and decompose them into a chain of simpler tasks that we know can be completed. As mentioned in the introduction, Dodds *et al.* [5] show that there exist task operators that preserve task decidability. They can be used to compose decidable tasks into more complex ones. This is similar to the object grasping in section II-A where we suggested to first move close to the target before re-orienting the gripper. These simple tasks can be specified in serial one after another with the robot completing each task before the next is specified. Another approach is to specify several tasks in parallel before the robot carries them out. This is useful when one task alone is not enough to fully determine the desired pose (position and orientation) of the end-effector; we can add more constraints by adding more tasks. An example is illustrated in Figure 5. We want to insert the yellow hexagon into the box. First the gripper is brought close to the hexagon with a point-to-point task. Then two alignments are performed to grasp the hexagon. Next the gripper is brought close to the box with another point-to-point task and finally a set of new tasks are specified to align the hexagon with the box opening. As an example of parallel specification we will describe the tasks needed for the grasping in more detail.

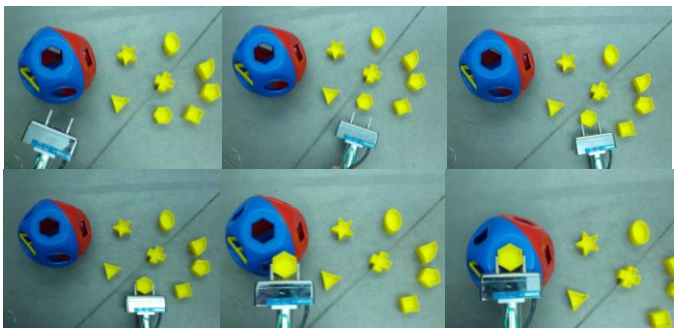

Figure 5. Inserting a hexagon into the box is done in several steps. First the gripper moves close to the target. Two steps are needed to align the gripper and grasp the hexagon. Finally the hexagon is brought close to the box and inserted after further alignment.

In order for the robot grasp the hexagon we need to specify several tasks (Figure 6) and turn that information into an error function for visual servoing. This requires an image encoding

for the tasks. Image encodings are described in detail in [4, 5]. Assume we have the image coordinates of the required points and lines from tracking. By specifying two point-to-line tasks between p2 and L1 and between p4 and L2, we can align the

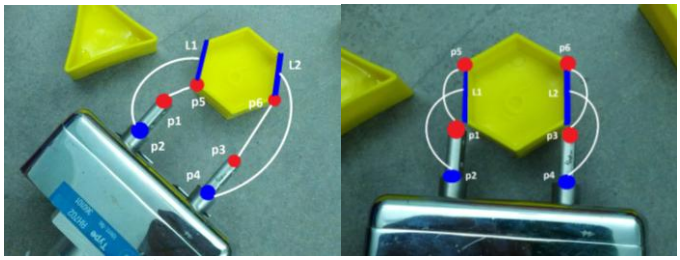

Figure 6. The left image illustrates two point-to-point tasks and two point-to-line tasks needed to align the gripper with the hexagon and bring it closer to the hexagon. They are used to create E_1. The right shows two new point-to-point tasks and the same point-to-line tasks. It shows the goal configuration for E_1, and the specification of a new visual goal E_2 that will slide the robot gripper onto the hexagon.

gripper with the hexagon. The point-to-point tasks between p1 and p5 and between p3 and p6 will decide how far along the specified lines the gripper should move (Figure 6, left image). Assume we are using two cameras, the subscripts L and R indicate image coordinates in the left and right camera view respectively. Then the image encoding for one of the point-to-point tasks is $Epp = (p5_L - p1_L, p5_R - p1_R)^T$. Similarly for one of the point-to-line tasks the encoding becomes $Epl = (p2_L \cdot L1_L, p2_R \cdot L1_R)^T$. One of the operators presented in [5] that preserve task decidability is the AND function. The AND function combines several tasks that all have to hold simultaneously by stacking their image encodings into one vector E. Hence to specify the above tasks form the left image in Figure 6 in parallel, we stack all of them into a vector E_1. In fact it is this vector E_1 that becomes the error function needed for visual servoing. Once E_1 has been driven to zero by visual servoing, we are in the situation displayed in the right image in Figure 6. Now we create E_2 that specifies the tasks needed to complete the grasp. The point-to-line tasks stay the same to preserve the gripper orientation, but the point-to-point tasks change to allow the gripper to move forward. Visual servoing will finally bring E_2 to zero and the grasp is completed.

