2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Collision Avoidance with Potential Fields Based on Parallel Processing of 3D-Point Cloud Data on the GPU

Knut B. Kaldestad*, Sami Haddadin**, Rico Belder***, Geir Hovland*, David A. Anisi****

***Abstract*— In this paper we present an experimental study on real-time collision avoidance with potential fields that are based on 3D point cloud data and processed on the Graphics Processing Unit (GPU). The virtual forces from the potential fields serve two purposes. First, they are used for changing the reference trajectory. Second they are projected to and applied on torque control level for generating according nullspace behavior together with a Cartesian impedance main control loop. The GPU algorithm creates a map representation that is quickly accessible. In addition, outliers and the robot structure are efficiently removed from the data, and the resolution of the representation can be easily adjusted. Based on the 3D robot representation and the remaining 3D environment data, the virtual forces that are fed to the trajectory planning and torque controller are calculated. The algorithm is experimentally verified with a 7-Degree of Freedom (DoF) torque controlled KUKA/DLR Lightweight Robot for static and dynamic environmental conditions. To the authors knowledge, this is the first time that collision avoidance is demonstrated in real-time on a real robot using parallel GPU processing.**

I. INTRODUCTION & STATE OF THE ART

A. Problem Statement

Over the last decades robots carried out various autonomous operations in the real world and were certainly a game changer in automation as we know it today. They perform repetitive and laborious work that requires high precision and large payloads. However, the success of manipulating robots in the real-world was so far limited to preplanned tasks that require no online re-planning on trajectory nor task level. In particular, no technology was mature enough yet to let robots do quick and safe low-level decision making even on collision avoidance level. The recent trend of enabling robots to physically interact with humans in co-worker settings, however, enforces the need for viable solutions to the problem. Also the need to let larger industrial robots carry out more flexible tasks in other domains than manufacturing in rather uncertain and potentially changing environments increases the need even further. Both from a safety as well as task completion perspective, sensor based dynamic motion planning, for both in- and outdoor robots, could avoid causing damage in an unplanned event where an obstacle comes to block the original robot path.

*K. B. Kaldestad and G. Hovland are with Faculty of Technology and Science, Department of Engineering, University of Agder, Norway {knut.b.kaldestad,geir.hovland}@uia.no

**S. Haddadin is with Institute of Automatic Control, Leibniz University Hanover (LUH), Hanover, Germany sami.haddadin@irt.uni-hannover.de

***R. Belder is with Robotics and Mechatronics Center, DLR, Wessling, Germany rico.belder@dlr.de

****D. A. Anisi is with Dept. of Technology & Innovation, Div. of Process Automation, ABB, Norway david.anisi@no.abb.com

Fig. 1. *The KUKA/DLR Lightweight Robot equipped with a chuck-tool, which enables the robot to conduct different operations equipping e.g. an umbraco bit or a bore bit.*

B. State of the art

The state of the art is divided into three parts: applications, sensing and collision avoidance. The focus is on two significant application fields, namely robotic co-workers with light-weight robots, where the main concerns are high-performance physical human-robot interaction (pHRI) and safety, as well as new applications for larger industrial robots in harsh environments.

1) Applications:

The KUKA/DLR Lightweight Robot (see Figure 1) was initially developed by DLR [1] and has its roots in the space mission project ROTEX [2]. Later developments of the robot, after version III onwards, have been done in cooperation with the robot manufacturer KUKA [3]. Major design considerations regarding the 7-DoF robot was the intention to let the robot support and intuitively interact with humans in industrial manufacturing domains. There has been considerable studies with the LWR in the pHRI context (e.g. [4], [5]). In the second work e.g. the human hand is allowed to interact with the robot. A Kinect depth sensor is used to observe the scene, a hand gesture initializes interaction, and the hand is filtered out. The robot actively avoids all parts of the scene, which are still present in sensor data.

Other work presented in the field of collision avoidance is the reactive real-time motion generator by [4]. More sophisticated path planning algorithms need considerable time for path calculation. Reactive motion generators are typically subject to getting stuck in local minima. The reactive algorithm in [4] runs at the inner control loop at

1kHz, it can generate smooth motions, while maintaining desired velocity profiles and ensure smooth human contact through velocity profiling. The algorithm demonstrated its performance by experiments for both statical and dynamical obstacles. The robot was able to circumvent obstacles and reach the respective goal. Also when an external force was applied to the robot, such that it deviated from its original path, it converged to the goal when the external contact force was removed. In addition, the algorithm was validated by using different types of sensors, such as laser scanner and tracking system for keeping track of the wrist.

