GRASPY – Object Manipulation with NAO

Judith Müller[1], Udo Frese[2], Thomas Röfer[2],
Rodolphe Gelin[3], and Alexandre Mazel[3]

[1] Transregional Collaborative Research Center for Spatial Cognition, SFB/TR8,
University of Bremen, Bremen, Germany
judith.mueller@uni-bremen.de
[2] Deutsches Forschungszentrum für Künstliche Intelligenz, Cyber Physical Systems,
Bremen, Germany
{udo.frese,thomas.roefer}@dfki.de
[3] Aldebaran Robotics, Paris, France
{rgelin,amazel}@aldebaran-robotics.com

Abstract. In this paper we introduce an online object manipulation system for the NAO robot that is able to detect and grasp an object out of a human hand and then give it back in real-time. Known objects are rendered from 3D models and detected stereo contour-based by using a new stereo vision head for NAO. In order to grasp objects, motion trajectories are generated by an A* planner while avoiding obstacles. In order to safely release objects back into a human hand, a combination of tactile and force sensors of the carrying arm is used to detect whether someone touched the grasped object. We performed quantitative experiments in order to evaluate the quality of the detector, the time to grasp an object, as well as the number of successful grasps. We demonstrated the whole system on the real robot.

Keywords: stereo vision object detection, online grasp motion planning.

1 Introduction

The aim of the GRASPY project is to make a small move towards the integration of humanoid robots in our everyday life. In a visionary scenario a small humanoid robot could be a personal assistant that not only is able to organize contacts and emails but also could support a person physically by getting objects for him. This would be especially interesting if a person is not able to get the object for himself. In the scope of this project we wanted to investigate what is necessary to allow the humanoid robot NAO [1] to have this functionality.

In this paper we present an online object manipulation system that is an extension of our previous work [2]. It not only combines a new stereo vision head for NAO with an object detector and an updated version of our grasping function, but also includes the releasing of the object. The objects to grasp are ones that can be completely clasped by the robot's fingers and palm. We focus

F. Röhrbein et al. (eds.), *Gearing Up and Accelerating Cross-Fertilization between Academic and Industrial Robotics Research in Europe*, Springer Tracts in Advanced Robotics 94,
DOI: 10.1007/978-3-319-02934-4_9,

on small objects or objects with a handle, for instance a light standard-sized coffee cup or a pencil.

The first step of the general procedure is to detect the object. Thereby edge detection is performed on the left and right stereo images by computing a contrast-normalized Sobel (cns) image instead of using color segmentation as in the work of Azad *et al.* [3]. Afterwards the contrast images are used to recognize the object by evaluating a range of possible object poses. The detected object is given to the grasp planner as a 6D pose.

The next step of the procedure is to decide whether an object can be grasped. By doing so, possible grasp hand poses as well as possible body positions are evaluated by using a pre-calculated workspace. Once a possible hand grasp pose is found, a motion path from the current hand position is planned using an A*-based algorithm. In contrast to the work of Cotugno and Mellmann [4], who use both arms to emulate a big two-finger gripper, but do not use NAO's real fingers, the grasping function proposed in this paper plans and executes actual single-handed grasps. At last, the resulting path is approximated by a Bezier curve and then executed by a trajectory based motion engine.

Because our system uses speech recognition, which is a part of NAO's standard software package, the robot is able to react to spoken commands. Therefore the robot is able to wait and hold a grasped object until someone asks him to release it. The releasing itself is the last part of our procedure. Thereby the robot uses a combination of tactile and force sensors of the carrying arm and detects whether someone touched the grasped object in order to start a safe object handover.

2 Related Work

A vast number of online manipulation systems can be found in the RoboCup @Home League [5] where robots – among other assignments – have to manage different grasp and detection tasks. One participating team is the German b-it-bots with their robot Jenny [6] from the Bonn-Rheine-Sieg University of Applied Sciences. Jenny is equipped with a 7 DOF arm and a three finger hand with at least one motor per finger. The robot categorizes objects with a so-called *Bag of Features* [7] approach that relies on the extraction of locally invariant features. In 2012, the robot Jenny demonstrated its ability to clean a room. In particular the robot was able to pick up bottles in order to insert them into a stash as well as it was able to wipe tables.

Another online manipulation system can be found in the work of Stückler *et al.* [8], where the robot Cosero from the University of Bonn uses a RGB-D camera to recognize and track objects on a table in real-time. Thereby the main horizontal support plane, i.e. the table, is distinguished from object candidates by the RANSAC [9] algorithm that is applied to a 3D point cloud similar to the work of Rusu *et al.* [10]. Feasible collision-free grasps are derived based on the work of Hsiao *et al.* [11] from the point cloud. The robot Cosero is equipped with two two-finger hands (2 DOF), two 7 DOF arms and an Intel I7 processor, which provides a much higher performance than NAO's Intel Atom processor. At the

RoboCup 2011 the robot demonstrated its ability to carry a table together with a human [12] as well as its ability to cook an omelet in a real pan.

In the work of Kuffner *et al.* [13], a full body motion planner plans a dynamically stable motion in the configuration space using rapidly-exploring random trees. It enables a robot to collision-free pick up a bottle that is placed under a table. Based on that approach Burget *et al.* [14] are using an external computer to plan full body grasp motions that enables NAO to open a drawer door. Since this solution requires a long calculation time before a motion is executed, it does not appear to be suitable for grasping objects out of a human hand.

In general most motion planners are operating either in Cartesian or in configuration space. While motion planners in configuration space as in the works of Kavraki *et al.* [15] and Harada *et al.* [16] are able to guarantee a solution given there is one, planners in Cartesian space as in the approach of Vahrenkamp *et al.* [17] are incomplete and difficult due to redundant kinematics. However, the integration of obstacle avoidance into a path planner operating in Cartesian space is simpler than in configuration space.

