2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Real-time collision detection for intrinsic safety of Multi-fingered SDH-2

Thomas Haase, Prof. Heinz Wörn
Institute for Process Control and Robotics
Karlsruhe Institute of Technology (KIT)
D-76131 Karlsruhe, Germany
haase@kit.edu

Abstract— **This paper presents an algorithm to detect finger collisions in Multi-fingered robot hand SDH-2. The need for this feature is discussed and advantages are shown. It will be presented, in which way the algorithm is build into a basic development environment of reactive grasping and what are possible problems and possibilities in collaboration with the tactile sensor elements.**

I. INTRODUCTION

Nowadays various Multi-fingered robot hands are state of the art in the worldwide robot research. Unfortunately not even one of them fulfills industrial needs. All of them are still fields of research and will be for many years. This is not caused because of missing ideas, by no means. This is based on the fact that necessary software is still absent or incomplete. In addition, for an industrial application the hands are still too valuable. The financial value of each robot hand slows down the developing work. Codes are analyzed repeatedly, each experiment is simulated several times and proceeds with little velocity only to guarantee not to injure the expensive technology or to constitute down times. Collision avoidance algorithms for robots are state of the art in actual robot research [4],[6]. Even collision detection for robot hands is no new valued idea. Already in existence is an algorithm described in [2] that uses cylindrical approximations for each finger. By means of calculated distances they are able to avoid finger collisions. Unfortunately the cylindrical representation doesn't offer high precision. In [3] a collision detection is used to achieve contact detection in teleoperating systems. The problem is reduced to interferences between planes, spheres and lines. No finger geometry is reproduced. [5] pursues an approach of detecting distances and collisions with the help of photo-sensitive devices and light emitters embedded on the surface of a robot hand. However it doesn't seem to be useful to attach additional sensor elements on the SDH-2 surface. On the one hand it's not possible to integrate additional sensors into the SDH-2 body. Moreover all joints may not be constrained. But even each lateral attachment of secondary sensors reduce the mobility of the SDH-2. Additionally the tactile sensors used by the SDH-2 do not permit attachments. Concerning this matter, a collision algorithm was developed that use nearly exact finger geometry to prevent the SDH-2 fingers to collide with each other. This paper presents the principal mathematical design and the use of existing algorithms (SWIFT). Exploiting existing codes speed up the development time. The resulting advantages will be discussed as well as further possibilities in working with the SDH-2. More detailed information about the SDH-2 can be seen in [1].

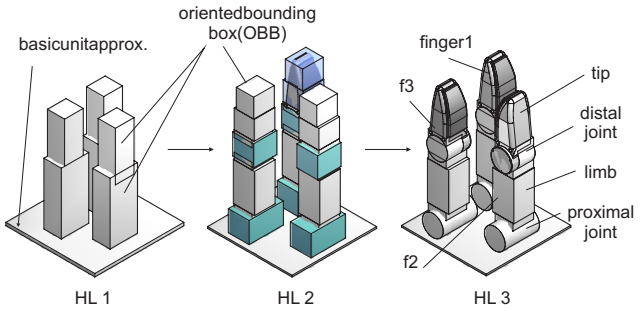

Fig. 1. Hierarchy System

A. Goals and Possibilities

Arbitrative for the design of that collision detection algorithm are the resulting code size and the calculating time. If the algorithm fulfils both demands, the code could be exported within the SDH-2 and runs on the integrated controller. The required hardware, that is necessary for it, is already integrated in the body of the hand. It would be an asset, if nary researcher has to care about a possible collision risk.

