JediBot – Experiments in Human-Robot Sword-Fighting

Torsten Kröger, Ken Oslund, Tim Jenkins, Dan Torczynski,
Nicholas Hippenmeyer, Radu Bogdan Rusu, and Oussama Khatib

Abstract. Real-world sword-fighting between human opponents requires extreme agility, fast reaction time, and dynamic perception capabilities. In this paper, we present experimental results achieved with a 3D vision system and a highly reactive control architecture which allows a robot to sword fight against human opponents. An online trajectory generator is used as an intermediate layer between low-level trajectory-following controllers and high-level visual perception. This architecture allows robots to react nearly instantaneously to the unpredictable human motions perceived by the vision system as well as to sudden sword contacts detected by force and torque sensors. Results show how smooth and highly dynamic motions are generated on-the-fly while using the vision and force/torque sensor signals in the feedback loops of the robot motion controller.

1 Introduction

Born as a class project [1, 2], the idea of the "JediBot" is a robot that performs sword fighting against a human opponent. The basic requirements for implementing a sword-fighting robot are *(i)* a reliable visual perception system that detects motions of the opponent and its sword, *(ii)* a reactive motion generation and control system for the robot to be able to immediately react to the opponent's motion, and *(iii)* compliant and reactive motion control capabilities for physical human-robot interaction. Furthermore, appropriate attack and defense strategies are required that make use of the three mentioned aspects.

Torsten Kröger · Ken Oslund · Tim Jenkins · Dan Torczynski ·
Nicholas Hippenmeyer · Oussama Khatib
Artificial Intelligence Laboratory at Stanford University, Stanford, CA 94305-9010, USA
e-mail: `tkr@stanford.edu`

Radu Bogdan Rusu
Open Perception, Inc. , 68 Willow Road, Menlo Park, CA 94025, USA

J.P. Desai et al. (Eds.): *Experimental Robotics*, STAR 88, pp. 155–166.
DOI: 10.1007/978-3-319-00065-7_12 © Springer International Publishing Switzerland 2013

This paper describes *(a)* the hardware and *(b)* software system of "JediBot", *(c)* experimental results of human-robot interaction during sword-fighting, and *(d)* how readers can replicate the system and run experiments with their own system set-up. Figure 1 illustrates the suggested setup: A KUKA/DLR Lightweight Robot [3] is equipped with a foamed wooden sword, and a 3D camera (MS Kinect, [4]) is used to perceive the human opponent and its sword. The two most challenging parts that make use of recent research outcomes are *3D Visual Perception* and *Online Trajectory Generation*.

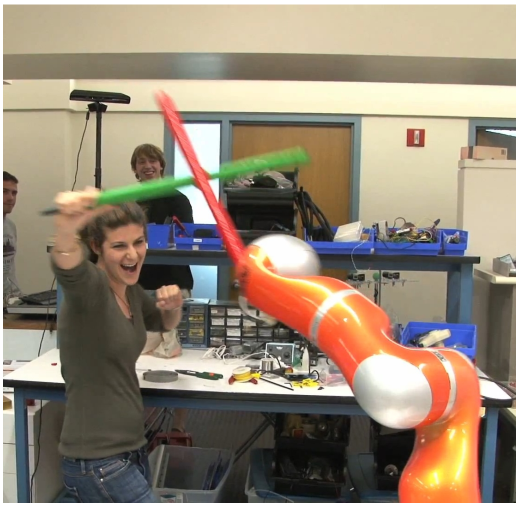
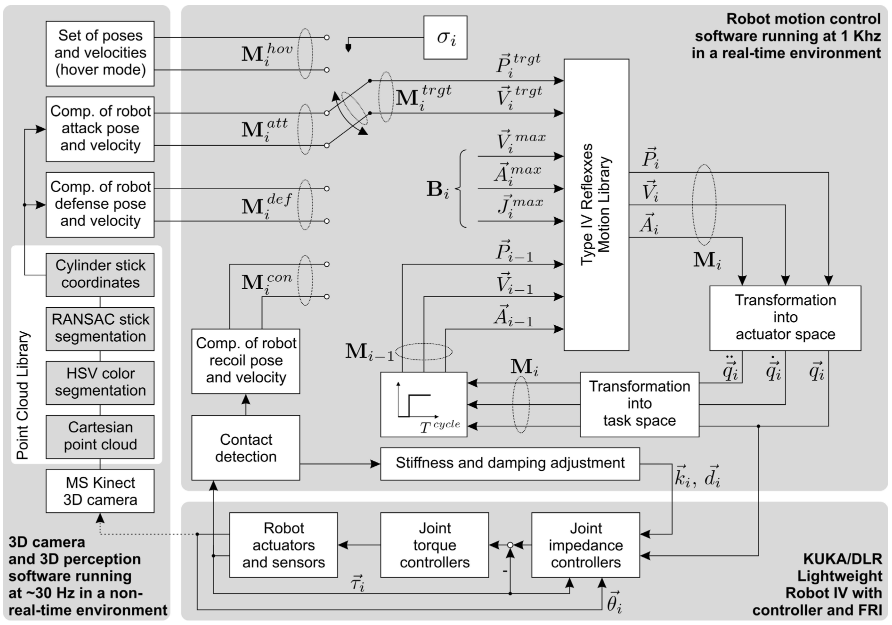