Furthermore, motion path planning in Cartesian space can be a very expensive process particularly when the grasping hand is attached to a humanoid robot, which can move in order to reach certain objects. Thus, the reachability needs to be checked by inverse kinematics for many points in order to select a suitable grasp and to validate the reachability along the path. This process can be speed up best by a pre-calculated table as the capability map of Zacharias *et al.* [18,19]. In our work we use the predefined workspace to solve redundant kinematics as well as the reachability along the motion path, which enables our motion path planner to quickly operate in Cartesian space.

Stereo vision based object detection as well as online grasping with NAO constitute as particular problems due to the limited processing power and the under-actuated hand design [20]. While more sophisticated robots have fingers that are controlled by at least one motor per finger, NAO's three flexible fingers per hand are controlled together by a single motor (1DOF). Additionally only if the hand is completely closed the fingers are really stiff. Because of that the hand can realistically only be in the states open or closed. Hence, experiments showed that solid objects such as coffee cups are only graspable if the grasp is form-closure [21]. Furthermore, it seems that performing force-closure single-handed grasps are not possible with NAO, since it is not able to move its fingers individually [22].

3 Stereo Head

The original head of NAO is equipped with two cameras: one in its forehead and one in its chin. This configuration results from the requirement of having a camera oriented upwards to detect people around NAO and another one oriented downwards to detect objects such as a soccer ball on the ground. Because the fields of view of the cameras in the original head design do not overlap each other, stereo vision is impossible with that approach. For the needs of the GRASPY

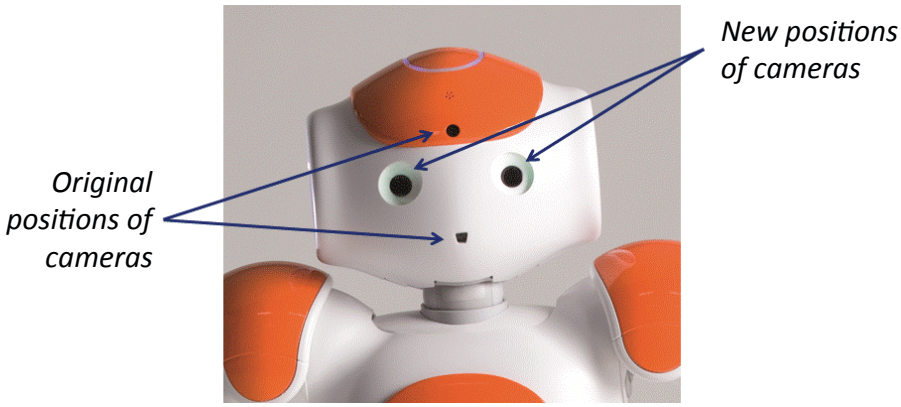


Fig. 1. New positions of the cameras for the stereo head

project, Aldebaran developed a new head with a more natural configuration: the two cameras are positioned in the eyes (Fig. 1). Thereby we selected a different camera model (Aptina MT9M114 [23]) with a higher resolution (1,3 MPixels), a wider field of view (72°) and a better sensitivity (2,24 Lux/(V.sec)) than the previous sensor used in NAO. This new positioning of the cameras in the eyes of NAO required us to suppress the colored LEDS in the eyes. We have to see how we can regain this important feature for man robot interaction in future versions of the stereo head.

The wide overlap (Fig. 2) of the two cameras offers the ability to perform stereo vision in front of the robot. The other major improvement required for stereo vision was the possibility to allow the synchronous acquisition of the two camera images. With the original head design, the application has to select between using the picture from the upper or from the lower camera. This hardware switch is necessary because there is only one video input on the embedded GEODE CPU. With two video inputs, switching between them would require more than 500 ms. As soon as the robot or the environment moves, this delay is generally too big to have a comparison of the two images usable for stereo vision computation.

In the new architecture of NAO's head, we have implemented an FPGA component that makes the acquisition of the two video streams and sends them via an I^2C bus to the ATOM CPU. The delay between two acquired images is less than 33 ms. We are currently working on a hardware synchronization of the two cameras to reduce this delay below 1 ms, but for the GRASPY project we had a delay of up to 33ms. This was reasonable considering the speed of the object to grasp.

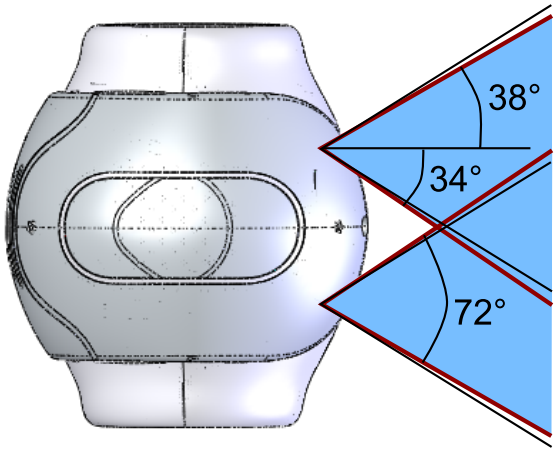


Fig. 2. Overlap of the fields of view with the stereo head

4 Stereo Contour-Based Object Detection

In computer vision there is the general insight that taking hard decisions early impairs robustness. Examples for that are pixel-wise color segmentation or Canny edge detection followed by line segment extraction followed by object detection. Instead, one should take a decision only after considering all relevant input data, in our case the whole stereo image, assessing which interpretation is overall most supported by the data. Compared to mono, stereo gives a better depth perception, and following the above paradigm we do a combined search in both images, not separately.

The first step of our detection is rasterization, i.e. rendering the object in a given hypothetical pose from the perspectives of the left and right cameras. The result is a 2D contour, i.e. a function $[0 \dots 1] \to \mathbb{R}^2$. The second step is contour evaluation, i.e. computing a response how much the contour is supported by the image. Its definition has already been described in detail in our previous work [2]. Based on this goal function, an optimizer searches through the space of possible poses, finding the cup pose with the largest response.