In principle, building up collision detection for multi-fingered robot hands could be solved in various ways. One possibility is to use given libraries and exact CAD geometries to check each feasible hand position for collision. Section III-D demonstrates an example of how to realize that. The results could be stored in a look up table and are read during execution. This makes only sense for hands with a small number of degrees of freedom. Otherwise it's a highly time and memory consuming solution. Supposed doing that on SDH-2 (7DOF, $6 \cdot 180°$, $1 \cdot 90°$, $\Delta\phi = 1°$) and supposed that each hand position needs $t = 0.5ms$ to be checked for collision and assumed we only save one bit (0 = no collision, 1 = collision) it follows:

$$\begin{aligned} p_{ositions} &= 180^6 \cdot 90 \approx 3.06 \cdot 10^{15} \\ t_{ime} &= 3.06 \cdot 0.5 \cdot 10^{15} \cdot 10^{-3} s = 1.53 \cdot 10^{12} s \\ m_{emory} &\approx 356 \cdot 10^3 Gb \end{aligned}$$

This option is not possible at all. Another alternative solution for discovering collisions is the decision tree. The assignment is to find out, which joint angles leads to certain hand

positions. However, this is only for a small number of degrees of freedom manageable, even seven degrees are too much. It is caused by unknown attribute values [11] - not each possible robot hand configuration can be predicted. That implies, that not each situation is familiar and defined and uncertain settings could appear. An alternative left is the approximation of each finger and pick up existing and fast collision algorithms. The development of such an algorithm is described in the following section.

II. SDH-2 Architecture And Approximation

A geometrical model is needed to represent the SDH-2. The developed collision detection is subdivided into three hierarchy levels. Figure 1 represents them. The way of approximating an element of a finger depends on the actual hierarchy level (HL). HL 1 and 2 are used to determine collision free situations with minimal computational effort. The predominant majority of all finger positions will not cross the steps. Only HL3 is able to detect real and exact collisions at the expense of more computing time. Hierarchy Level 3 in Figure 1 represents the general partitioning. Each finger consists of the lower proximal joint, the limb, the upper distal joint and the fingertip. Since all fingers have same physical dimensions, only one finger approximation is required. All deformable geometry based on tactile sensor elements is not taken into consideration. Additionally the clearance distance 's' is introduced. Thereby, the finger approximation can be enlarged in comparison to the real geometry.

A. Hierarchy Level 1 (HL1)

Level 1 is constructed with the help of 6 oriented bounding boxes (OBB's). With given joint angles, each bounding box is checked for collision using the "separating axis" algorithm [8],[9]. Without break condition and in spite of a given separating line the "separating axis" algorithm requires only $4-7\mu s$, cp. [8]. HL 1 requires an *average* calculation time of $t_{avg} = 36\mu s$ if no collision is found - (Matlab Code, Dell Optiplex 745). HL1 is equivalent to the decision tree that was mentioned in section I-A. HL1 is not able to find exact collisions but is useful to determine permitted robot hand positions. For that reason, HL1 minimizes computational effort in secured positions. The maximal approxiamtion error of HL1 is $A_{max} = 9.5mm$ (s=0).

B. Hierarchy Level 2 (HL2)

HL2 is entered, if a collision in HL1 is found. HL2 does not rebuild a whole SDH-2. Only those two bounding boxes that collide in HL1 are transferred into that higher resolution and tested again. Each finger is partitioned into 8 bounding boxes, even all joints are described as OBB's. All fingertips are represented by two rectangles with a maximal approximation error $A_{max} \approx 8.8mm$ (s=0). An OBB from HL1 is represented by 4 or 5 new OBB's in HL2 - the distal-joint-OBB in HL2 is part of both OBB's in HL1. Thus, only two half fingers are transformed into HL2. The average calculation time t_{avg} of HL2 is $t_{avg,HL2} = 59\mu s$.