Fig. 1 Reactive motion generation and control during physical human-robot interaction with "JediBot"

3D Visual Perception. Based on underlying technology from PrimeSense [5], 3D cameras such as Microsoft Kinect [4] have started to simplify 3D visual perception procedures in robotics. Our approach to detect the opponents posture and sword in unstructured environments operates on 3D data. One of the most popular descriptors for 3D data is the Spin-image presented by Johnson *et al.* [6], which is a 2D representation of the surface surrounding a 3D point and is computed for every point in the scene. Two of the key technologies for online segmentation operations are provided by FLANN (Fast Library for Approximate Nearest Neighbors, [7]) and RANSAC (Random Sample Consensus, [8]). All required 3D perception methods and algorithms for this task are provided by PCL (Point Cloud Library, [9,10]). How these methods are applied is, for instance, shown in [11].

Online Trajectory Generation. Reactive online motion generation is required to immediately react to motions of the human opponent and to contacts between the robot's and the opponent's sword. Broquère *et al.* [12] published a method that uses an online trajectory generator for an arbitrary number of independently acting degrees of freedom. The approach is very similar to the one of Liu [13] and is based on the classic seven-segment acceleration profile. The work of Haschke *et al.* [14] presents an online trajectory planner in the very same sense as [15] does. The proposed algorithms generate jerk-limited trajectories from arbitrary states of motion. All required motion generation concepts for this task are provided by the Reflexxes Motion Libraries [16, 17].



Robot motion control software running at 1 Khz in a real-time environment
Set of poses and velocities (hover mode)
Comp. of robot attack pose and velocity
Comp. of robot defense pose and velocity
Cylinder stick coordinates
RANSAC stick segmentation
HSV color segmentation
Cartesian point cloud
MS Kinect 3D camera
Point Cloud Library
Comp. of robot recoil pose and velocity
Contact detection
Stiffness and damping adjustment
Type IV Reflexxes Motion Library
Transformation into actuator space
Transformation into task space
T^{cycle}
$\mathbf{M}_i^{hov}$
$\mathbf{M}_i^{att}$
$\mathbf{M}_i^{def}$
$\mathbf{M}_i^{con}$
$\mathbf{M}_i^{trgt}$
σ_i
$\vec{P}_i^{trgt}$
$\vec{V}_i^{trgt}$
$\mathbf{B}_i$
$\vec{V}_i^{max}$
$\vec{A}_i^{max}$
$\vec{J}_i^{max}$
$\vec{P}_{i-1}$
$\vec{V}_{i-1}$
$\vec{A}_{i-1}$
$\vec{P}_i$
$\vec{V}_i$
$\vec{A}_i$
$\mathbf{M}_i$
$\mathbf{M}_{i-1}$
$\ddot{\vec{q}}_i$
$\dot{\vec{q}}_i$
$\vec{q}_i$
$\vec{k}_i,\ \vec{d}_i$
Robot actuators and sensors
Joint torque controllers
Joint impedance controllers
$\vec{\tau}_i$
$\vec{\theta}_i$
3D camera and 3D perception software running at ~30 Hz in a non-real-time environment
KUKA/DLR Lightweight Robot IV with controller and FRI

