2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Depth Space pproach to Human-Robot Collision voidance

Fabrizio Flacco Torsten Kröger lessandro De Luca Oussama Khatib

***bstract*— In this paper a real-time collision avoidance approach is presented for safe human-robot coexistence. The main contribution is a fast method to evaluate distances between the robot and possibly moving obstacles (including humans), based on the concept of depth space. The distances are used to generate repulsive vectors that are used to control the robot while executing a generic motion task. The repulsive vectors can also take advantage of an estimation of the obstacle velocity. In order to preserve the execution of a Cartesian task with a redundant manipulator, a simple collision avoidance algorithm has been implemented where different reaction behaviors are set up for the end-effector and for other control points along the robot structure. The complete collision avoidance framework, from perception of the environment to joint-level robot control, is presented for a 7-dof KUK Light-Weight-Robot IV using the Microsoft Kinect sensor. Experimental results are reported for dynamic environments with obstacles and a human.**

I. Introduction

flexible, reactive, and safety-oriented control of physical interaction between humans and robots allows a closer cooperation in service and industrial tasks that require the adaptability skills of humans to be merged with the high performance in terms of precision, speed and payload of robots [1]. The avoidance and safer handling of collisions are basic components of this challenge. While potential injuries of unexpected human-robot impacts can be limited by lightweight/compliant mechanical design of manipulators [2] and collision detection/reaction strategies [3], preventing collisions in a dynamic and largely unpredictable environment relies on the extensive use of exteroceptive sensors.

real-time collision avoidance method is composed essentially by three parts: (1) Perception of the environment; (2) Collision avoidance algorithm; (3) Robot control. For its importance collision avoidance has been one of the most studied field in robotics, and many different planning and control approaches for obstacle avoidance have been proposed. large majority of the real-time capable planning concepts are based on the famous potential field approach introduced in [4] and further elaborated, e.g., in [5], [6]. Virtual repulsive and attractive fields are associated respectively to obstacles and target, such that a motion towards the goal is achieved while obstacles are avoided. Real-time adaptive motion planning methods [7]–[9] are key to give reactive motion control behaviors to robotic systems. These works use parametrized collision-free paths (e.g, splines) to represent calculated trajectories, and update the trajectory parameters at runtime as the environment changes or is discovered by the robot sensors. n on-line motion planning approach where paths and trajectories are calculated on line in the *configuration* $\times$ *time* space, so that the robot can act in unknown dynamic environments has been proposed in [10]. Collision-free vertices ("milestones") and edges on a road map, which is another kind of representation of currently planned trajectories, are used in [11], [12]. In [13] both the robot and the human are represented by a number of spheres and a collision-free trajectory is obtained by exploring possible end-effector movements in predefined directions. combination of potential and circular fields, which is suitable for complex environments and provides good convergence properties to the goal, has been recently proposed in [14]. Finally, the concept proposed in [15] uses virtual springs and damping elements to be used as input values for a Cartesian impedance controller that will generate the motion trajectories.

Most of the above works assume that the information about the environment needed to avoid obstacles is already available, skipping the perception part. s a common characteristic, collision avoidance algorithms are based at least on a measure of the distances between the robot and the obstacles. The idea of computing this distance directly from an image of the environment was introduced in [16]. The minimum robot-obstacle distance is obtained by expanding the convex hull associated to the robot until the image associated to an obstacle is reached. Since this method use only on a 2D image, knowledge of the vector between the two points of minimum distance is not available. The distance information alone is useful just to slow down or to stop the robot motion for collision prevention.

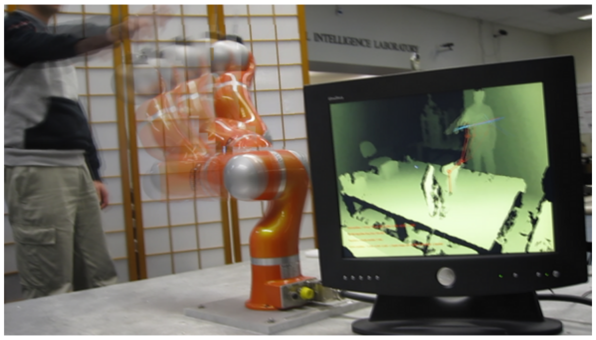

Fig. 1. robot arm reacts instantaneously to motions of humans and other dynamic obstacles that are detected in depth space, such that collisions are avoided

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via riosto 25, 00185 Rome, Italy {fflacco,deluca}@dis.uniroma1.it. rtificial Intelligence Laboratory, Stanford University, Stanford, C 94305, US {tkr, khatib}@stanford.edu.