TABLE I
Approximation Accuracy

recursion depth b	3	4	5	6	7	8
number n of OBB's	8	16	32	64	128	256
A_{max} in [mm]	3.3	1.74	0.9	0.46	0.23	0.12

C. Hierarchy Level 3 (HL3)

Each joint is described as a horizontal cylinder. The remark "horizontal" is discussed in section III-A. The remake of each fingertip is realized with a high number of OBB's. Figure 2 demonstrates the body structure of a fingertip approximation with 30 OBB's. The physical dimensions are directly determined from the CAD data of the manufacturer. Additionally, a clearance distance 's' can be defined that scales up the approximation. In figure 2 the clearance distance is $s = 5mm$. The fingertip is not recreated with the whole

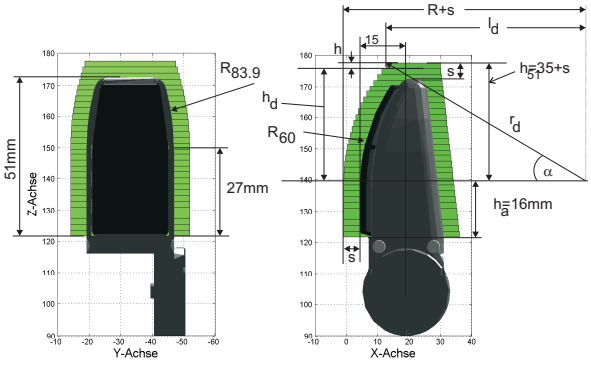

Fig. 2. Fingertip Approximation

number of bounding boxes. In fact, the collision algorithm is recursive. At the beginning, one bounding box is used to approximate the fingertip, cf. HL1. If no collision is detected, there is no need to raise the resolution and the calculation time is decreased. Otherwise the first approximation is replaced by two more detailed bounding boxes (HL2). If a collision with one of them still exists, it is replaced again until a specified depth of the recursion is reached or no collision is detected. The recursion depth determines the approximation accuracy. The more OBB's are used to approximate the fingertip, the merrier is the approximation accuracy. A_{max} defines the maximal approximation error and is calculated with equation (2) and demonstrated in Table I:

$$n = 2^b \quad (1)$$

$$A_{max} = r_{d,max} - (R+s) \quad (2)$$

$$r_{d,max} = \left(\sqrt{h_{51}^2 + l_d^2}\right)_{max} \quad (3)$$

$$l_d = \sqrt{(R+s)^2 - h_d^2} \quad (4)$$

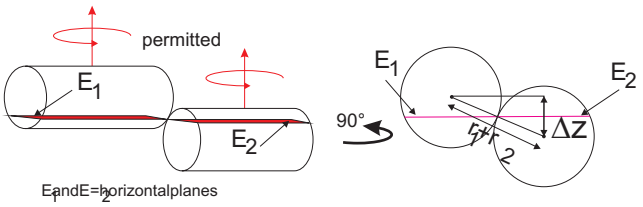

Fig. 3. Joint - Joint Collision Detection

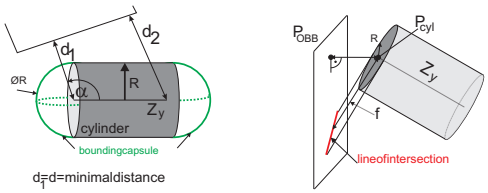

Fig. 5. Fingertip - Joint Collision Detection

III. Collision Detection Algorithm

A. Joint - Joint Collision

A joint - joint collision is reduced to a cylinder - cylinder collision with an additional condition: Each cylinder is always in horizontal alignment. In addition they are allowed to rotate about their longitudinal axis without influencing the collision detection, fig. 3. The vertical distance between two joints Δz is given in their transformation matrices. There is definitely no collision, if the vertical distance is greater than the sum of both radii ($\Delta z > r_1 + r_2$). Otherwise, both joints potentially collide at $\pm\frac{\Delta z}{2}$. This is due to the fact that all joints have the same diameter. That obviously simplifies the collision detection because each cylinder is now described as horizontal 2D plane (E_1, E_2). A 2D - separating axis algorithm verifies the collision ($t_{avg,JJ} \approx 3\mu s$).