Fig. 2 Overall control scheme of “JediBot”. It can be subdivided into three units: the visual perception hardware and software (left), the real-time motion generation and control unit (top), and the KUKA/DLR Lightweight Robot IV (bottom).

2 Technical Approach

Figure 2 gives an overview of the system’s hardware and software components, which consist of three main blocks:

- hardware and software for visual detection of the human opponent its sword,
- a real-time motion generation and control unit, and
- the robot arm with a sword mounted to its hand (cf. Fig. 1).

The following three subsections describe these components, and Sec. 3 discusses the interplay between them.

2.1 Human Opponent and Sword Detection

The hardware component of the JediBot vision system is a Microsoft Kinect sensor which provides both a standard RGB color image and a stereo camera derived depth image at up to 30 fps and 640 × 480 pixels in resolution. The software component is based on the Point Cloud Library (PCL) [9]. At startup the system goes through an automatic calibration routine, in which the sword is placed in two perpendicular positions which are known in the robot’s frame and then detected in the Kinect

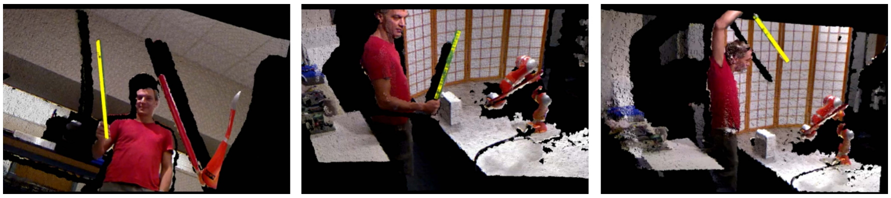


Fig. 3 Screen shots showing the precision and robustness of the RANSAC segmentation [8] of the human's sword (yellow) in different configurations

reference frame. This is done using the same algorithm that detects the opponent's sword (described bellow).

Multiple measurements are taken at each position and averaged to reduce noise. Using this data, an affine transformation between the robot and Kinect reference frames is calculated using functions built into PCL. Once the calibration routine is complete, the entire point cloud is transformed into the robot frame as soon as it is captured, so that the results of all further processing are automatically in the robot reference frame.

To detect the sword, the point cloud is first filtered based on depth to eliminate background points. Based on the calibration data, an HSV (hue, saturation, value) color segmentation is performed to select points, which are approximately the same color as the sword. Then a RANSAC algorithm is used to fit a line in three dimensional space to determine end points of the most stick-like object. Finally, the length of the detected sword is compared to the expected length, and the detection discarded if it is too long or too short to further reduce false positives. Figure 3 illustrates three sample results. The sword speed, calculated from its movement between two or more successive frames, is also used because it proved to be the most reliable method of detecting when an opponent begins their swing. This processing pipeline is split across multiple threads to improve performance.

2.2 Online Trajectory Generation

The Online Trajectory Generation algorithms of [16] are contained in the Reflexxes Motion Libraries [16,17]. They lets us compute synchronized motions for N degrees of freedom from any state of motion $\mathbf{M}_{i-1}$ at instant T_{i-1} represented by position, velocity, and acceleration vectors with N elements each,

$$\mathbf{M}_{i-1} = (\vec{P}_{i-1}, \vec{V}_{i-1}, \vec{A}_{i-1}) \ . \tag{1}$$

The algorithm will transfer the system from this state of motion into the desired target state

$$\mathbf{M}_i^{trgt} = (\vec{P}_i^{trgt}, \vec{V}_i^{trgt}, \vec{0}) \tag{2}$$

under consideration of the current maximum values for velocity, acceleration, and jerk,

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

The output values of the algorithm define the desired state of motion $\mathbf{M}_i$ at $T_i = T_{i-1} + T^{cycle}$, where T^{cycle} is the value of the control cycle time. After the transformation into actuator space, the values of joint position $\vec{q}_i$ and its derivatives are used as command variables for the joint controller.