Nowadays, visual sensing is one of the best choices for integrating sensor-based collision avoidance concepts in motion control system. Moreover, the development of new low-cost depth sensors such as the Microsoft Kinect$^{\mathrm{TM}}$ [17] allows to meet many requirement with a very cheap and powerful sensor system.. The classical way to use depth data is to project them into a robot-oriented space, reassemble representations of obstacles in this space, and finally compute the information needed for collision avoidance. Examples of this approach are [18], where Cartesian space control is used, and [19], in which obstacles are represented in the configuration space.

With reference to the setup shown in Fig. 1, with a lightweight KUK LWR IV robot [20] sharing its workspace with a human and the associated depth image of a monitoring Kinect camera, in this paper a new fast approach is proposed that computes distances between robot and workspace obstacles directly from depth data. This mimics the human behavior for obstacle avoidance where, at least at the reflex level, only visual feedback is used for a rough estimation of the relative distances between the obstacles and ourselves. The robot to obstacles distances are then used in a simple variant of a classical potential field method, so as to generate repulsive commands for the robot to avoid collisions.

The paper is organized as follows. The concept of depth space is summarized in Sect. II. Section III introduces our new approach to estimate robot-obstacle distances based on depth space computations. In Sect. IV, the obtained distances are used to generate a repulsive vector from obstacle(s) to a point of interest on the robot. The basic repulsion concept can be improved by considering also estimated velocities of the obstacles (Sect. IV-) and by the use of multiple points of interest (*control points*) along the manipulator (Sect. IV-B). Section V presents the general framework of our robot motion controller, where the collision avoidance scheme has been integrated. The laboratory setup and the obtained experimental results are described in Sec. VI, using a 7-dof KUK LWR IV.

II. Depth Sp ce

The depth space is a non-homogeneous $2\frac{1}{2}$-dimensional space, where two elements represent the coordinate of the projection of a Cartesian point on a plane, and the third element represents the distance between the point and the plane. The depth space of an environment can be captured by a depth sensor (e.g., a stereo, time of flight, or structured light camera), which is modeled as a classic pin-hole camera. The pin-hole camera model is composed by two sets of parameters, the intrinsic parameters in matrix $\mathcal{K}$, which model the projection of a Cartesian point on the image plane, and the extrinsic parameters in matrix $\mathcal{E}$, which represent the coordinate transformation between the reference and the sensor frame, i.e.,

$$\mathcal{K} = \begin{pmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathcal{E} = \quad \boldsymbol{R} \;\mid\; \boldsymbol{t} \,\big), \quad (1)$$

where f is the focal length of the camera, s_x and s_y are the dimensions of a pixel in meters, c_x and c_y are the pixel coordinates of the center (on the focal axis) of the image plane, and $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation and translation between the camera and the reference frame. Each pixel of a depth image contains the depth of the observed point, namely the distance between the Cartesian point and the camera image plane. Note that only the depth of the closest point along a given ray is stored; all occluded points that are beyond compose, for all camera rays, a region of uncertainty called *gray area*, see the example in Fig. 2.

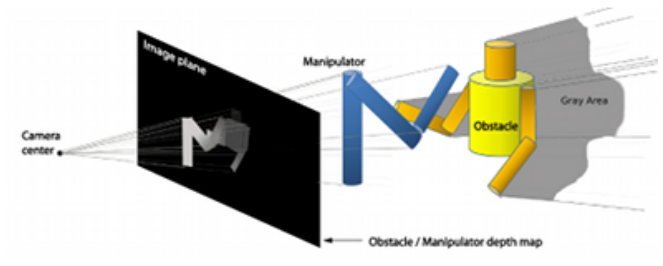

Fig. 2. Generation of a depth image, with lighter intensities representing closer objects. Points occluded by the obstacle compose the gray area in the Cartesian space. The manipulator does not contribute to the gray area, because it is removed from the image as explained in Sec. IV-B

Consider a generic Cartesian point expressed in the reference frame as $\boldsymbol{P}_R = \quad x_R \quad y_R \quad z_R \,\big)^T$. Its expression in the sensor frame is

$$\boldsymbol{P}_C = \quad x_C \quad y_C \quad z_C \,\big)^T = \boldsymbol{R}\boldsymbol{P}_R + \boldsymbol{t}, \qquad (2)$$

and its projection $\boldsymbol{P}_D = \quad p_x \quad p_y \quad d_p \,\big)^T$ in the depth space is given by

$$\begin{aligned} p_x &= \frac{x_C f s_x}{z_C} + c_x \\ p_y &= \frac{y_C f s_y}{z_C} + c_y \\ d_p &= z_C, \end{aligned} \qquad (3)$$

where p_x and p_y are the pixel coordinates in the image plane and d_p is the depth of the point.