B. Fingertip - Fingertip Collision

As mentioned before, each fingertip is built up with a number of bounding boxes. A collision detection between two fingertips with 30 OBB's each required $n = 30 \cdot 30 = 900$ passes of the separating axis algorithm and is expensive in calculation time. Using a recursive algorithm is a suitable solution to reduce the number of tests, cp. figure 4. At the beginning, two bounding boxes are used to approximate the fingertip, cf. HL2. If a collision is detected, each corresponding OBB is decomposed into two more detailed bounding boxes and checked again. If collisions with one of them still exist, it is replaced again until a specified depth of the recursion is reached. Secure bounding boxes stay untouched, cp. figure 4 (OBB(a) and OBB(b)). They definitely don't clash.

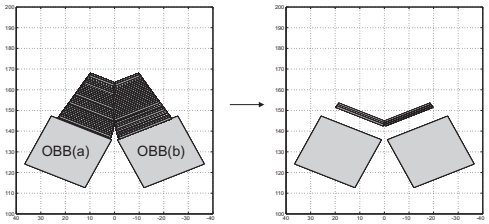

Fig. 4. Fingertip - Fingertip Collision Detection

C. Fingertip - Joint Collision

The collision detection between a joint and a fingertip is most sophisticated because both shapes have structural discontinuities. In particular the mathematical description of the joint surface is not trivial. Therefore, an algorithm for detecting intersections between OBB's and cylinders in 3D has to be found. [12] and [13] gives an introduction of how to realize a computation ,fig. 5:

1) Determine the closest Points P_{cyl}, P_{OBB} and the minimal distance d between the longitudinal axis of the cylinder z_y and each plane of the OBB.
2) If: $d > R$ the bounding capsule and therewith the cylinder do not intersect with the OBB.
3) Else: Determine α. If $\alpha = 90°$ a collision is confirmed. Otherwise the cylinder end cap includes the minimal distance point, fig. 5. An intersection is not established
4) Calculate the line of intersection and determine the minimal distance f to P_{cyl}.
5) If $P_{cyl} < R$: determine the line segment inside the cylinder end cap. Verifying that segment is in contact with the OBB plane concludes, that they collide, elsewhere not.

D. Example Of Integrating External Collision Libraries

Finger 2 is able to get in contact with the basic unit of the SDH-2, fig. 1. The problem occurs if the combined joint 0 between finger 1 and 3 opens space for finger 2. Figure 6 demonstrates two possible finger positions, one that is permitted (red) and another one that is interdicted (blue). To intercept that case it's not reasonable to program a possible collision between finger 2 and an exact replica of that basic unit. By reason that this case is extreme unusual the average calculation time of the whole algorithm should not be raised by this. If a collision occurs only depends on the proximal ϕ_3 and distal ϕ_4 joint angles of finger 2: $collision = f(\phi_3, \phi_4)$. With the given CAD geometries of finger 2 and the basic unit and a collision detection library (here: SWIFT) it is possible to calculate iteratively the minimal distance between these two objects. Figure 6 presents the results. One can see that the threshold between collision and no collision could be approximately described as straight line:

$$\phi_4 = -1.6 \cdot \phi_3 + 209 \tag{5}$$

In addition, if $\phi_3 < 74$ no collision can be occur. By simply check one or both terms exact collision detection between finger 2 and the basic unit is realized:

```
if((phi3 > 74) && (phi4 > (-1.6*phi3 + 209)))
collision =1;
else
collision =0;
end.
```

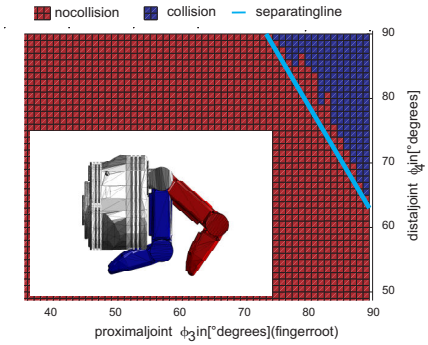

Fig. 6. 2D Collision Field of Permitted and Interdicted Finger Positions

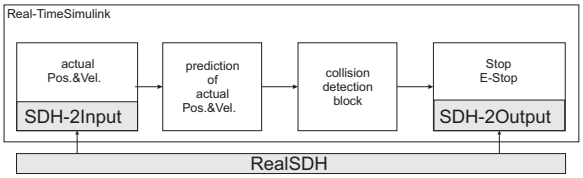

Fig. 7. Integrated Collision Detection

IV. EXPERIMENTAL RESULTS

At IPR, RTWT Simulink respectively RTAI Simulink is used to build up reactive grasping skills. The SDH-2 is integrated into such Simulink models with defined input and model output arguments. The control systems uses the SCHUNK SDHLibrary and reaches an average operating frequency f_o with $f_o = 30Hz$.