The circular fields methods use an analogy to electromagnetic fields and was recently extended in [6]. In contrast to potential fields, circular fields associate magnetic fields to environmental obstacles and take into account the robot velocity as well. However, at the same time circular fields methods require some additional knowledge of the environment, such as surface normals. Important to notice is that the algorithm is not prone to local minima, which is demonstrated in [6] by simulating different well-known trap scenarios. Furthermore, the paper demonstrates dynamical obstacle avoidance with complex 3D geometry.

Fig. 2. *A robot located in a potentially explosive environment, among pipes carrying gas. This shows a state of the art petrochemical related application for industrial robots.*

Industrial robots are exposed to more demanding environments than ever before (see Figure 2), performing on-site process inspection and manipulation while being remotely operated. Within the oil and gas industry, the strict regulations these robots have to comply with, such as ATEX [7] (French for "atmospheres explosibles") certified equipment and high reliability pose a real challenge when researching real-time collision avoidance. The work by [8] presents a robotic valve manipulation application and showcases the according demands. The robot is located outdoor at a running hydrocarbon process facility doing valve manipulation with sensor based online trajectory planning. Two other applications are presented in [9], the first is an indoor vision based valve manipulation with two collaborating robots. The first robot determines the orientation and the exact position of the valve, using an end-effector mounted network camera and a gradient based optimization algorithm. The second robot picks up the tool from a tool change holder and moves over to the valve and conducts the manipulation. The second presented application is a semi-automated scraper handling task, where a pipe is cleaned by sending the pressure driven scraper, from one location to the receiving destination. At the receiving end the robot opens the door to a de-pressurized chamber and locates the scraper by a proximity switch mounted at the tool. The scraper is extracted and the door is closed.

The literature referenced above demonstrates a few applications in an industry with strict regulations, where safety for humans and equipment is of highest importance. This shows that the oil and gas industry is starting to look at the opportunities that off-the-shelf industrial robots can provide. One of the challenges in this field is to best utilize already certified equipment and introduce new solutions to improve operations and safety.

2) Kinect, 3D Points, Graphics Card:

Since the depth camera Kinect was released in November 2010, a vast amount of research has been done in relation to the device, such as 3rd party drivers from OpenKinect and OpenNI, Microsoft's own SDK for Windows and libraries for image and point cloud processing such as Point Cloud Library (PCL) [10]. PCL version 1.6 has focus on CPU based algorithms, while algorithms for the GPU is under development. The work in [11] uses Kinect data and processes it on the GPU, for point registration purposes. One of the described advantages of the GPU utilization, is the transformation of the Kinect data to 3D points. Each pixel in the 640x480 RGB data is associated with a depth value. The transformation of each pixel to its corresponding 3D coordinate is highly suitable for parallel processing.

While there is significant research being done on the GPU, it has still not been a real alternative to the CPU in the majority of robotics applications. The reason is that the calculations need to be parallel and of a certain size before the GPU will outperform the CPU. Another factor has been development time and increased complexity, when comparing the complexity of C or C++ code for a CPU with CUDA C code for a GPU.

3) Collision avoidance:

Robot collision avoidance could be described as a set of instructions sent to the robot such that it avoids unintended interaction with objects while moving to its desired position. Such instructions could be generated from models of the environment [12] or sensor data [13] (or a combination of both). The pioneering work in real-time robot manipulator collision avoidance and potential fields started with [14] and [15]. The principle of the method is that a distance dependent repulsive force is generated as a function of the distance that is either fed as control input or modifies the path of a stable desirably attractive dynamical system, this in turn, generates a suitable reference trajectory. For obstacle avoidance the force will be repulsive, and is e.g. formulated as a polynomial function. This typical, very broad class of functions could give application desired characteristics such as the force increasing polynomially the closer the manipulator gets to the obstacle. Even though the field of obstacle avoidance is

A. Notation Used

TABLE I

For ease of use our notation is summarised below.