Given that we have an imprecisely modelled two-camera system; can we say something about what tasks can be accomplished solely based on observing point features? According to [4, 5] the answer depends on the camera model and level of calibration. Point coincidence tasks are decidable on any family of injective two-camera models and these are also the only tasks decidable on injective camera models. Furthermore for weakly calibrated projective cameras, a task T is decidable iff T is projectively invariant. Finally if T is decidable for uncalibrated cameras, then T is projectively invariant. This means that we can verify a robot arm positioning task with absolute certainty using weakly calibrated, noise free, stereo-vision systems iff the task is projectively invariant. However with point coincidence tasks this can be done with an uncalibrated system.

Using results from visual task specification and multi-view geometry we can develop different computer vision based HRI interfaces that will bring us closer to using visual servoing in real-world applications for a wide range of users.

III. TRACKING

Visual tasks that are specified by the user have a set of points associated with them. These points must be tracked. In the case of a point-to-line task, we need to track a point on the end-effector of the robot as well as the line in the workspace. For the latter we can use a line tracker or simply track the two end-points. The tracking is necessary since the object we are working with might move. Also, depending on the placement of the cameras we are using for visual servoing (eye-in-hand and/or eye-to-hand), the robot end-effector or the surrounding world will move in the recorded images. A successful application depends on robust and accurate trackers. We must also be able to initialize several trackers in one image. These trackers might be of different types. We need the ability to track points, lines and conics in order to handle different shaped targets.

Our work has focused on registration-based tracking. In registration-based tracking, also called sum of squared intensity difference (SSD) tracking, we specify a small image template that surrounds the desired object in the first image frame. We want to be able to find the location of this template in future frames. We warp incoming image frames to align them with the original template using a warp function with a corresponding warp parameter. With each new image frame the goal is to update the warp parameter. The Lucas-Kanade algorithm [11] serves as a foundation for many registration-based tracking algorithms [11, 12]. By minimizing the SSD between the current image warped with the latest warp parameter and the initial template, we are able to find the new update to the warp parameter.

In earlier work in our lab Dick *et al*. [13] tried a different approach to registration-based tracking by introducing machine learning into their system. They use approximate Nearest Neighbour Search with pre-computed partially aligned template images to find the corresponding warp parameter update. Later we have tried to replace the Nearest Neighbour Search by a different machine learning approach to see if the behavior was similar. In this case we learn the function between the warp updates and partially aligned image templates using Regularized Ridge Regression. For registration-based tracking to be useful to visual servoing, the algorithms must be able to handle large image motion between subsequent frames. A common problem with the Lucas-Kanade like approaches is that they only converge for small image motion. By introducing machine learning into the tracking procedure we improve on convergence for large image motion.

IV. SYTEMS AND RESULTS

In this paper we present ongoing work and describe different relevant experiments that we have conducted so far for HRI interfaces and tracking.

Point-to-Line Visual Servoing

Going back to the initial surgery example we have implemented an experiment where a WAM robot arm does point-to-line visual servoing. We track the end-effector as

well as two points on the line in the workspace using the Camshift tracker [14] in OpenCV (Figure 7). We capture images from two cameras and combine the visual servoing point-to-line error function, Epl, from both of them. Figure 8 displays the convergence of the visual servoing Epl error in the two images. The noise in the error comes from the tracker. The error in image I_1 is lower than the error in image I_2 because camera 1 is better positioned with respect to the scene than camera 2. Despite this we can see that the final position of the end-effector is indeed on the specified line (Figure 9). For a video of the visual servoing see [16].