A. Simulink SDH-2 development environment

How to integrate the collision detection into an existing control system depends on how the robot hand is controlled. If, with a view to the actual position, only minor changes are generated at the control output, the collision detection should use these as input. If target positions with large movements are generated (e.g. PTP), the incremental changes at the control input are needed. Each incoming position data contains a delay time t_d. The received position is not equal to the real one. For that reason the use of the collision detection at the model output is preferred but not feasible on working with the SDH-2. This is due to the integrated control system that assumes the path planning between two positions and offers incremental changes at the SDH-2 output. Figure 7 illustrates the realized integration of the collision detection. Each model uses the input parameter of the SDH-2. If the SDH-2 is moving, each $t_d \approx 30ms$ new positions and velocities are read. A maximal velocity $\dot{\phi}_{max}$ with effective $\dot{\phi}_{max} = \frac{100^\circ}{s}$ leads with the maximal finger length $l_{max} = 155mm$ to a possible joint angle $\Delta\phi_{max}$ and position error Δx_{max} of:

$$\Delta\phi_{max} = \dot{\phi}_{max} \cdot t_{d,max} = \frac{100^\circ}{s} \cdot 0.03s = 3^\circ \quad (6)$$

$$\Delta x_{max} = l_{max} \cdot \sin(\Delta\phi_{max}) = 8.11mm \quad (7)$$

Assuming that a collision takes place in ϕ_{ideal}. By means of equation 6 the measurable collision joint angle ϕ_{real} is defined with an inaccuracy $\Delta\phi_{max}$. Figure 8 illustrates some positioning errors that could be measured. All errors are inside of the controlled range but not predictable. Thus, the collision detection is velocity-dependent and the max. permissible error should be assumed. Due to equation (7)

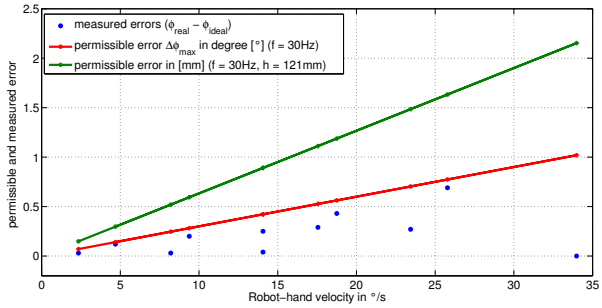

Fig. 8. Max. Permissible Position Error

the actual positions $\phi_{act,i}$ are tried to estimate. The SDH-2 provides 2 velocity profiles ramp and sin^2 in position controller mode. If ramp is selected as actual moving mode:

$$\dot{\phi}_{act,i} = \dot{\phi}_{in,i} + \Delta\dot{\phi} = \dot{\phi}_{in,i} + (\dot{\phi}_{in,i} - \dot{\phi}_{in,i}^{-1}) \quad (8)$$

$$\phi_{act,i} = \left[\frac{\Delta\dot{\phi} \cdot f_o}{2} \cdot t^2 + \phi_{in} \cdot t\right]_{t_{start}}^{t_{act}} + \phi_{in} \quad (9)$$