Symbol	Explanation
$\boldsymbol{q}$	Robot joint angle vector
$\boldsymbol{\tau}_d$	Desired control input
$\boldsymbol{x}$	Real end-effector positon
$\boldsymbol{x}_d$	Desired end-effector position
$\tilde{\boldsymbol{x}}$	Difference between real and desired positions
D_x	Cartesian damping matrix
K_x	Cartesian stiffness matrix
$\boldsymbol{g}(\boldsymbol{q})$	Gravity compensation torque in joints
$J(\boldsymbol{q})$	Robot Jacobian
$\mathcal{N}(\boldsymbol{q})$	Nullspace projector
$\boldsymbol{\tau}_v$	Virtual torques
$\boldsymbol{f}_n, \boldsymbol{m}_n$	Link force/moment vector
$F(\boldsymbol{d})$	Reactive force
$\mathcal{F}_{v,n}$	Virtual forces/moments
$T_l(\boldsymbol{q})$	Transformation matrices, one for each link: robot base to link frame as a function of $\boldsymbol{q}$
$\boldsymbol{p}_{in}$	Depth data from Kinect sensor
$\boldsymbol{r}_{in}$	Robot vertices
$\boldsymbol{r}_j$	Transformed vertices in robot model
d	Distance between robot and env. points
d_c	Distance from an environment point to the center of rotation for a robot link
r_{max}	Threshold force distance
s_l	Voxel side length
v_s	Voxel size
x_0, y_0, z_0	Offsets applied to measured points
x_w, y_h, z_d	Edge lengths of camera volume

not new, it is still being researched heavily today. At the time of writing, a viable solution for collision avoidance in this field has not been demonstrated. Some of the more recent work on improving potential field like methods is the significant extension in [6] of the original circular fields approach developed in [16].

The work presented in this paper is related to [17], however in this paper the robot is represented as a vertex model as opposed to spheres. As such, this gives a more realistic representation of the robot. The volume map is represented as voxels with a user defined resolution. The calculations are performed using both the GPU and CPU, which is one of the main contributions in this paper, in contrast to previous work focusing only CPU implementation.

II. ALGORITHM

B. Overview of Approach

In our framework, collision avoidance behavior is incorporated on two levels:

1) on trajectory deformation level, i.e. $\boldsymbol{x}_d(t)$ responds to the virtual forces
2) on torque control level, i.e. a control input $\boldsymbol{\tau}_v$ causes causes avoidance joint torques

In order for the first level to respond to virtual forces, the trajectory generation needs to establish a dynamical system with physical motivation (essentially a virtual impedance behavior) such as the one presented in [4]. The virtual dynamics generate a reference trajectory that responds to disturbance wrenches on operational task level and is then fed to a Cartesian impedance controller. Clearly, this scheme can only ensure end-effector collision retraction/avoidance. However, for kinematically redundant manipulators such as the LWR, this does not cover appropriate nullspace reactions. Therefore, a nullspace collision avoidance controller $\boldsymbol{\tau}_v$ is designed to implement according behavior. For this, the virtual forces $\mathcal{F}_{v,n}$ that act on each link n are projected via the respective sub-Jacobians J_n to joint space, followed by a suitable nullspace projector $\mathcal{N}(\boldsymbol{q})$. This ensures that $\boldsymbol{\tau}_v$ does not interfere with the primary impedance task. The overall controller can be written as

$$\boldsymbol{\tau}_d = J^T(K_x\tilde{\boldsymbol{x}} + D_x\dot{\tilde{\boldsymbol{x}}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_v \quad (1)$$

$$= J^T(K_x\tilde{\boldsymbol{x}} + D_x\dot{\tilde{\boldsymbol{x}}}) + \boldsymbol{g}(\boldsymbol{q}) + \mathcal{N}(\boldsymbol{q})\sum_{n=1}^{N} J_n^T\mathcal{F}_{v,n}, \quad (2)$$

where the notation is consistent with Table I. The virtual disturbance wrenches $\mathcal{F}_{v,n}$ for each link n are typically generated from sensory data and/or geometric knowledge from the environment. Environment 3D data is gathered using a Microsoft Kinect. Applying Algorithm 1 with 3D environmental point cloud data and a 3D vertex model of the robot, the set of virtual forces/moments $\mathcal{F}_{v,n}$ is generated as explained next.

C. Calculation of reactive forces

The calculation of the reactive forces are well suited for parallel processing, because the distance between each point represented by the robot model and each point in the environment can be calculated separately. The distance $\boldsymbol{d}$ would then be used, such that the force is a function of the distance, $F(\boldsymbol{d})$. The force function could take many forms, where as in our case, we use a second order polynomial function. To avoid any forces from objects located at a predefined distance farther from the robot, a threshold force distance r_{max} is set such that the robot only reacts to the objects located at $\|d\| \leq r_{max}$. These forces are then summed to create a force vector and a moment vector for each link and the end effector.

$$\boldsymbol{f}_n = \sum F(\boldsymbol{d}) \quad (3)$$

$$\boldsymbol{m}_n = \sum F(\boldsymbol{d}) \times \boldsymbol{d}_c \quad (4)$$

where $\boldsymbol{f}_n$ and $\boldsymbol{m}_n$ are the forces and moments for each respective link $\boldsymbol{n}$, and $\boldsymbol{d}_c$ is the distance from the environment point to the center of rotation for the robot link.