4.1 3D Object Search Process

Rasterization. The rasterization (Fig. 3) takes a triangle mesh as 3D object model, a camera calibration, and a hypothetical object pose as input and renders the contour of the object at the given pose as viewed from the camera. The first step is to determine which edges of the model form the contour. At the moment, we go through all edges and select those where one adjacent face is viewed from the front and one from the back. This does not consider global occlusion, an extension that could be implemented in the future. As a special rule, faces can

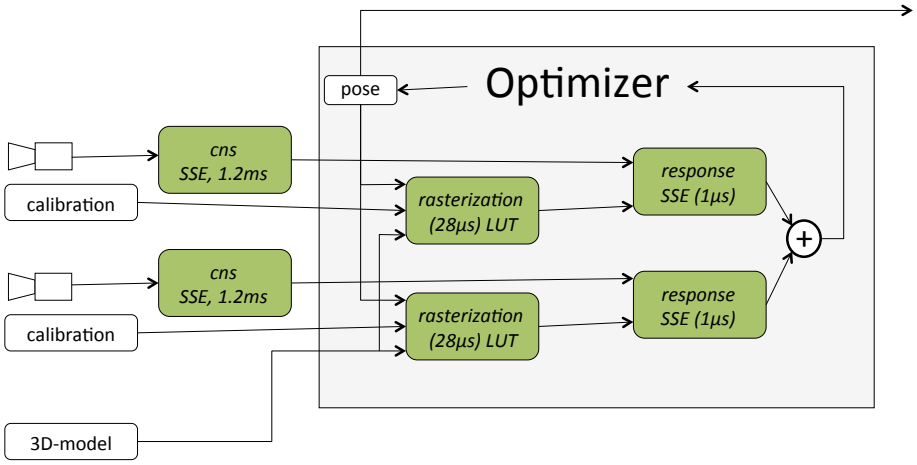


Fig. 3. Dataflow overview of our stereo contour-based object detector

be marked by a color label and edges between visible faces of different labels are also added.

Next, the vertices involved are perspectively projected into the image in an SSE implementation. The projection ignores distortion, which is ≈ 1 pixel only for NAO. The precomputed edge list is sorted such that projected vertices can be used twice.

Global Search Heuristic. The textbook solution for global object search would be to find the maximum response of all poses within the grasping space (6DOF). However, this is computationally beyond scope. Instead, we use an application-specific heuristic. We search only for a single cup orientation by assuming it is roughly vertically aligned and by removing the handle, making it rotationally symmetric. This orientation is obtained from the robot's forward kinematic.

For the position, we go through the image in patches of 64×48 pixels and rasterize the cup at several positions along the center pixel's ray. For each contour, 64×48 responses are computed and the largest overall response is refined.

Then, the cup rotation is determined by evaluating the response of several rotated cups with handle. Finally, the full model is refined. If the response exceeds a threshold (0.65), we switch to tracking mode. If the response in tracking mode falls below 0.5 for 15 frames, we switch back to the global search.

The global search takes $\approx$ 320ms, so we spread it over several frames, evaluating only between one and two 64×48 blocks in each frame (13–26ms).

Local Search (Refinement and Tracking). During local search, the optimizer (Fig. 3) changes the pose towards growing responses. This procedure is used for tracking as well as to refine a coarse initial pose obtained by our global

search heuristic. We use the simple approach to optimize DOFs round-robin one at a time, although there are of course more sophisticated optimization algorithms. However, we exploit that the response computation provides an array of 8×8 responses for shifted contours (2D translation).

So, to refine one DOF, we compute 8×8 responses around the original pose, around a pose changed on step in the considered DOF, and around the inversely changed pose. The subpixel-refined maximum of these $3 \times 8 \times 8$ responses defines the new pose. Therefore the image translation must be converted into a change of pose. This is approximated by a rotation of the object around the camera which moves the object's center in the image according to the obtained image translation.

As image translation is already covered, the 4 remaining DOFs are translation in viewing direction and object rotation around X, Y, and Z (skipped in case of symmetry). The step size is roughly determined to create 3 pixel changes in the image based on object size and distance.

We noticed that the convergence is fairly robust (Sec. 6.3). This motivates to use only one orientation in the global search.

5 Object Manipulation

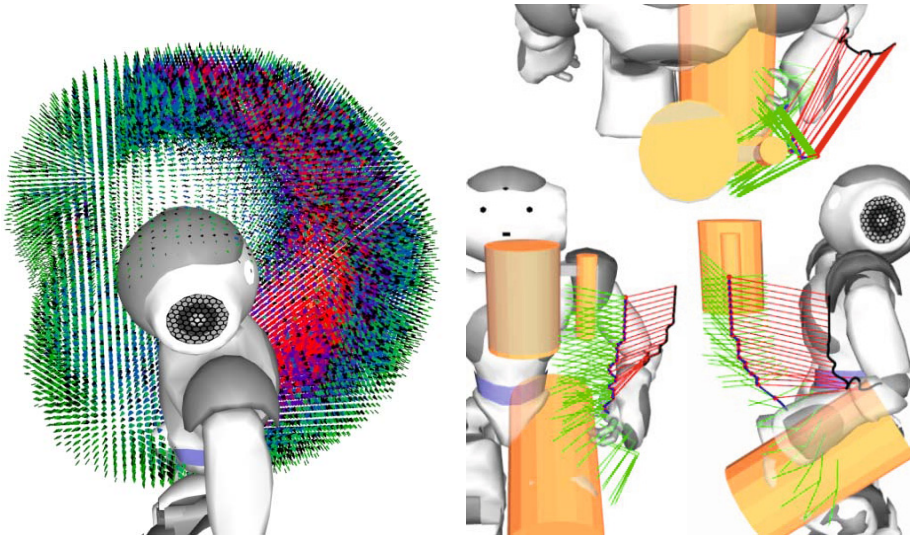
On the motion side there are two tasks to solve: grasp an object from a human hand and give it back if the human asks for it. In order to grasp an object, we need to calculate a valid motion path from the hand position to the target and avoid obstacles such as the object itself or body parts that may be in the direct path. By doing so we were using the grasp planner of our previous work [2]. Because we had problems with the overlap of detection and grasp space, we modified our old planner to calculate possible body positions online so that the grasp space can be increased without increasing the calculation time. We also added a least-square Bezier fit algorithm in order to smooth the planned way points and increase the execution speed.