Figure 7. The Camshift tracker is used to track the robot end-effector as well as two points on the line in the workspace.

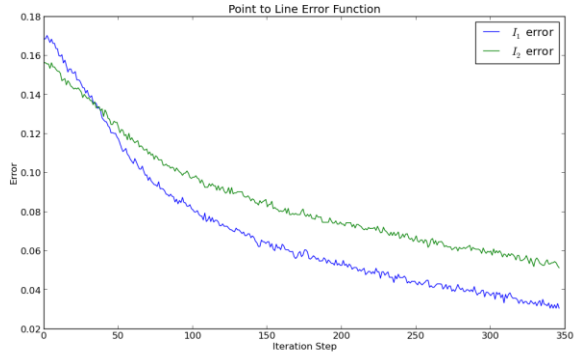

Figure 8. The convergence of the visual servoing error in the two image frames.

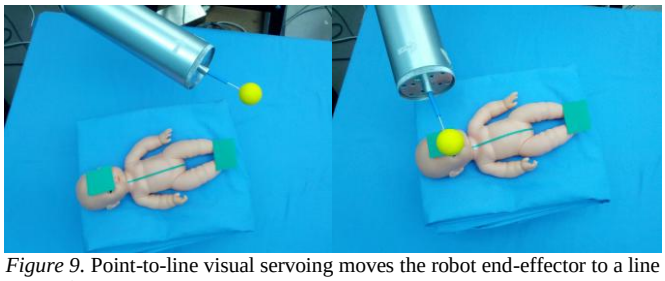

Figure 9. Point-to-line visual servoing moves the robot end-effector to a line defined by two green markers that are tracked as points on the line.

3D Pointing

In section II we described an interaction mode that lets the user point to objects he/or she wants the robot arm to grasp and manipulate [10]. We built on the SEPO (Select-by-Pointing) system from our lab presented by Quintero *et al.* in [15]. Here the location of the user's pointing gesture is detected by a Microsoft Kinect sensor. They found that the SEPO interface had users select objects with average position accuracy of 9.6 ± 1.6 cm. That means objects such as a cereal box on the table or a vacuum cleaner on the floor can be successfully chosen by a user. We use the 3D location from the SEPO interface to move the WAM robot arm using inverse kinematics. The objects are grasped from above using information from a point cloud to align the gripper. The participants interact with the robot arm to remove objects from a table and sort them into two containers [10] (Figure 10). The robot arm provides feedback by pointing at the closest object to the chosen location so the user can confirm or decline the selection. The final system was tested both with and without feedback. We conducted both human-human and human-robot experiments. In the human-human interaction one participant pointed at objects while the other had to interpret the pointing gestures and pick up objects. Eight people took part in the study. The participants stated that interacting with the robot was no more difficult than interacting with another human. Four objects were placed in two configurations with different level of difficulty. Without feedback the average success rate (success means the right object was picked and placed in the correct bin) for human-human experiments was 75% for the more challenging object configuration and 95% for the simpler one. For the human-robot experiments the average success rate was 88% for both configurations. When feedback was incorporated in the experiments both interactions had a 100% success rate.

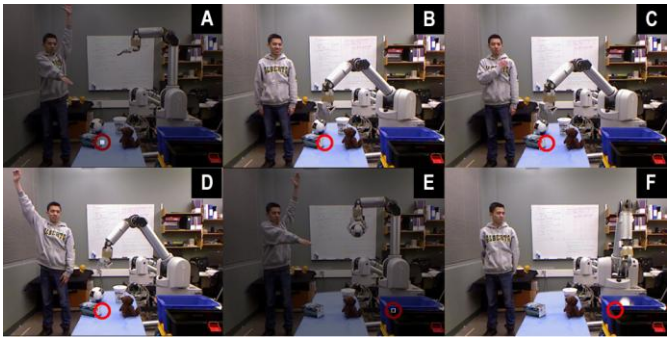

Figure 10. A user interacts through pointing and gestures with a robot arm that picks objects from a table and drops them into boxes. The robot points to the object closest to the chose location to receive confirmation from the user.