D. Voxel map creation

When creating the voxel map, a small footprint and a parallel scheme of the data is important. To reduce the size of the map, a simple compression method is proposed, which is highly parallel and easily deployable on the GPU.

A regular voxel map for 3D Cartesian space representation could be represented by points of type $\boldsymbol{P} = (x\ y\ z)^T$, and demonstrate its presence in a 3D array by

$$Map_3(x, y, z) = 1 \quad (5)$$

Algorithm 1: Reactive Force/Moment Calculation

Input Map:
Depth data $\boldsymbol{p}_{in}$
Voxel size v_s
Bounds x_0, y_0, z_0, x_w, y_h, z_d
Neighbours n
Input Robot:
Vertices $\boldsymbol{r}_{in}$ for all n links
Joint angles $\boldsymbol{q}$
Transformation matrices $T_l(\boldsymbol{q})$
Input Potential field:
Pointer to force function $F(\boldsymbol{d})$
Threshold force distance r_{max}

Voxel insertion:
for *each point* $\boldsymbol{p}$ *in* $\boldsymbol{p}_{in}$ *within bounds* **do**
  if $\boldsymbol{p}$ *within VoxelMap* **then**
    VoxelMap $\leftarrow \boldsymbol{p}$
  end
end

Remove robot from VoxelMap:
for *each vertex* $\boldsymbol{r}$ *in* $\boldsymbol{r}_{in}$ **do**
  $\boldsymbol{r}_j \leftarrow T_l(\boldsymbol{q}) \times \boldsymbol{r}$
  if *VoxelMap contains* $\boldsymbol{r}$ *in* $\boldsymbol{r}_j$ *as voxel* **then**
    Remove $\boldsymbol{r}_j$ from VoxelMap
  end
end

Remove Outliers:
for *each voxel* $\boldsymbol{v}$ *in VoxelMap* **do**
  if $\boldsymbol{v}$ *has less than n neighbours* **then**
    Remove $\boldsymbol{v}$ from VoxelMap
  end
end

Calculate forces and moments:
for *each link* n **do**
  $\hat{\boldsymbol{f}} \leftarrow \boldsymbol{0},\ \hat{\boldsymbol{m}} \leftarrow \boldsymbol{0}$
  for *each voxel center* $\boldsymbol{v}_c$ *in VoxelMap* **do**
    for *each* $\boldsymbol{r}$ *in* $\boldsymbol{r}_j$ *belonging to link* n **do**
      $\boldsymbol{d} \leftarrow \boldsymbol{r} - \boldsymbol{v}_c$
      if $\|\boldsymbol{d}\| < r_{max}$ **then**
        $\hat{\boldsymbol{f}} \leftarrow \hat{\boldsymbol{f}} + F(\boldsymbol{d})$
        $\hat{\boldsymbol{m}} \leftarrow \hat{\boldsymbol{m}} + F(\boldsymbol{d}) \times \boldsymbol{d}_c$
      end
    end
  end
  $\mathcal{F}_{v,n} \leftarrow \hat{\boldsymbol{f}}, \hat{\boldsymbol{m}}$
end

return $\mathcal{F}_{v,n}$

If the map is limited e.g. to 5 m x 5 m x 5 m with a resolution of 5 mm, the map would consist of 10^9 elements, or 1 gigabyte of memory allocated in an uncompressed state. Such an approach is clearly not space efficient. Methods such as *kd-trees*[18] could now be used to reduce the memory footprint. There, the 3D points are structured in a 3-dimensional tree which allows for faster search. The time to generate the tree and calculate the k nearest neighbors (kNN) is, however, not fast enough for our demands. The work in [19] provides, in addition to their own algorithm GPU-FS-kNN, a good overview of different kNN algorithms for both CPU and GPU. Due to the current speed limitations of these approaches, we chose to follow a different algorithmic path:

If the 3D-Sensor is located in a fixed position and orientation, the only input to the map is the (x, y, z)-location of the according measurements. If the z-coordinate changes, the map needs to be updated (we can not get multiple depth values for the same pixel) due to the fact that originally the data is $2\frac{1}{2}$D. Because of this limitation, the voxel map can be represented as a 2D array, where its implicit structure may be written as

$$Map_{2.5}(x, y) = z. \tag{6}$$

This representation is in fact very similar to the original one, except for the fact that it has now a clear structure and it is possible to index the points directly. This is done by creating the map with seven parameters: Offsets x_0, y_0, z_0, dimensions x_w, y_h and z_d and voxel size v_s. This results in the following relationship between sensor points $P_k = (x_k\ y_k\ z_k)$ and $Map_{2.5}(x_i\ y_i)$.

$$\hat{z} = z_k \tag{7}$$

$$\hat{x}_i = \text{floor}\left(\frac{x_k - x_0}{v_s}\right) \tag{8}$$

$$\hat{y}_i = \text{floor}\left(\frac{y_k - y_0}{v_s}\right) \tag{9}$$

From eqs. (8) and (9) it should be fairly straightforward to see that large values for v_S will lower the total resolution. The consequence is that not all points P_k will appear in the $Map_{2.5}$-representation. Averaging the depth value of the points could result in hallucinated distances in open space, e.g. the mean distance between an object that is close to the sensor, and an object farther away. If the location of the Kinect is chosen such that it is close to the volume that it shall observe, a reasonable choice is to insert the point with the lowest z-value, into the $Map_{2.5}$. If the depth camera were to provide a resolution of 1 mm, a worst case with a voxel side length of s_l would lead to a loss of $m = s_l^2 - 1$ points.

This loss is not of great importance as long as s_l is kept reasonably low compared to the size of the object surface area.

Some of the benefits of the $Map_{2.5}$ structure are:

- Omission of the z-dimension, for 3 m depth with resolution 0.01 m which results in a 300 times more compact structure than Map_3.

- Structured in a way, which enables fast localization and removal on the GPU. Example: The robot could be removed from the map without using any search algorithm. The robot is simply inserted into a separate voxel map, created with the same parameters as the environment. Then in parallel for all points, remove the points where the robot overlaps with the environment.
- No need to search for neighbors which can be indexed directly. Example: To find N neighbors for all points, located within radius r from point p, it is only necessary to check exactly those voxels within radius r of the point. This can be done for each point in parallel.

Well known problems with the depth data from the Kinect are noise and presence of outliers [20]. Outliers could contribute to residual motion of the robot if it is located within the threshold range of the link. This could severely affect the potential field force acting on the robot, if it is located close to the link. An outlier could potentially increase the risk of collisions, jeopardizing the entire purpose of collision avoidance. A simple and fast algorithm for outlier removal for the GPU is therefore proposed in algorithm 1.

Finally, the concrete robot (LWR) is represented by seven vertex models, one for each link, base and end-effector. The robot has two representations in the algorithm, the first is a voxel map ($voxel_{rob}$) used for robot removal. The second is an unstructured vertex model (see Figure 3) used for calculating the forces. $voxel_{rob}$ is created with the same parameters as the environment map ($voxel_{env}$). For each iteration, a homogeneous transformation accordingly to the robot's (sub) forward kinematics are applied to each of the vertex models of the robot and inserted into a new map $voxel_{rob}$. The robot is then removed from the environment. Each voxel in $voxel_{rob}$ that corresponds to a voxel in $voxel_{env}$ (plus a tolerated offset) is thus removed from $voxel_{env}$.

The unstructured vertex model of the robot, which gives its proper 3D representation, is used for calculating the forces generated from the potential field. For each vertex on every link in the robot vertex model, a distance is calculated to all environment voxels. Each of these distances contributes to its link with three forces and three moments. The forces and moments are then added for every link. In the end, three total forces and moments act on each joint, respectively.

III. EXPERIMENTS

A. System Setup

The experimental system consists of a KUKA LWR IV manipulator with 7-DoFs and the robot controller system *Beasty* [21] running at 1000 Hz. The computer for the calculation of algorithm 1 is equipped with an Intel Xeon 3.3 GHz CPU, 8GB of memory and a NVIDIA Geforce GTX 680 graphics card.

B. Experiment Design

The experiments are carried out in an industrial setting, partly shown in Figures 6 to 9. The robot is located in the environment center and the Kinect depth sensor is placed such that it captures the robot in a side view (please see accompanying video for details).