The releasing task is to transfer an object to the human hand once a human asks for the object. Thereby, we are using a sensor feedback solution in order to detect whether it is safe to release the object.

5.1 Object Grasping

As described in more detail by Müller *et al.* [2] our grasp motion planning approach is based on a predefined reachability map. This reachability map is a discretization of the workspace with a cube that is divided into equally sized smaller cubes. Each sub cube serves as a region in the workspace. Each region stores a set of reachable lower arm directions for that position (1 DOF).

Thereby we use 4 of the 5 DOF of NAO's arms to define a certain hand pose, and handle the wrist-DOF later. This leaves only one DOF of four for the lower arm direction while a fixed hand position is commanded, less than the two possible DOF. The fifth DOF represents the wrist angle and has only minor



(a) Reachability Map (b) Grasp Path

Fig. 4. (a) The best reachable regions are marked in red, less well-reachable regions are marked in blue, and badly reachable regions are marked in green (b) The linear interpolated motion path of the hand and the elbow with obstacles present

influence on the planning. Because the wrist rotation can be calculated later from the lower arm direction and the joint limits, we only need to store a set of possible lower arm directions per region instead of a set of full hand orientations. Figure 4(a) pictures the reachability map used, where only reachable directions per region are marked.

According to that, the reachability of NAO's hand is clearly very limited and the lower arm direction depends on the hand position. For that reason it is necessary to check for each grasp pose and each point on a motion path whether it can be reached. This leads to the problem that a large number of reachability checks are necessary for motion planning. This can be sped up best by predefining the workspace in a reachability map.

The origin of the reachability map is located in the shoulder of the robot. Thus, it is possible to test with different shoulder positions whether a certain hand position is reachable without the use of inverse kinematics.

Grasp Motion Planning. The first step of the grasp planning is to evaluate a range of grasp poses by using the map. Once a reachable grasp pose is found, the grasp planner plans a path through the grid cells of the map. The reachability map provides the planner with 6D information on the possible hand positions and lower arm directions. Since planning in 6D is very expensive, our A*-based planning algorithm initially only uses the 3D area grid and considers the lower

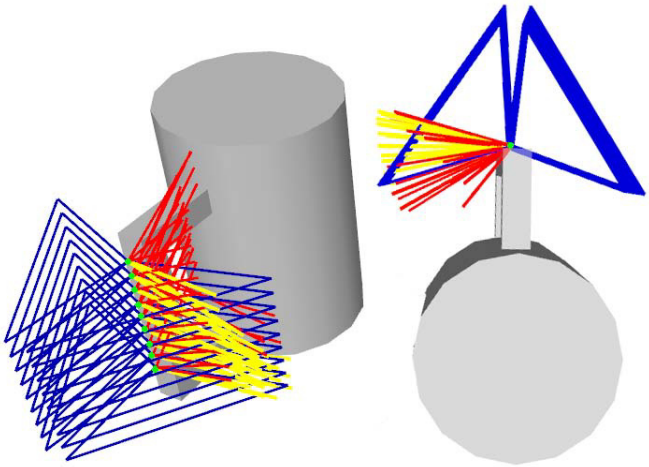


Fig. 5. Each object has its own grasp map, which is generated from a set of predefined grasp rules. Each rule connects a range of lower arm directions to a grasp position (blue). The reachability map (red) is matched to the object's grasp map. Matches are marked yellow.

arm direction only via the cost and heuristic function. Thereby, to be evaluated, nodes are checked for reachability and obstacle collision in order to calculate the heuristics only for verified nodes. In this process, nodes with more suitable lower arm directions are rated better than nodes with greater deviations from the lower arm goal direction. Also the distance between the node evaluated and the goal node in 3D are taken into account.

The output from the planning algorithm is a list of waypoints through the reachability map, which are represented as red dots in Fig 4(b). Since there is a dependence between the hand positions and the directions of the lower arm, a waypoint also includes a direction. Each direction defines the elbow position corresponding to the waypoint and is marked by red lines in Fig 4(b). The final hand orientation is defined by the grasp selection rules as described in the next section and is calculated from the lower arm direction, forward kinematics, and the object pose.

Grasp Selection. In our previous work we selected grasps by testing with a certain amount of predefined body poses, i.e. shoulder positions, whether the resulting reachability map matches with a set of predefined grasp rules. Those grasp rules indicate how an object can be grasped in order to find a suitable grasp point on the object. Each rule is defined by a grasping point and a range of lower arm directions and final hand rotations relative to the object. In Fig. 5, grasp rules are marked with blue triangles. The green dots constitute the position where to grasp and the triangle defines a range of lower arm directions.