The values of $\vec{A}_i^{max}$ are permanently updated using the forward dynamic model, for which a constant maximum torque vector $\vec{\tau}^{max}$ is assumed:

$$\vec{A}_i^{max} = \vec{f}\left(\vec{q}_i,\, \dot{\vec{q}}_i,\, \vec{\tau}^{max}\right) \; . \tag{4}$$

As we will learn in Sec. 3, this will allow for the use of discontinuous input signals $\mathbf{M}_i^{trgt}$ of a switched system while permanently guaranteeing steady, jerk-limited, synchronized robot motions for all N degrees of freedom:

$$\begin{aligned}
&\forall\, n \in \{1, \ldots, N\} : \\
&\qquad |_nP_i - {}_nP_{i-1}| \le {}_nV_{i-1}\, T^{cycle} + \tfrac{1}{2}\, {}_nA_{i-1} \left(T^{cycle}\right)^2 \pm \tfrac{1}{6}\, {}_nJ_i^{max} \left(T^{cycle}\right)^3 \;\wedge \\
&\qquad |_nV_i - {}_nV_{i-1}| \le {}_nA_{i-1}\, T^{cycle} \pm \tfrac{1}{2}\, {}_nJ_i^{max} \left(T^{cycle}\right)^2 \;\wedge \\
&\qquad |_nA_i - {}_nA_{i-1}| \le {}_nJ_i^{max}\, T^{cycle} \;\wedge\; |_nV_i| \le {}_nV_i^{max} \;\wedge\; |_nA_i| \le {}_nA_i^{max} \; .
\end{aligned} \tag{5}$$

Because of this property, the trajectory generation can guarantee that it will send control commands to the robot at perfectly regular intervals (a requirement for the robot to opperate properly) even if it recieves target states of motion at highly irregular intervals. Thus, the image processing hardware and software does *not* necessarily have to be real-time capable, and its interface with the real-time trajectory generator can be very simple, consisting only of sending the desired states of motion whenever they become available. (cf. Fig. 2). A sample trajectory for three degrees of freedom is shown in Fig. 4.

2.3 Robot Hardware

A *KUKA Light-Weight Robot IV* [3, 18] was controlled through the *Fast Research Interface* [19, 20] with a control cycle time of $T^{cycle} = 1\,ms$. The simplicity of our setup is based on the control scheme of Fig. 2. Its three components as well as their interfaces were implemented with a focus on overall computational efficiency. The robot end-effector only consists of a foamed wooden sword (cf. Fig. 1).