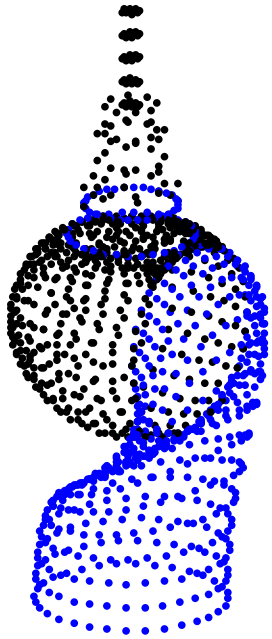

Fig. 3. *Vertices of the last arm segment, wrist, flange and tool.*

For both experiments the voxels had a resolution of 10 mm, while the volume covered by the depth camera is x_{width}=2000 mm, y_{height}=2000 mm and z_{depth}=1800 mm. The robot model consists of two sets of 7 vertex models, which contain 2468 and 4701 vertices in total.

1) Experiment 1, Static Objects: In the static environment, the robot first runs the path simply generated by eq. (2) if no collision forces modify the path. The depth data is checked for changes at the same rate as the data becomes available. First, a box is placed in the environment to block the robot path. Different types of objects are then placed on top of the box, thus changing the environment. The robot actively avoids every object that partly blocks its free movement volume.

2) Experiment 2, Dynamic Objects: In this experiment the robot end-effector is set to reach a goal location. A human worker then enters the environment during the robot movement, causing the manipulator to deviate from its nominal path.

IV. RESULTS

The experimental evaluation indicates the performance of the algorithms for both experimental setups. The robot shows whole-arm collision avoidance, while still being able to reach its final goal position if the according volume is clear. The time-stamps in the screenshots (Figure 6 – 9) are all relative, the entire experiment was done in one shot.

A. Static Obstacles

Figure 6 depicts the desired path if no obstacles obstruct the motion. The robot intends to move along this path for

the entire experiment. While being in motion, a blue box is placed in the path of the robot, see Figure 7. As can be seen from the figure, the robot actively avoided the box. To make it more difficult for the robot to reach its goal, the white obstacle is placed on top of the box, see Figure 8. The white obstacle is moved to another position further away from the robot. The robot responds by avoiding the obstacle on the right side (not shown in the figures).

B. Dynamic Obstacles

The second phase of the experiment includes a dynamic obstacle, a person enters the workspace while the robot is in motion. As one can see, the robot is able to quickly avoid the human and prevent the collision. In the accompanying video, the dynamic response can be seen more clearly and it is shown that the robot converges to the goal again as soon as the human leaves the workspace.

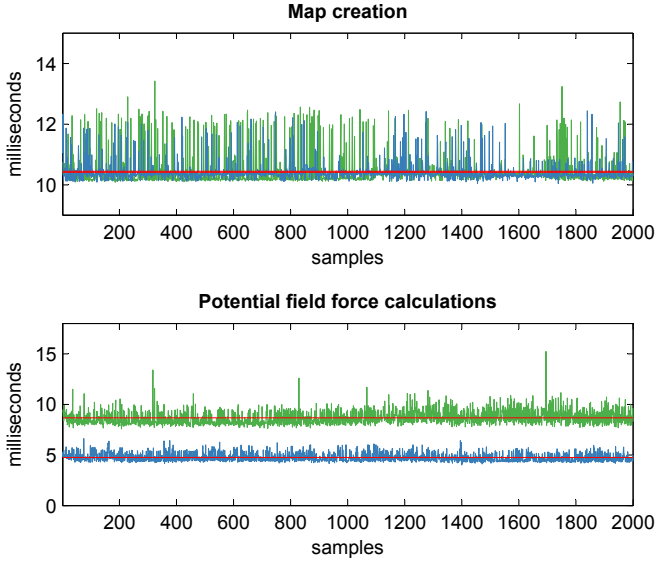

Fig. 4. *Algorithm performance, calculation time on the vertical axis in milliseconds for 2000 samples. The robot model consists of 2468 vertices. The green graph shows an environment voxel resolution of 5 mm, while the blue graph shows a voxel resolution of 10 mm. The red lines depicts the average measurement value.*

C. Calculation Time

Since the map is recreated each time new sensor data becomes available, it is possible to calculate the potential forces in the mean time. Even though the environment update rate is restricted by the Kinect, the robot state can be retrieved at a rate of 1 kHz. In parallel to waiting for new depth data, the algorithm then calculates the current potential field forces based on the updated robot position. The potential field force calculations continuously receive new robot joint angles, transforming the robot vertices accordingly, and calculating the respective potential field forces.

Figure 4 depicts the calculation time for 2468 robot vertices with 10 mm and 5 mm resolution. The average time for the full map creation is 10.41 ms and 10.44 ms, respectively.