In order to select a grasp, the grasp rules are matched with the reachability map. In this process, areas that include a grasping point are examined further

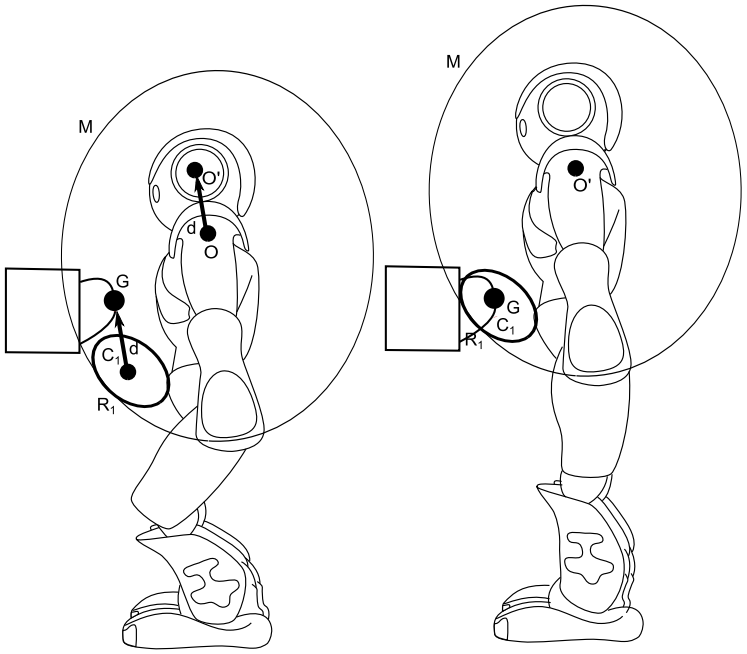


Fig. 6. Schematic depiction of the body pose calculation, M represents the reachability map, O the map origin, R_1 the region of similar cells of the map, C_1 the summed center of a region and d is the translational offset between current and target position G

in order to check whether the corresponding possible lower arm directions are qualified for the grasp. In this process, the possible lower arm directions of the grasp areas are compared to the angle ranges from the grasp rules. The best match is selected.

In our previous approach the search space increases with the amount of predefined body poses and can slow down the planning process. For that reason we investigated – similar to Zacharias *et al.* [19] – each region in the reachability map in detail.

Since the DOF of NAO's arms are very limited, we discovered that we have certain ranges of regions that are very similar to one another, mostly cone-like. So in our new approach, we calculated a fixed amount of center points of similar regions. These center points are used to calculate the body offset between the grasp points defined by the grasp rules and the current body position as it is demonstrated in Fig. 6. The body poses are calculated online by inverse kinematics for a fixed amount of positions. In doing so, we were able to keep the calculation time constant but could increase the grasp space heavily – especially for the highly reachable regions.

Since the robot has two arms, each body offset is calculated for each arm respectively. Thereby body positions, which lead to shorter distances between hand and grasp pose, are rated inferior.

Motion Path Execution. In order to smoothly execute a motion path, it is necessary to minimize the velocity discontinuities at the way points. This can be done by approximating the way points with cubic Bezier curves instead of lines.

In our previous work, we were converting a found grasp plan into a Bezier spline by using the method by DeRose *et al.* [24] in order to generate a trajectory for our motion engine based on the work of Müller *et al.* [25]. In that approach we connected each way point to the next with a cubic Bezier curve using a fixed duration for each sub curve. Although the smoothed path was free of velocity discontinuities, we discovered the problem that sub curves that were short in distance led to a slower movement than sub curves that were longer in distance. As a result the hand accelerates and decelerates unnecessarily.

In order to overcome this problem, we added a least square Bezier fit method as described by Itoh *et al.* [26] and Herold [27] in order to initially approximate all waypoints with a single cubic Bezier spline. This method is using the percentage of the length between each adjacent point of the path with equation

$$t_i = \frac{|d_i - d_{i-1}|}{\sum_{j=1}^{n} |d_j - d_{j-1}|} \tag{1}$$

to synchronize the points with the cubic Bezier curve function

$$B(t, C) = c_0(1-t)^3 + 3c_1 t(1-t)^2 + 3c_2 t^2(1-t) + c_3 t^3 \tag{2}$$

with $C = [c_0, c_1, c_2, c_3]$. The distance d_i to the i-th point is defined by equation

$$d_i = \sum_{j=1}^{i} |P_j - P_{j-1}| \tag{3}$$

with $d_0 = 0$.

We used the residual sum of squares to calculate the error fit. In doing so, in equation

$$E(C) = \sum_{i=1}^{n} (p_i - B(t_i, C))^2 \tag{4}$$

we are summing the squared distance of each waypoint p_i to its Bezier curve approximation defined by the control points $c_0 \dots c_3$.

By setting the derivative of Equation (4) equal to zero, the control points $C = [c_0, c_1, c_2, c_3]$ with minimum error [28] can be found. With Equation (2) the best fitting Bezier curve is defined.

Since the plan is recalculated in each frame, we need to consider that the path changes even if parts of the old plan already had been executed. It is not possible to change a Bezier curve $B(t, C)$ after a certain t^* without changing the whole curve. For that reason we need to split the path at the next point that is to be executed before a replanning is done. Hence we are splitting the path at t^* by using De Casteljau's algorithm in two curves: already executed ($ae(t)$ in Fig. 7) and to be executed ($be(t)$ in Fig. 7).

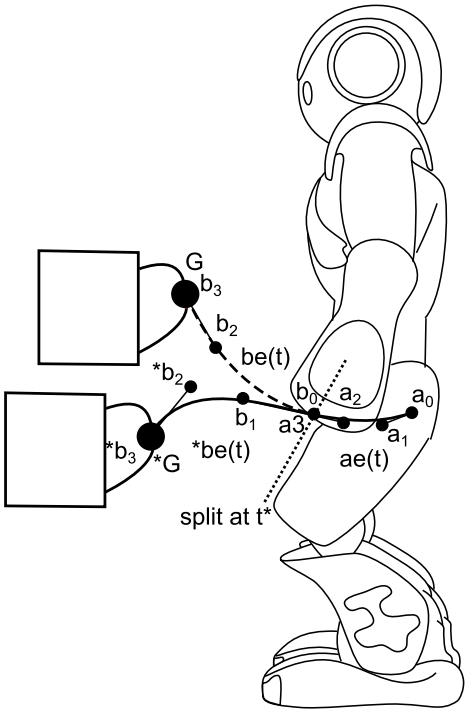


Fig. 7. Schematic depiction of the curve splitting at t^* during the path recalculation; the initial curve from a_0 to G is splitted at the next position to be executed. The new plan is approximated by ${}^*be(t)$ and connected with the already executed curve $ae(t)$.