Otherwise a sin^2-function interpolates the hand velocity between the start and target position. Assuming that a movement starts at $t_s = 0$ and will be finished at t_e and with $t \in [t_s, t_e]$, $\dot{\phi}_i$ and $\phi_{act,i}$ are calculated as follows:

$$\dot{\phi}_i = \dot{\phi}_{max} \cdot sin^2\left(\frac{t}{t_e - t_s}\right)$$

$$\phi_{act,i} = \int \dot{\phi}(t)dt$$

Finally, if the velocity controller is activated, the predicted actual position is

$$\phi_{act,i} = \dot{\phi}_i \cdot T + \phi_{in,i}. \quad (10)$$

By means of that prediction the collision detection is

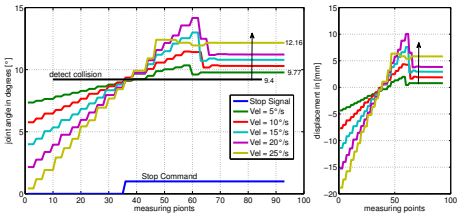

Fig. 9. Displacement Measurements STOP

achieved. The algorithm is able to influence the robot-hand stop- and emergency-stop-commands. Figures 9 and 11 illustrate the original SDH-2 behavior if a stop or emergency stop command is executed. All divergences are given in degree (left) and absolute (right) divergence. The absolute

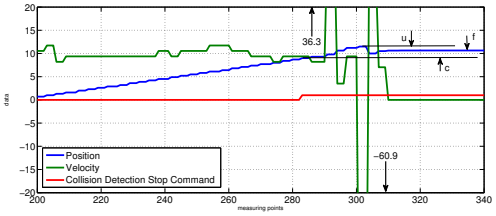

Fig. 10. STOP Command Position Control

divergence is variable and depends on the distance to the joint.

One can see that the stop command tries to achieve the original SDH-2 position at which the stop command was executed. Unfortunately that is combined with high velocities and it's not possible for the controller to achieve that position. Due to an error in the SDH-2 firmware the velocity is raised at the beginning of the control, figure 10. Thus, the current position seems to be lost and a higher value is adjusted. Associated with that high velocity, the finger doesn't stop actuating and reaches not predictable positions u and f, fig. 10. Both positions are not linearly dependent on the finger velocity. On this account and considering high finger velocities the stop command is unsuitable for testing the collision detection. The manipulation of the emergency stop command is not reasonable for each model. For testing the collision detection it is useful because it can be used to damp an inevitable collision. All fingers are stopped to be controlled and able to avoid malicious damages. Depending on the actual velocity the joints overrun the target position (fig. 11). The breaking distance is shown in figure 12. By

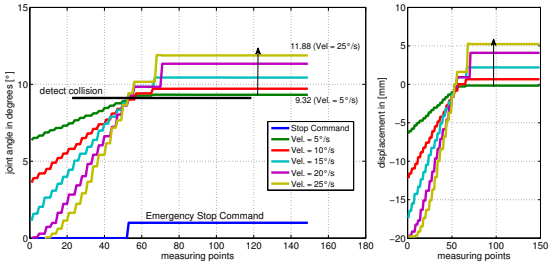

Fig. 11. Displacement Measurements Emergency-STOP

means of all measuring points in figures 11 and 12 it can adopt that a linear approximation of the breaking distance is available. It's possible to assign a supposed point of rest.

Figure 13 presents the improved accuracy of the emergency stop command from figure 11. A maximal joint angle displacement of $\Delta\phi_{max} = 0.55^\circ$ and therewith a total distance error of $\Delta x_{max} = 1.17mm$ is established. A velocity dependency is indistinguishable in spite of the given deviations. Depending on all velocities and if an actual position is predicted (fig. 7) the emergency stop command is now executed before the real collision is reached. It must be pointed out that the delay times are included within the measuring points from figure 12.