III. Dist nce Ev lu tion

The distance between the robot and an obstacle is the essential information needed for obstacle avoidance. Consider an obstacle point   and its depth space representation $\;_D =$ $\quad o_x \quad o_y \quad d_o \,\big)^T$ captured by the camera. To evaluate an useful Cartesian distance between the obstacle point   and a point of interest $\boldsymbol{P}$, also represented in the depth space as $\boldsymbol{P}_D = \quad p_x \quad p_y \quad d_p \,\big)^T$ by means of eqs. (2) and (3), two possible cases arise (see Fig. 3). If the obstacle point has a larger depth than the point of interest ($d_o > d_p$), then the distance is computed as

$$\begin{aligned} v_x &= \frac{(o_x \quad c_x)\, d_o \quad (p_x \quad c_x)\, d_p}{f s_x} \\ v_y &= \frac{(o_y \quad c_y)\, d_o \quad (p_y \quad c_y)\, d_p}{f s_y} \\ v_z &= d_o \quad d_p \\ \|\boldsymbol{D}(\boldsymbol{P}, \quad)\| &= \sqrt{v_x^2 + v_y^2 + v_z^2}\,, \end{aligned} \tag{4}$$

where $\boldsymbol{D}(\boldsymbol{P}, \quad) = \quad v_x \quad v_y \quad v_z \quad)^T$. Otherwise, the distance w.r.t. the occluded points is considered. For this, we assume the depth of the obstacle to be $d_o = d_p$ and the distance is then obtained from eq. (4). Note that the resulting value is not the actual Cartesian distance, but it contains enough information for collision avoidance. This distance evaluation is based on simple relations using only depth space data associated to the camera. Moreover, it properly takes into account also the gray area whereas with other methods the distance for occluded points would not be evaluated in an useful way.

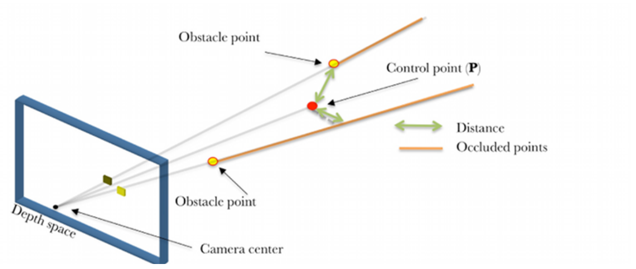

Fig. 3. Depth space distance evaluation to a point of interest $\boldsymbol{P}$: The two possible cases of obstacle depth larger or not than the depth of the point of interest are shown

Next, we would like to evaluate the distances between $\boldsymbol{P}$ and all obstacles sufficiently close to it. Consider a Cartesian *region of surveillance*, constituted by a cube of side 2ρ centered at $\boldsymbol{P}$, where the presence of obstacles must be detected. The associated region of surveillance in the image plane has dimensions

$$x_s = \rho\, \frac{f s_x}{d_p \quad \rho}, \qquad y_s = \rho\, \frac{f s_y}{d_p \quad \rho}. \tag{5}$$

Therefore, all pixels in the image plane within the region of surveillance $\boldsymbol{S} = \left[p_x \quad \frac{x_s}{2}, p_x + \frac{x_s}{2}\right] \times \left[p_y \quad \frac{y_s}{2}, p_y + \frac{y_s}{2}\right]$ must be considered. The distance evaluation for each obstacle pixel is completely independent, thus distances can be computed concurrently speeding up the method. Moreover, if only the minimum distance is required, the number of distance evaluations can be reduced by considering pixels that are closer to (p_x, p_y) first. s soon as a new local minimum $D_{\min}(\boldsymbol{P}) = \min \quad _{\in \boldsymbol{S}} \|\boldsymbol{D}(\boldsymbol{P}, \quad)\| < \rho$ is found among the pixels in the already explored area $\boldsymbol{S}' \subset \boldsymbol{S}$, the region of surveillance can be shrunk by setting $\rho = D_{\min}$ and using again eq. (5). Finally, distance computation is applied only to pixels whose depth is compatible with the workspace of the robot manipulator, so that points too far or too near are rejected.