The last control point of $ae(t)$ is used as new start point to replan the path to the current goal point. The resulting way points are converted to a new Bezier curve (${}^*be(t)$ in Fig. 7) with the condition that the first two control points are fixed in order to keep continuous velocities in the connection point.

Since a plan can be longer or shorter after the replanning, a duration update is also necessary. In doing so, we calculate in each frame i with equation

$$o_i = o_{i-1} \frac{|d_{*be(1)}|}{|d_{b(1)}|} \tag{5}$$

the change of the path length. Thereby we multiply the previous remaining duration o_{i-1} with the ratio between the path length of the previous sub curve $be(t)$ (with path length $d_{be(1)}$) and the updated curve ${}^*be(t)$ (with path length $d_{*be(1)}$).

5.2 Object Releasing

The aim of the release function is to transmit the grasped object to the human in a safe way. The safety mainly concerns the manipulated object that can break

if it falls down during the transmission. But it can also concern the safety of the human user: an elderly or immobile person, whom the balance can be unstable or destabilized if the object he should get from the robot falls down during the transmission. We wanted to make sure that the robot opens its hand when the object is correctly caught by the human partner.

The main idea is to detect that the object is pulled from the hand of the robot before releasing it. Because NAOs fingers are not equipped with force sensors, it is not possible to detect the traction force applied by the user pulling the object directly with the fingers. But the traction force is transmitted to the arm, through the rigidity of the wrist. By reducing the stiffness of the arm joints, it is possible to detect the traction force by an unexpected motion of the arm joints. When the robot wants to give its object back, it monitors the position of the arm carrying the object. As soon as this arm moves the robot detects that a traction force is applied on the object and opens its hand.

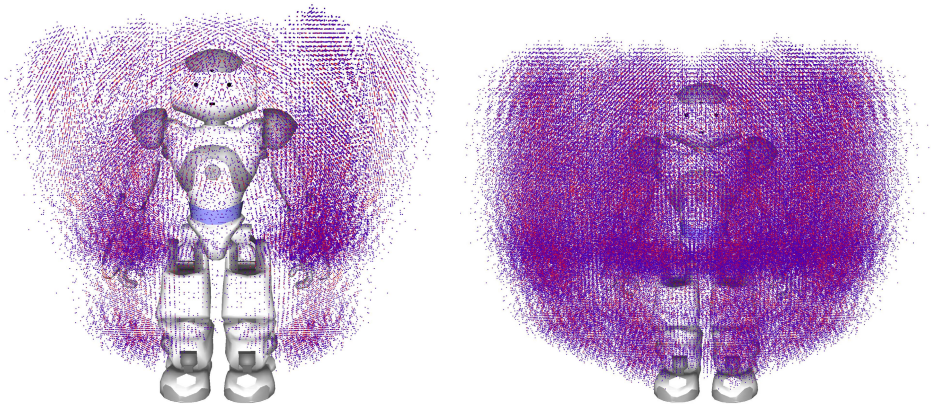
Of course reacting to an unexpected force with letting go of the object is a dangerous behavior. So we require the user to say *"give it to me"* to put NAO into object return mode and also NAO needs to detect the users hand. This detection is made by one of the two modalities: tactile or vision. The robot expects to see the hand of the user close to its own hand or detects with the tactile sensor on the back of its hand that the user touches its hand. If one of these conditions is fulfilled, the robot can safely open its hand to release its object. To safely release an object in an intuitive way it appears that the robot needs to combine four modalities of perception: audio (*give it to me*), kinesthetic (unexpected motion of the arm), vision (detection of the hand) and tactile (contact of the back of NAO's hand).

When the robot is walking with the object in its hand to bring it to its user, it may happen that the object slips from the fingers and falls. NAO is able to detect this event as well thanks to its proprioceptive sensors. To grasp an object, NAO tries to close its hand to its maximum closed position. Because the object does not allow the complete closing the goal position of the finger is not reached. The difference between the expected finger position and the actual finger position indicates if the object has been grasped or not. If the robot, supposed to be carrying an object, detects that its finger are in the maximum required closed position, it means that the object is no more in its hand.

6 Experiments

The GRASPY experiment concerns the exchange of objects between NAO and a human. The objects to grasp are cylinder-like objects such as a pen or a cup with handles. The places where the robot should grasp them from should be from a human hand or from a table.

The man machine interface is a simple dialog, in which the human tells NAO to grasp a certain object. In case the object is reachable by the robot, NAO confirms this and grasps it. If the robot has the object in its hand, the human may tell it to give it back. Only if the robot has an object in one of its hands



(a) (b)

Fig. 8. (a) Depiction of grasp space when using only four fixed positions (b) Depiction of grasp space when body position is calculated online

it offers it by speech and gesture to the human. Once the human touches the object it will be released by the robot.

The evaluation criteria are the time to grasp an object, the number of wrongly detected objects, and the number of successfully grasped objects. All experiments were made on a Nao robot using its Intel Atom (1,6 GHz) processor with 1 GB SDRAM.