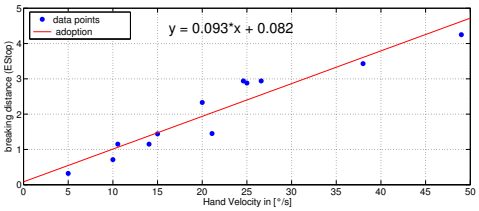

Fig. 12. Breaking Distance at Emergency Stop Command

TABLE II
CODE CALCULATION TIME

scenario	calculation time
Matlab M-Code	$\approx 874\mu s$
Simulink S-Function (C-Code)	$(12-142)\mu s$
Simulink RT Windows Target (1kHz)	$4\mu s$

B. Calculation Times

As mentioned the calculation time of the algorithm is one of the crucial properties. Collision detection is not a basic event in reactive grasping skills. It is only a module that makes it easier to develop new applications and therefore shouldn't need to much cpu calculation time. Table II shows the calculation times in different Matlab or Simulink scenarios. The compiled S-Function Simulink block is many times faster than the Java M-Code. The use of C-Code S-Functions instead of using embedded matlab functions is recommended. This is based on the fact that the embedded matlab is compiled during c-code export each time the model is reestablished. The Simulink RTWT-time is calculated on the basis of the percentage the RT-model needs the cpu to calculate a simulation step. Operating with $f_s = 1kHz$, a Simulink model requires $p_1 = 0.3\%$ of cpu time without and $p_2 = 0.7\%$ with the collision detection block. The achievable calculation times fulfill the required demands.

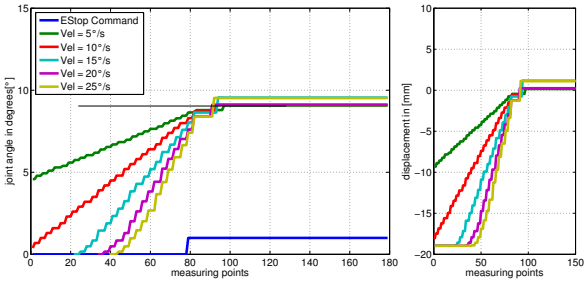

Fig. 13. Reduced displacement Measurements Emergency-STOP

C. Calibration Procedure

In fact each finger is able to move more than the defined $\Delta\phi_{max} = 180^\circ$. The calibration of the absolute position encoder is based on the maximal displacement of the joint. Thereby and by reasons of the general inaccuracy of the position encoder ($\frac{360^\circ}{1024} = 0,352^\circ$) one can conclude, that the proper finger positions differ from the ideal ones. The

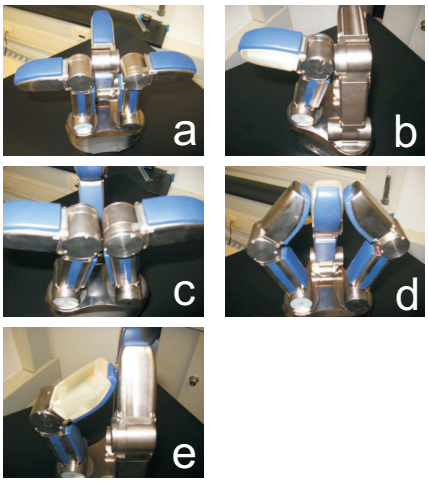

Fig. 14. SDH-2 Collision Calibration

collision algorithm diverges a little from the real finger position and orientation. With a given maximum length l of each finger ($l_{max} = 155mm$) a slight difference of only $\Delta\phi = 1^\circ$ effect a total error of $l_{error} = \sin(\Delta\phi) \cdot l_{max} = 2,7mm$. Real and ideal joints differ in range from $\Delta\phi = [0^\circ, 3^\circ]$. Offsets are introduced to match real and ideal joint angles. The identification of these offsets is realized with the help of clear positions and an existing SDH-2 simulation environment, fig. 14. Each distal - distal joint collision is used to determine the offsets of all proximal joints (a,b). Either a parallel orientation or a collision along a straight line between the distal joints of finger 1 and 3 is used to define and set their pivoting joint (c). Assuming that the distal and pivoting joints are exact, all fingertip collisions (d,e) lead to the missing proximal joint offsets.