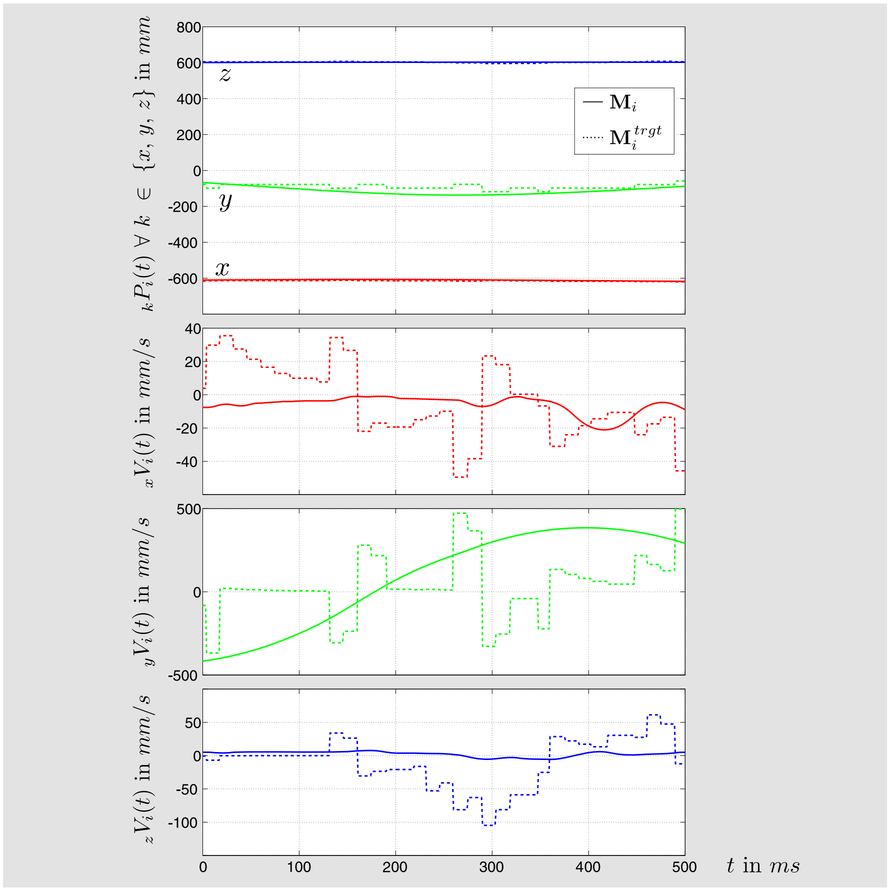


Fig. 4 Position and velocity progressions of $\mathbf{M}_i$ and $\mathbf{M}_i^{trgt}$ during a motion in defense mode. While $\mathbf{M}_i^{trgt}$ is only updated sporadically, the computed trajectory is provided at a rate of 1 KHz, such that the robot reacts instantaneously to sensor signals.

3 Sword Fighting Strategies

In order to implement appropriate sword fighting strategies using the three components described in the previous section, four control modes have been implemented: defense, attack, contact, and hover.

Defense *(def)* The robot defends itself by attempting to block attack swings of the human. The pose of the human's sword, obtained by the vision system, is used to determine the desired position and orientation of the robot's sword. When the human's sword is within a specified defense range and the velocity of that sword in the direction of the robot exceeds a certain threshold value, the robot moves to the blocking position.
In the blocking position, the midpoint of the robot's sword is placed on the defense line, which is a line between the midpoint of the human's sword and a predefined defense point which represents the center of the robot (cf. Fig. 5). The robot's sword is oriented such that it is orthogonal to both the defense line and the human's sword. Mathematically speaking, the robot's sword is oriented parallel to the direction of the cross product between the directions of the human's sword and the defense line. As the human's sword moves in towards the robot's sword, the robot will push its sword proportionally outward along the defense line until they meet.

Attack *(att)* When the opponent's sword is beyond the defense range, the robot executes periodic attacking motions, swinging toward the opponent from a randomly selected direction.

Contact *(con)* Detection of collisions between the human and robot swords is done using the torque sensors in each of the robot's joints. If the human blocks an attack motion, the robot detects this though the increased torque in its joints, and it immediately recoils, returning to the defensive position. This mode is activated in the same control cycle contact is detected.

Hover *(hov)* If no human sword is detected, the robot enters idle mode, where the bot sword just hovers around a specified position and orientation until a human oponent's sword is detected.

In the scheme of Fig. 2, the discrete value of

$$\sigma_i \in \{def, att, con, hov\} \tag{6}$$

selects the signal source of $\mathbf{M}_i^{trgt}$ that is used to feed the online trajectory generator:

$$\mathbf{M}_i^{trgt} = \begin{cases} \mathbf{M}_i^{def} & \text{if} \quad \sigma_i = def \\ \mathbf{M}_i^{att} & \text{if} \quad \sigma_i = att \\ \mathbf{M}_i^{con} & \text{if} \quad \sigma_i = con \\ \mathbf{M}_i^{hov} & \text{if} \quad \sigma_i = hov \end{cases} . \tag{7}$$

σ_i can change spontaneously based on the fight strategy and on sensor signals. For instance, if contact is detected, $\sigma_i = con$ will be applied in the same control cycle. How the value of σ_i is selected and how the components of Fig. 2 interact, will be described in Sec. 4.