Interactive Teleoperation Interface for Semi-Autonomous Control of Robot Arms

In this system we propose an intuitive gesture interface that allows the human operator to switch between teleoperation mode and visual servoing (Figure 11). We design two alternate motion modes: (1) a direct linkage to the arm motion from the tracked human skeleton and (2) an image-based visual servoing routine. A successful application of the interface is presented for a WAM arm equipped with an eye-in-hand camera. Coarse motions are executed by human teleoperation and fine motions by image-based visual servoing.

By using an intuitive gesture interface, the user tracked by the Kinect is able to teleoperate the robot arm, by using a regular PC monitor that receives the image streams from cameras 1 and 2. The HRI combines the strengths of both teleoperation and visual servoing. For large manipulation

teleoperation is quicker than visual servoing, since the eye-in-hand camera has a limited field of view and it would be tedious and unintuitive for the user to define several segments of visual servoing. For precise manipulation on the other hand, the direct mapping of tracked human arm motions to robot motions suffers from noise in the tracking and it is difficult for the human to deal with the dynamics of the robot. We find that teleoperated motions, while fast, are jittery and not very precise. Here visual servoing relieves the human from dealing with the dynamics of the robot, and allows for precise motions.

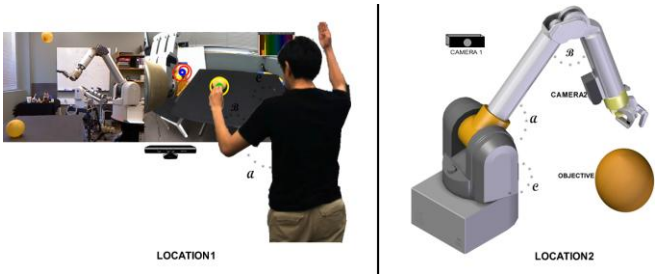

Figure 11. The user in location 1 needs a kinect and a regular PC. The user can only control the robot arm in location 2 by gesturing without touching any external device like e.g. a keyboard or a joystick.

Tracking

As mentioned earlier a problem for many Lucas-Kanade like tracking approaches is that they only converge for small image motion. Dick *et al.*'s [13] inclusion of machine learning to tracking improved the convergence for large image motion. They implemented and tested the Inverse Compositional (I.C.) [12] and Efficient Second Order Minimization (E.S.M) [11] algorithms as well as their own Nearest Neighbour tracker (N.N. + I.C.) on increasing static image motion. We repeat the same experiment, but include the Regularized Ridge Regression (R.R. + I.C.) algorithm as well. We add random Gaussian noise to the original object template image to create the updated template. The experiment is run with standard deviation from 1 to 20. Each experiment is run 5000 times and they record how often the different trackers fail. The results of our experiment are shown in Figure 12. We can see that the Nearest Neighbour and Ridge Regression algorithms handle large image motion better than the I.C. and E.S.M. algorithms.

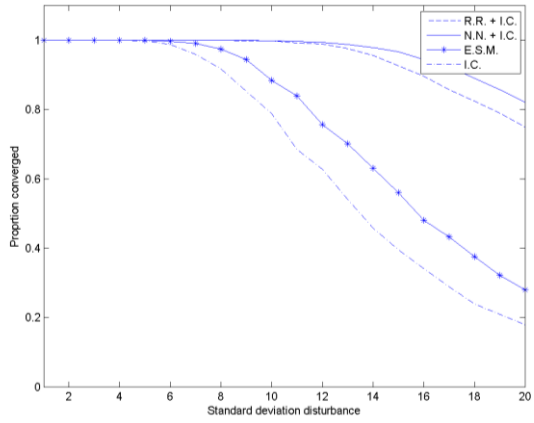

Figure 12. Testing of tracking algorithms on increasing static image motion. The results show the proportion of converged experiments for different standard deviations of Gaussian noise added to the original image.