D. Integration of tactile Sensors

All tactile sensor elements are responsive to contact. Grasped objects deform some sensor material and with it the external body structure of the SDH-2. All tactile areas shouldn't be part of the collision detection. The sensor material is $\Delta x = 1.3mm$ compressible. To integrate that deformability into the collision detection, the material is bared out. In doing so, the retraction depth Δx is arbitrary. In this way it might be possible to limit the maximum pressure on the sensor. Therefore, an exact calibration process for the joint positions is essential.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper Real-Time collision detection for the Multi-fingered hand SDH-2 was introduced. The general code construction and the implementation into the existing Simulink SDH-2 programming environment were proposed. It could be proved, that the algorithm keeps all Real-Time requirements. The proper advantage of this algorithm is the faster pace of development. Software engineers could now test some SDH-2 behaviors without being afraid of demolishing some fingers. The number of malfunctions will be reduced as well as supplementary costs. Safe working paves the way for extensive and complicated test series.

B. Future Works

There are additional possibilities to minimize the code and the number of mathematical calculations. The aim is to make the code be part of the SDH-2 library respectively the SDH-2 controller. Therefore, the algorithm has to be moreover as small as possible. Unfortunately, this condensed mathematical form will make the code no longer be readable or rather presentable. In addition, the calibration of each finger is not as fast as it should be. A calibration process has to be developed that works automatically and (economically) reasonable. Flexible procedures for movement optimization and path planning are supposed to be developed. Therewith, unnatural hand positions can be transferred into different new positions without finger contact. Furthermore, an interaction between the integrated tactile sensor matrices and the collision algorithm is imaginable.

VI. ACKNOWLEDGMENTS

T. Haase is supported by SCHUNK GmbH & Co. KG, Lauffen. Only by means of Dirk Osswald and Dr. Matthias Haag the Real-Time collision detection algorithm could be realized.

REFERENCES

[1] SCHUNK SDH-2, http://www.schunk-modular-robotics.com/, 07.01.2010
[2] F.So, N. Masahiro, E.Tadashi, "Collision Detection and Avoidance of Fingers for a Robot Hand", Kanagawa Univ., JPN, Journal: Proceedings of the Annual Conference of the Institute of Systems, Control and Information Engineers, 2005
[3] K.Matsumoto, S.Wakabayashi(NAL),I. Kawabuchi (TechExperts), T. Terashita(Hosei Univ.),"Space Robot Hand for the General Purpose On-orbit Task", National Aerospace Laboratory, Proceeding of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS 2001, Canadian Space Agency, St-Hubert, Quebec, Canada, June 18-22, 2001.
[4] Mark A.C. Gill and Albert Y. Zomaya, "A Parallel Collision-Avoidance Algorithm for Robot Manipulators", IEEE Concurrency, IEEE Computer Society, 1998, issn = 1092-3063, pages 68-78
[5] Lee, Sukhan (4913 Revlon Dr., La Canada, CA, 91011),"Distributed proximity sensor system having embedded light emitters and detectors", http://www.freepatentsonline.com/4893025.html, 1990
[6] U.Reiser, R.Volz, F.Geibel, "A flexible manipulation framework for collision avoidance and robot control", 39th International Symposium on Robotics, Seoul, Korea, 2008
[7] Stephen A. Ehmann and Ming C. Lin, "Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching",Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000
[8] S. Gottschalk, M. Lin and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. To Appear in Proc.of ACM Siggraph96,1996
[9] M. Lin, D. Manocha, J. Cohen, and S. Gottschalk. "Collision Detection: Algorithms and Applications", 1996.
[10] S. A. Ehmann and M. C. Lin.,SWIFT: Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching. Technical Report, Computer Science Department, University of North Carolina at Chapel Hill, 2000
[11] J.R.Quinlan, Induction of Decision Trees,Kluwer Academic Publishers, Machine Learning1: 81-106, 1986,
[12] K. Petersen, Efficient collision detection and realtime simulation of kinematics, dynamics and sensors of autonomous vehicles, TU Darmstadt, 2007
[13] R. Sambavaram, "Cylinder Collision", Engine pipe line programmer, Insomniac Games, 2007