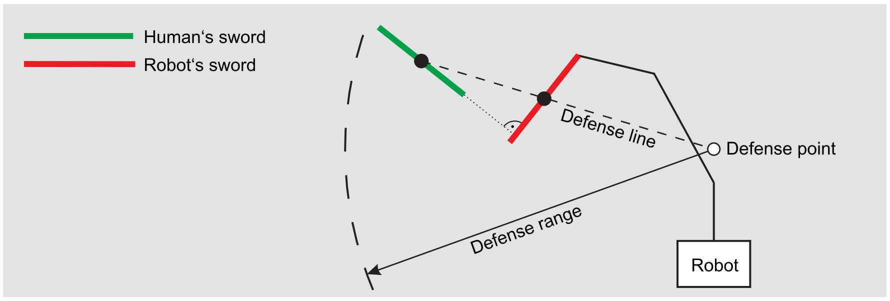


Fig. 5 During *defense mode*, the midpoint of the robot's sword (red) lies on the defense line, which is a line between the midpoint of the human's sword (green) and a predefined defense point

In order to provide a compliant behavior of the end-effector while both swords are in contact, the stiffness vector $\vec{k}_i$ is adjusted. Assuming a contact was detected at a time instant T_c, the following function is used:

$$\vec{k}_i = \begin{cases} \vec{k}^{min} + \left(\frac{T_i - T_c}{T^{recover}}\right)\left(\vec{k}^{max} - \vec{k}^{min}\right) & \text{if} \quad T_i \le T_c + T^{recover} \\ \vec{k}^{max} & \text{if} \quad T_i > T_c + T^{recover} \end{cases} . \tag{8}$$

$T^{recover}$ is the time until the maximum stiffness $\vec{k}^{max}$ is achieved again. As long as the contact detection indicates contact, T_c is set to T_i. The damping vector $\vec{d}_i$ remains constant.

4 Robot Sword Fighting Experiments and Results

The key aspect for achieving a good human-robot interaction behavior during the experiments is an appropriate motion strategy. The simplicity and very high reactivity of the proposed control scheme allow the system to freely realize different strategies. Based on the usage of multiple sources for desired states of motion $\mathbf{M}_i^{trgt}$, the system switches between them depending on the current state, strategy, and situation. This allows the robot make *instantaneous* use of sensor signals.

For example, if contact is detected at an instant T_i ($\sigma_i = con$), $\mathbf{M}_i^{con}$ and an adjusted value of $\vec{k}_i$ will be applied in the same control cycle already. After the recoiling motion is completed, the robot can either attack again or switch to defend mode.

The overall strategy defines when and how to set the selection variable σ_i. The most appropriate behavior has been achieved with *defense mode* as default strategy. If the human's sword is beyond the defense radius, then the *attack mode* is initiated. The *contact* mode is only active while the swords are in contact after an attack