IV. REPULSIVE CTION

Once the robot-obstacle distances have been evaluated, they are used to modify on-line the current trajectory of the manipulator so as to avoid collision. Many different approaches for obstacle avoidance have been proposed (see Sect. I). We present here a simple but effective method based on the generation of repulsive vectors in Cartesian space, which can be used as input of any preferred collision avoidance algorithm.

ssociated to the distance vector from the obstacle to the point of interest $\boldsymbol{P}$ obtained from eq. (4), a repulsive vector is defined as

$$\boldsymbol{V}_C\,(\boldsymbol{P}, \quad) = v\,(\boldsymbol{P}, \quad)\, \frac{\boldsymbol{D}(\boldsymbol{P}, \quad)}{\|\boldsymbol{D}(\boldsymbol{P}, \quad)\|}, \tag{6}$$

i.e., having the same direction of $\boldsymbol{D}(\boldsymbol{P}, \quad)$ but with magnitude

$$v\,(\boldsymbol{P}, \quad) = \frac{V_{\max}}{1 + e^{(\|\boldsymbol{D}(\boldsymbol{P}, \quad)\|(2/\rho) \quad 1)}}, \tag{7}$$

where $V_{\max}$ is the maximum admissible magnitude and is a shape factor. The magnitude v of the repulsive vector will be V_{max} if $\|\boldsymbol{D}(\boldsymbol{P}, \quad)\| = 0$, and will approach zero when the distance reaches ρ (beyond, $\boldsymbol{V}_C$ is not defined). n example profile is shown in Fig. 4.

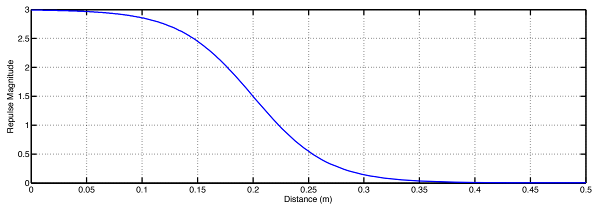

Fig. 4. Repulsive magnitude from eq. (7), with parameters $V_{m \quad x} \quad 3$ [m/s], $\rho \quad 0.4$ [m], and $\quad 6$

first possibility for obstacle avoidance would be to use the repulsive vector associated *only* to the obstacle point with the minimum distance

$$_{\min} = \text{argmin} \quad _{\in \boldsymbol{S}} \|\boldsymbol{D}(\boldsymbol{P}, \quad)\|, \tag{8}$$

namely

$$\boldsymbol{V}_{C_{\min}}(\boldsymbol{P}) = \boldsymbol{V}_C(\boldsymbol{P}, \quad _{\min}) = \max_{\in \boldsymbol{S}} \boldsymbol{V}_C(\boldsymbol{P}, \quad),$$

which is also the most common choice in the related literature. We propose instead to consider in a suitable way *all* obstacle points that lie inside the region of surveillance:

$$\begin{aligned} \boldsymbol{V}_{C_T}(\boldsymbol{P}) &= \textstyle\sum \quad _{\in \boldsymbol{S}} \boldsymbol{V}_C\,(\boldsymbol{P}, \quad) \\ \boldsymbol{V}_{C \quad ll}(\boldsymbol{P}) &= v\,(\boldsymbol{P}, \quad _{\min})\, \frac{\boldsymbol{V}_{C_T}(\boldsymbol{P})}{\|\boldsymbol{V}_{C_T}(\boldsymbol{P})\|}. \end{aligned} \tag{9}$$

In this way, all obstacle points contribute to the direction of the resulting repulsive vector, while the magnitude depends only on the minimum distance to all obstacle points. If all points were used to compute the magnitude, this would be influenced by the number of obstacle points, or if the mean value of the distances were used, it would still be affected by the ratio of near and far obstacles. These behaviors are not desirable, since the presence of a close obstacle with high risk of collision should provide always the same repulsive vector magnitude. The main benefit of using all points are that *i)* the repulsive vector is less sensible to noise of the depth sensor, producing a smoother variation of the vector direction, and *ii)* the presence of multiple obstacles is handled in a better way, as shown in Fig. 5.