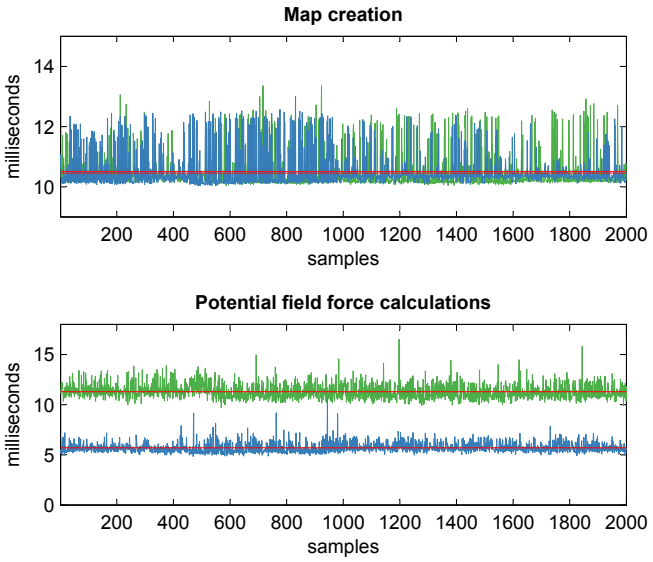

Fig. 5. *Algorithm performance, calculation time on the vertical axis in milliseconds for 2000 samples. The robot model consists of 4701 vertices. The green graph shows an environment voxel resolution of 5 mm, while the blue graph shows a voxel resolution of 10 mm. The red lines show the average measurement value.*

Architecture	Voxel map resolution	Robot vertices	Algorithm calculation time	Performance factor
CPU	10 mm	4701	1.426 s	129.63x
GPU	10 mm	4701	0.011 s	
CPU	5 mm	4701	5.692 s	402.07x
GPU	5 mm	4701	0.014 s	
CPU	10 mm	2468	0.755 s	68.63x
GPU	10 mm	2468	0.011 s	
CPU	5 mm	2468	3.009 s	273.54x
GPU	5 mm	2468	0.011 s	

TABLE II

CPU vs GPU performance. In the performance calculation x_{width} and y_{height} are both 2000 mm, z_{depth} is 1800mm and r_{max} is 300 mm

The potential field calculations take on average 4.74 ms and 8.68 ms, respectively. Figure 5 is generated with a robot consisting of 4701 vertices, and for two different voxel resolutions of 10 mm and 5 mm, respectively. The average time for the map creation was 10.45 ms and 10.51 ms, while the potential field calculation takes 5.70 ms and 11.29 ms, respectively.

D. Comparison GPU and CPU speeds

The results in Table IV-D show a significantly lower calculation time for the GPU algorithm. The performance increase varies from a factor of 68.63 to 402.07 depending on voxel map resolution and number of robot vertices. The significantly better performance on the GPU is the enabling factor allowing real-time collision avoidance.

V. CONCLUSION

In this paper, a study on GPU based collision avoidance with Potential Fields is presented, using 3D point cloud data converted to voxels as environmental representation. The virtual forces that are fed to the trajectory planning and torque control level are calculated in real-time. In fact, the proposed algorithm may even run significantly faster than the sensor frame rate. The experimental performance of the scheme showed good results with a 7-DoF KUKA/DLR Lightweight robot for various static and dynamic environmental conditions. In particular, the combination of trajectory deformation based on virtual dynamics that are affected by the virtual forces on Operational space level, together with the projection of the forces into the nullspace of the Cartesian impedance controller led to convincing whole body real-time collision avoidance responses. To the authors knowledge, this paper presents for the first time real-time collision avoidance using a real robot and parallel GPU processing.

ACKNOWLEDGMENTS

This work was performed while the first author was visiting DLR, whose hosting is gratefully acknowledged. This work has been partially funded by the European Commission's Sixth Framework Programme as part of the project SAPHARI under grant no. 287513. The work is also funded by ABB and the Norwegian Research Council through project number 193411/S60.

REFERENCES

[1] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, "The DLR lightweight robot - lightweight design and soft robotics control concepts for robots in human environments," *Industrial Robot Journal*, vol. 34, no. 5, pp. 376–385, 2007.

[2] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "ROTEX-the first remotely controlled robot in space," in *Robotics and Automation, 1994. Proc. IEEE Intl. Conf.*, May, pp. 2604–2611 vol.3.

[3] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, "The KUKA-DLR Lightweight Robot arm - a new reference platform for robotics research and manufacturing," in *Robotics (ISR), 2010 41st Intl. Symp. on and 2010 6th German Conf. on Robotics (ROBOTIK)*, June, pp. 1–8.