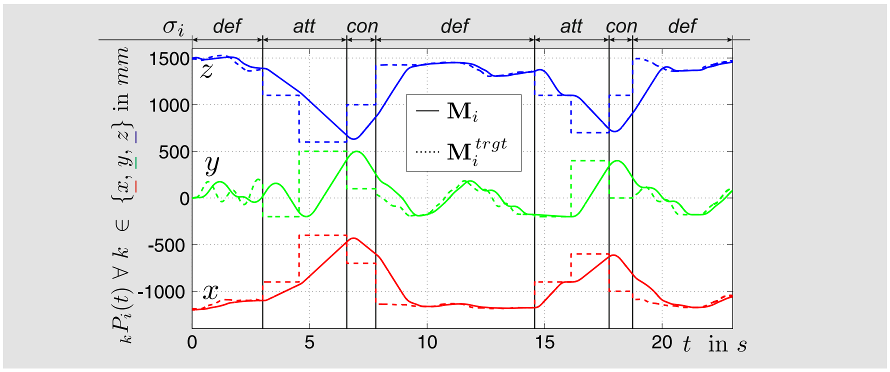


Fig. 6 Position progressions of $\mathbf{M}_i$ and $\mathbf{M}_i^{trgt}$ while switching between different modes over a period of 23 s. Although arbitrary switching procedures are triggered by σ_i (i.e., $\mathbf{M}_i^{trgt} = \mathbf{M}_i^{\sigma}$ is not steady), a jerk-limited and executable motion $\mathbf{M}_i$ is generated online in each control cycle (cf. Fig. 4). The switching sequence of σ_i is *def–att–con–def–att–con–def.*

motion (cf. eqn. 8). Figure 6 shows the desired and the achieved steady progressions of the robot's end-effector position that is achieved during arbitrary switchings of σ_i.

The vision processing chain on the left of Fig. 2 typically runs at rates of 10 – 15 fps, but since some parts of the processing pipeline take place in parallel, the latency is up to 300 ms. The need to calculate sword speed between two successive frames and other delays in the sword command and control further increase the latency. The total system latency between the opponent beginning a swing with their sword and the beginning of robot motion in response is typically around 500 ms. While it is possible for the opponent to move faster than this, it is commonly fast enough for the robot to respond.

As indicated in Fig. 3, the quality of recognition is very high, with very few false positives or dropped frames. If the sword was pointed straight at the Kinect, such that only its tip is visible, it could not be detected, but it typically only needs to be angled about 15 degrees away from the camera to allow detection. By properly positioning the camera it is possible to ensure that the sword would rarely pass through this narrow cone, and when it does, it would only be for a very brief amount of time.

Based on the visual sword detection, the desired defense pose and velocity ($\mathbf{M}_i^{def}$) are continuously updated. As indicated in Fig. 4, the image processing loop runs at a different rate than the robot motion controller (10 – 15 Hz and 1 KHz); as soon as a new value of $\mathbf{M}_i^{def}$ is provided, it will be immediately applied in the next robot control cycle.

5 Conclusions

Target State Switching

Real-world experimental results of a switched-system using a selection variable (σ_i) to select between different desired target states of motion were shown (cf. Fig. 6). Despite its simplicity, this approach promises to simplify and improve sensor-based robot motion control, because robots can react instantaneously and in different ways to unforeseen sensor signals and events.

Human-Robot Interaction

Using the approach of target state switching, jerk-limited executable motions are generated online, such that robots can *permanently* respond to human motions deploying sensor signals in the feedback loops.

Reliability and Robustness

The recognition of the opponent's sword based on camera data of a MS Kinect 3D camera using segmentation procedures of the Point Cloud Library (PCL) works very reliably. If there is a significant translational and/or rotational error or if there is a dropped frame, a jerk-limited and executable motion will always be generated by the Reflexxes Motion Libraries, such that the system is stable despite erroneous sensor signals. The overall control scheme reacts deterministically and runs very robustly because only a joint position or impedance controller is required.

Replicable Implementations and Experiments

The two main software components, the Point Cloud Library (PCL), the Reflexxes Motion Library, and the interface software for the KUKA/DLR Lightweight Robot are freely available [10, 17, 20], such that the proposed control scheme can be easily duplicated with very reasonable efforts. If other robots are used, only a trajectory tracking controller is required. For visual perception, only a MS Kinect 3D camera is required [4].

Implementation Time: Three Weeks

The original "JediBot" was entirely created by students of the class CS225A at Stanford University [1] within only three weeks. Despite many iterations to improve the system and to exhibit it at the 2011 IEEE International Conferences on Intelligent Robots and Systems, the control scheme of Fig. 2 remained with relatively few modifications. Using [10, 17], the scheme is very simple and straight-forward to implement even for students with limited experience in robotics. The websites of [10, 17] provide tutorials and examples for the presented matter.