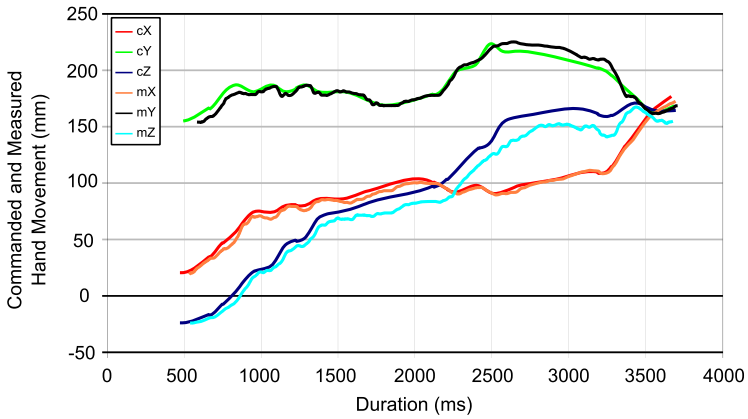
6.1 Planning

We evaluated the planning algorithm with the new grasp selection function and compared the results to the results of our previous work. In previous last work we reached a calculation time of 20 ms per frame by using a heuristic and cost function that combined the translation distance between the nodes and the goal with the differences in the lower arm directions per node.

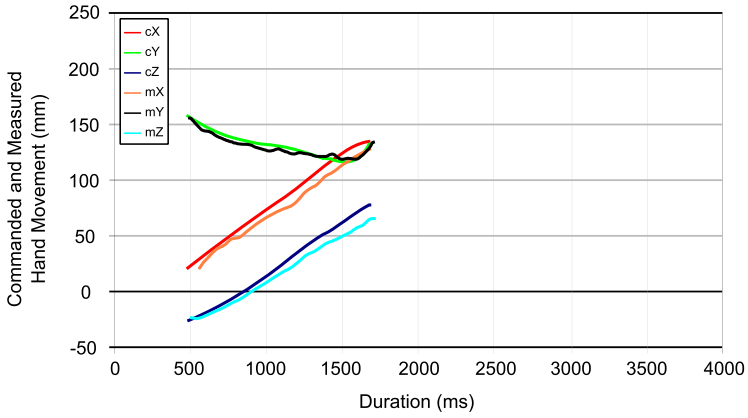
In our new experiment we used the same parameters and the same planner with the extension that the body pose is calculated online. We calculated four possible body positions per arm per frame and tested whether the object can be reached. The average calculation time per frame is 21 ms, which is almost the same as in our previous experiments but the amount of good regions in the overall grasp space could be heavily increased as it can be seen in Fig. 8(a) and Fig. 8(b).

6.2 Motion Path Execution

We also compared the execution of the smoothed paths. Thereby we recorded the commanded hand position (cX, cY, cZ) and the measured hand position (mX, mY, mZ) per frame.



(a)



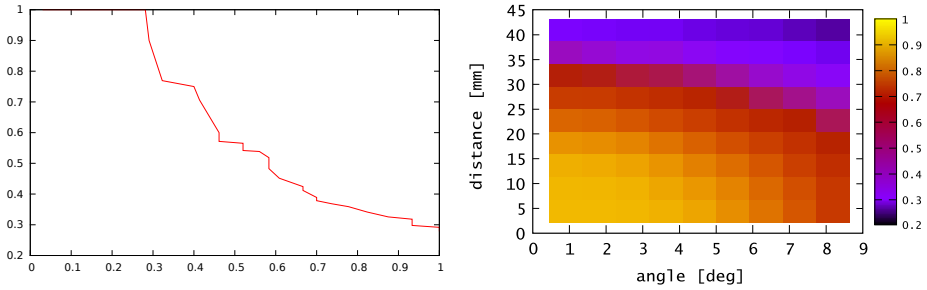
(b)

Fig. 9. (a) Comparision of commanded and measured hand motion using our old smoothing method and (b) using the presented least square Bezier fit method

Figure 9(a) depicts a motion path produced by the planner with the smoothing method of our previous work. Although that path was converted into a Bezier path there are a lot of passages where the commanded direction changes a lot.

Figure 9(b) depicts the commanded and measured hand motion of a similar but slightly shorter motion path smoothed by our new method. The commanded path is clearly straighter and the error between commanded and measured hand position on the y-axis is smaller. The deviation on the x- and z-axis results from the backlash of approximately $\pm 3°$ of NAO's arm motors in combination with the weight of the arm.

Due to the much straighter motion paths, we also increased the execution speed. This reduces the duration of the whole grasp.



(a) ROC curve (b) Range of Refinement

Fig. 10. (a) Precision over recall for the stereo image based cup detector (b) Probability of the local refinement to converge into the true pose as a function of the angular and translational distance between starting and final pose. The probability is computed with 100 tries in each image. The cup is $95 \times 75mm$ large, so nearly half a cup-diameter in the initial guess leads to a good final pose.

6.3 Stereo Contour-Based Object Response

We evaluated the contour-based stereo detector on a set of 53 images taken in a cluttered office environment. Figure 10(a) shows a roc-curve of the detector. In our opinion the performance is good given the highly cluttered scenes and the fact that often the cup is only partially visible in the image and partially occluded by the hand. Figure 10(b) shows that the detector has a rather large range of convergence, which allowed us to perform the global search efficiently with a rather coarse grid and only a single orientation.

Computation time of the detector is 2×1.2ms for the cns computation, 28μs for rasterization of one pose in one camera and 1μs for response evaluation of one contour, when always blocks of 8×8 contours are evaluated.

6.4 System Level Experiments

Since we could decrease the time to execute a grasping motion, we repeated the system level experiment. In our previous work we did 30 experiments and tested whether a cup could be grasped and how long it took. In doing so, we recorded the time between the first detection and the successful grasp in a normally illuminated simple office environment. We discovered that the average grasping time is now 7.11s instead of 10.026s. In addition, each try was successful.

During the trials we also measured the timings of the object detector. Both cns images are calculated with an average time of 2.5ms per frame, the global search with 21.6ms per frame and the refinement in 1.8ms per frame.