CONCLUSION

We think that visual HRI through intuitive task specification is a first step on the way to facilitate effortless control of and interaction with a robot arm. Our suggested interfaces allow us to start working in more detail with visual task specification in a practical setting. In order to create better user control we need to gain more understanding of visual constraints and visual task specification. HRI, tracking and multi-view geometry as well as the theory behind visual servoing all need to be tackled to create a successful visual servoing system that can be used in real world applications. We have focused on HRI and tracking as these have received less focus in traditional visual servoing publications.

REFERENCES

[1] C. C. Kemp, A. Erdsinger, and E. Torres-Jara. Challenges for Robot Manipulation in Human Environments [Grand Challenges of Robotics]. IEEE Robotics and Automation Magazine, vol. 14, no. 1, pp. 20-29, March 2007.

[2] A. Edsinger, and C. C. Kemp. Human-Robot Interaction for Cooperative Manipulation: Handing Objects to One Another. Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication, 2007.

[3] K. M. Tsui, D. Kim, A. Behal, D. Kontak, and H. A. Yanco. "I Want That": Human-in-the-Loop Control of a Wheelchair-Mounted Robotic Arm. Journal of Applied Bionics and Biomechanics, vol. 8, no. 1, pp. 127–147, April 2010.

[4] J. P. Hespana, Z. Dodds, G.D. Hager, and A. S. Morse. What Tasks can be Performed with an Uncalibrated Stereo Vision System? International Journal of Computer Vision 35(1), 65-85 1999.

[5] Z. Dodds, G.D. Hager, A. S. Morse, and J. P. Hespana. Task Specification and Monitoring for Uncalibrated Hand/Eye Coordination. Proceedings of the IEEE International Conference on Robotics & Automation, 1999.

[6] Z. B. Dodds. Task Specification Languages for Uncalibrated Visual Servoing. PhD Thesis, 2000.

[7] S. Hutchinson, G. D. Hager, and P. I. Corke. A Tutorial on Visual Servo Control. IEEE Transactions on Robotics and Automation, vol. 12, no. 5, pp. 651-670, 19.

[8] Amir Massoud Farahmand, Azad Shademan, Martin Jägersand, Csaba Szepesvári: Model-based and model-free reinforcement learning for visual servoing. IEEE ICRA 2009: 2917-2924

[9] Kinova Robotics. www.kinovarobotics.com

[10] C. P Quintero, R. T. Fomena, M, Gridseth, and M. Jagersand. Visual Pointing Gestures for Bi-directional Human Robot Interaction in a Pick-and-Place Task. Submitted to IROS 2013.

[11] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. International Journal of Computer Vision, Vol. 56, No. 3, March, 2004, pp. 221 - 255.

[12] S. Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. IEEE/RSJ Int. Conference on Intelligent Robots and Systems, vol. 1, sept.-2 oct. 2004, pp. 943 – 948.

[13] T. Dick, C. Perez, A. Shademan, and M. Jagersand. A supervised learning approach to registration-based visual tracking. Robotics: Science and Systems, 2013.

[14] G.R. Bradski. Computer video face tracking for use in a perceptual user interface. Intel Technology Journal, Q2 1998.

[15] C. P. Quintero, R. T. Fomena, A. Shademan, N. Wolleb, T. Dick, and M. Jagersand. SEPO: Selecting by Pointing as an Intuitive Human-Robot Command Interface. IEEE International Conference on Robotics and Automation (ICRA), 2013.

[16] http://webdocs.cs.ualberta.ca/~vis/HRI/PointToLineVisualServoing.avi