Acknowledgements. All authors would like to thank *QNX Software Systems* and *Wind River Systems* for providing free software licenses. The first author would like to thank the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) that is currently co-funding him.

References

1. Stanford Artificial Intelligence Laboratory, Stanford Universiy, 353 Serra Mall, Stanford, CA 94305-9010, USA. Course Description: CS225A Experimental Robotics. Internet (2012), `http://cs.stanford.edu/groups/manips/teaching/cs225a` (accessed: June 2, 2012)
2. Fyffe, S.: Students Create 'JediBot'. YouTube Video, Internet (2011), `http://www.youtube.com/watch?v=VuSCErmoYpY` (accessed: June 2, 2012)
3. Bischoff, R., Kurth, J., Schreiber, G., Köppe, R., Albu-Schäffer, A., Beyer, A., Eiberger, O., Haddadin, S., Stemmer, A., Grunwald, G., Hirzinger, G.: The KUKA-DLR lightweight robot arm — A new reference platform for robotics research and manufacturing. In: Proc. of the Joint Conference of ISR 2010 (41st Internationel Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), Munich, Germany. VDE Verlag (June 2010)
4. Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052-7329, USA. Microsoft kinect homepage. Internet (2012), `http://xbox.com/Kinect` (accessed: June 2, 2012)
5. PrimeSense, 28 Habarzel St. Tel-Aviv, 69710, Israel. Homepage, Internet (2012), `http://www.primesense.com` (accessed: June 2, 2012)
6. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Trans. on Pattern Analysis and Machine Intelligence 21(5), 433–449 (1999)
7. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: Proc. of the International Conference on Computer Vision Theory and Application, Lisbon, Portugal, pp. 331–340 (June 2009)
8. Fischler, A.M., Bolles, C.R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)
9. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: Proc. of the IEEE International Conference on Robotics and Automation, Shanghai, China (May 2011)
10. Point Cloud Library (PCL). Homepage, Internet (2012), `http://www.pointclouds.org` (accessed: June 2, 2012)
11. Steder, B., Rusu, R.B., Konolige, K., Burgard, W.: Point feature extraction on 3d range scans taking into account object boundaries. In: Proc. of the IEEE International Conference on Robotics and Automation, Shanghai, China, pp. 2601–2608 (May 2011)
12. Broquère, X., Sidobre, D., Herrera-Aguilar, I.: Soft motion trajectory planner for service manipulator robot. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, pp. 2808–2813 (September 2008)
13. Liu, S.: An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators. In: Proc. of the Seventh International Workshop on Advanced Motion Control, Maribor, Slovenia, pp. 365–370 (July 2002)

14. Haschke, R., Weitnauer, E., Ritter, H.: On-line planning of time-optimal, jerk-limited trajectories. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, pp. 3248–3253 (September 2008)
15. Kröger, T., Wahl, F.M.: On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. IEEE Trans. on Robotics 26(1), 94–111 (2010)
16. Kröger, T.: Opening the door to new sensor-based robot applications — The Reflexxes Motion Libraries. In: Proc. of the IEEE International Conference on Robotics and Automation, Shanghai, China (May 2011)
17. Reflexxes GmbH, Sandknöll 7, D-24805 Hamdorf, Germany. Hompepage, Internet (2012), `http://www.reflexxes.com` (accessed: June 2, 2012)
18. KUKA Laboratories GmbH, Zugspitzstraße 140, D-86165 Augsburg, Germany. Homepage, Internet (2012), `http://www.kuka-labs.com/en` (accessed: June 2, 2012)
19. Schreiber, G., Stemmer, A., Bischoff, R.: The fast research interface for the KUKA lightweight robot. In: Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications — How to Modify and Enhance Commercial Controllers at the IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, pp. 15–21 (May 2010)
20. Kröger, T.: Documentation of the fast research interface library, version 1.0. Internet (November 2011), `http://cs.stanford.edu/people/tkr/fri/html` (accessed: June 2, 2012)