7 Conclusion

We successfully improved our grasp planner by increasing the grasp space but kept the calculation time constant. The whole system operates at 30Hz. We also

were able to improve the motion execution by decreasing the duration of the full execution and achieving a smoother movement. In addition, we introduced a releasing function, which completes the whole system and converts it into a prototypical application.

In future work we are planning to include the possibility to grasp an object two-handed. The variety of graspable objects would increase instead of being limited to objects with a handle, because this would enable the robot to perform force-closure grasps. Another point we are planning to investigate is to measure the weight of grasped objects and the impact on the robot's walk. The goal will be that NAO carries an object around in order to bring it to someone or somewhere.

Acknowledgements. This work has been funded by the European Commission in the 7th Framework Programme for RTD in the context of the project "ECHORD – European Clearing House for Open Robotics Development" under the contract number FP7-231143 and by DFG under grant FR2620/1-1 and SFB/TR 8 Spatial Cognition.

References

1. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 769–774 (May 2009)
2. Müller, J., Frese, U., Röfer, T.: Grab a mug - object detection and grasp motion planning with the Nao robot. In: IEEE-RAS International Conference on Humanoid Robots (2012)
3. Azad, P., Asfour, T., Dillmann, R.: Stereo-based 6D object localization for grasping with humanoid robot systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, October 29-November 2, pp. 919–924 (2007)
4. Cotugno, G., Mellmann, H.: Dynamic motion control: Adaptive bimanual grasping for a humanoid robot. In: Proceedings of the Workshop on Concurrency, Specification, and Programming CS&P 2010, Börnicke (near Berlin), Germany, vol. 2 (September 2010)
5. Stuckler, J., Holz, D., Behnke, S.: Robocup@home: Demonstrating everyday manipulation skills in robocup@home. IEEE Robotics Automation Magazine 19(2), 34–42 (2012)
6. Hegger, F., Mueller, C.A., Jin, Z., Alvarez Ruiz, J., Giorgana, G.R.G., Hochgeschwender, N., Reckhaus, M., Paulus, J., Ploeger, P.G., Kraetzschmar, G.K.: The b-it-bots robocup@home 2011 team description paper. In: Proc. RoboCup Symp. (2011)
7. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004)
8. Stückler, J., Steffens, R., Holz, D., Behnke, S.: Efficient 3d object perception and grasp planning for mobile manipulation in domestic environments. Robotics and Autonomous Systems (2012)

9. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24(6), 381–395 (1981)
10. Rusu, R., Blodow, N., Marton, Z., Beetz, M.: Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, pp. 1–6 (2009)
11. Hsiao, K., Chitta, S., Ciocarlie, M., Jones, E.: Contact-reactive grasping of objects with partial shape information. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1228–1235 (2010)
12. Stückler, J., Behnke, S.: Following human guidance to cooperatively carry a large object. In: IEEE-RAS Int. Conf. Humanoid Robots (Humanoids), Bled, Slowenia, pp. 218–223 (2011)
13. Kuffner Jr., J.J., Kagami, S., Nishiwaki, K., Inaba, M., Inoue, H.: Dynamically-stable motion planning for humanoid robots. Autonomous Robots 12(1), 105–118 (2002)
14. Burget, F., Hornung, A., Bennewitz, M.: Whole-body motion planning for manipulation of articulated objects. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA (2013)
15. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
16. Harada, K., Kaneko, K., Kanehiro, F.: Fast grasp planning for hand/arm systems based on convex model. In: IEEE International Conference on Robotics and Automation, ICRA 2008, pp. 1162–1168 (May 2008)
17. Vahrenkamp, N., Asfour, T., Dillmann, R.: Simultaneous grasp and motion planning: Humanoid robot armar-iii. IEEE Robotics Automation Magazine 19(2), 43–57 (2012)
18. Zacharias, F., Borst, C., Hirzinger, G.: Object-specific grasp maps for use in planning manipulation actions. In: Kröger, T., Wahl, F.M. (eds.) Advances in Robotics Research, pp. 203–213. Springer, Heidelberg (2009)
19. Zacharias, F., Borst, C., Hirzinger, G.: Capturing robot workspace structure: representing robot capabilities. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, October 29-November 2, pp. 3229–3236 (2007)
20. Lalibertže, K., Birglen, L., Gosselin, C.M.: Underactuation in robotic grasping hands. Journal of Machine Intelligence and Robotic Control 4, 77–87 (2002)
21. Borst, C., Fischer, M., Hirzinger, G.: Grasping the dice by dicing the grasp. In: Proceedings. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2003, vol. 4, pp. 3692–3697 (October 2003)
22. Kragten, G., Kool, A., Herder, J.: Ability to hold grasped objects by underactuated hands: Performance prediction and experiments. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 2493–2498 (May 2009)
23. Aptina, `http://www.aptina.com/products/soc/mt9m114/` (accessed: June 03, 2013)
24. DeRose, T.D., Barsky, B.A.: Geometric continuity, shape parameters, and geometric constructions for Catmull-Rom splines. ACM Trans. Graph. 7(1), 1–41 (1988)
25. Müller, J., Laue, T., Röfer, T.: Kicking a ball – modeling complex dynamic motions for humanoid robots. In: Ruiz-del-Solar, J., Chown, E., Ploeger, P.G. (eds.) RoboCup 2010. LNCS (LNAI), vol. 6556, pp. 109–120. Springer, Heidelberg (2010)

26. Itoh, K., Ohno, Y.: A curve fitting algorithm for character fonts. Electronic Publishing 6(3), 195–198 (1993)
27. Herold, J.: Least squares Bezier fit, `http://jimherold.com/2012/04/20/least-squares-bezier-fit/` (accessed: June 03, 2013)
28. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C the art of scientific computing, 2nd edn. Cambridge University Press, New York (1992)