[4] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2010.

[5] A. De Luca and F. Flacco, "Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration," in *Biomedical Robotics and Biomechatronics (BioRob), 2012 4th IEEE RAS EMBS Intl. Conf.*, June, pp. 288–295.

[6] S. Haddadin, R. Belder, and A. Albu-Schäffer, "Dynamic motion planning for robots in partially unknown environments," in *IFAC World Congress (IFAC2011)*, Milano, Italy, September 2011.

[7] P. Leroux, "New regulations and rules for atex directives," *Industry Applications Magazine, IEEE*, vol. 13, no. 1, pp. 43–51, 2007.

[8] D. Anisi, E. Persson, C. Heyer, and C. Skourup, "Real-World Demonstration of Sensor-Based Robotic Automation in Oil & Gas Facilities," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, California, Sep. 2011.

[9] D. Anisi, J. Gunnar, T. Lillehagen, and C. Skourup, "Robot Automation in Oil and Gas Facilities: Indoor and Onsite Demonstrations," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010, pp. 4729–4734.

[10] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[11] D. Neumann, F. Lugauer, S. Bauer, J. Wasza, and J. Hornegger, "Real-time RGB-D Mapping and 3-D Modeling on the GPU using the Random Ball Cover Data Structure," in *IEEE Intl. Conf. on Computer Vision (ICCV) Workshops*, A. Fossati, J. Gall, H. Grabner, X. Ren, and K. Konolige, Eds., 2011, pp. 1161–1167. [Online]. Available: http://www5.informatik.uni-erlangen.de/Forschung/Publikationen/2011/Neumann11-RRM.pdf

[12] D. Henrich, C. Wurll, and H. Wörn, "6 dof path planning in dynamic environments-a parallel online approach," in *Robotics and Automation, Proc. IEEE Intl. Conf.*, vol. 1, May 1998, pp. 330 –335 vol.1.

[13] J. Borenstein, Y. Koren, and S. Member, "The vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation*, vol. 7, pp. 278–288, 1991.

[14] O. Khatib, "Real-time control of manipulators in operational space," in *Proc. of the 28th Annual Stanford Conf. American Society for Quality Control*, Palo Alto, CA, USA, October 1984, pp. 9/1–9/7.

[15] ——, "Real-time obstacle avoidance for manipulators and mobile robots," *The Intl. Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, Spring 1986.

[16] L. Singh, H. Stephanou, and J. Wen, "Real-time robot motion control with circulatory fields," in *Robotics and Automation, 1996. Proc., 1996 IEEE Intl. Conf. on*, vol. 3, 1996, pp. 2737–2742 vol.3.

[17] F. Flacco, T. Kroger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Robotics and Automation (ICRA), 2012 IEEE Intl. Conf.*, May 2012, pp. 338 –345.

[18] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975. [Online]. Available: http://doi.acm.org/10.1145/361002.361007

[19] A. Arefin, C. Riveros, R. Berretta, and P. Moscato, "GPU-FS-kNN: A Software Tool for Fast and Scalable kNN Computation Using GPUs." *PLoS One*, vol. 7, no. 8, p. e44000, 2012.

[20] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/2/1437

[21] S. Parusel, S. Haddadin, and A. Albu-Schäffer, "Modular state-based behavior control for safe human-robot interaction: A lightweight control architecture for a lightweight robot," in *Robotics and Automation (ICRA), 2011 IEEE Intl. Conf. on*, May, pp. 4298–4305.

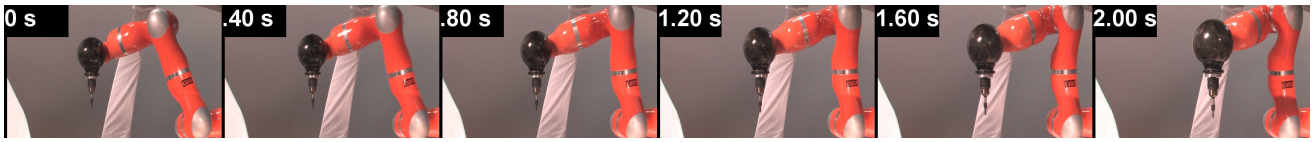

Fig. 6. *No obstacle: Showing the original robot path.*

Fig. 7. *Static obstacle: A box blocks the robot path.*

Fig. 8. *Static obstacle: Another object is placed in front of the robot, to make the free movement volume even smaller.*

Fig. 9. *Dynamic obstacle: A human worker enters the work area and the robot actively avoids him.*