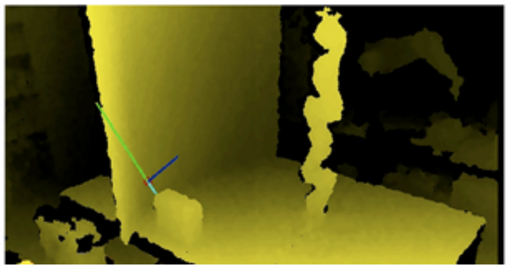

Fig. 5. Example of repulsive vector computation. Lighter colors refer to obstacle points with smaller depth, the point of interest $\boldsymbol{P}$ is represented by a red circle, and the minimum distance is represented in cyan. The repulsive vector obtained by using the minimum distance only is shown in green, while the one obtained by using all points in the range of surveillance is in blue. It can be seen that the green repulsive vector points to another obstacle (dangerous), while the blue vector points to a free area (safer)

ll above repulsive vectors are expressed in the camera frame, but can be transformed in the reference frame as $\boldsymbol{V}_R(\boldsymbol{P}) = \boldsymbol{R}^T\boldsymbol{V}_C(\boldsymbol{P})$. In the following, $\boldsymbol{V}_R(\boldsymbol{P})$ will be a generic repulsive vector generated by the obstacles on the point of interest $\boldsymbol{P}$ and expressed in the reference frame. We shall indicate with an additional subscript a particular implementation of the repulsive vector (e.g., $\boldsymbol{V}_{R\ \mathrm{ll}}(\boldsymbol{P})$).

. *Using an Estimation of the Obstacle Velocity*

Knowing in advance the velocity of a moving obstacle or, more realistically, estimating it on-line in a reliable way would clearly improve the collision avoidance behavior. For example, when an obstacle proceeds towards the manipulator with a greater speed than the motion capability of the manipulator it would be hopeless to avoid collision by retracting the robot in the same direction of the obstacle velocity. Like for humans, a better reaction strategy is to escape collision by moving the manipulator in a direction (approximately) normal to the obstacle velocity.

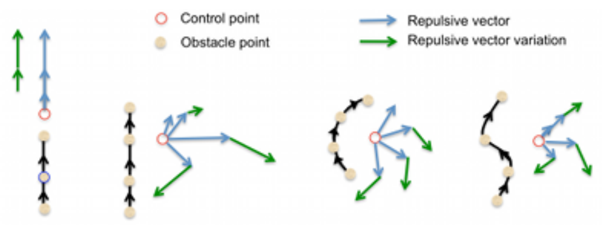

Fig. 6. When an obstacle moves in the vicinity of the control point, the pivot effect is clearly visible

The estimation of the obstacle velocity from depth images is not trivial, and also computational expensive. Our idea to tackle this problem is to extrapolate velocity information by observing the time variation of the repulsive vector, i.e., $\dot{\boldsymbol{V}}_R(\boldsymbol{P}) = d\boldsymbol{V}_R(\boldsymbol{P})/dt$. Figure 6 sketches the behavior of the repulsive vector when an obstacle moves in the vicinity of the point of interest (also called *control point*) in different ways. This point acts like a 'pivot' for the repulsive vector, and the variation $\dot{\boldsymbol{V}}_R(\boldsymbol{P})$ approximately describes a vortex flow around this pivot.

Taking into account this effect, we developed the following *Pivot lgorithm*, which modifies the direction of the repulsive vector according to its variation:

```
$\boldsymbol{a} = \frac{\dot{\boldsymbol{V}}_R(\boldsymbol{P})}{\|\dot{\boldsymbol{V}}_R(\boldsymbol{P})\|}, \quad \boldsymbol{r} = \frac{\boldsymbol{V}_R(\boldsymbol{P})}{\|\boldsymbol{V}_R(\boldsymbol{P})\|}, \quad \beta = \arccos \ \boldsymbol{a}^T\boldsymbol{r})$
if $\beta < \frac{\pi}{2}$ then
    $\boldsymbol{n} = \boldsymbol{a} \times \boldsymbol{r}, \quad \boldsymbol{v} = \frac{\boldsymbol{n} \times \boldsymbol{a}}{\|\boldsymbol{n} \times \boldsymbol{a}\|},$
    $\gamma = \beta + \frac{\beta \quad \frac{\pi}{2}}{1 + e^{\ (\|\dot{\mathbf{V}} \quad (\boldsymbol{P})\|(2/\dot{\mathrm{V}} \ _{\mathrm{m\ x}}) \quad 1)c}},$
    $\boldsymbol{V}_{R_{\mathrm{pivot}}}(\boldsymbol{P}) = \|\boldsymbol{V}_R(\boldsymbol{P})\| \left(\cos\gamma\,\boldsymbol{a} + \sin\gamma\,\boldsymbol{v}\right)$
else
    $\boldsymbol{V}_{R_{\mathrm{pivot}}}(\boldsymbol{P}) = \boldsymbol{V}_R(\boldsymbol{P})$
end if
```

In this algorithm, β represents the angle between $\boldsymbol{V}_R(\boldsymbol{P})$ and $\dot{\boldsymbol{V}}_R(\boldsymbol{P})$. If β is larger than $\pi/2$, the obstacle is moving away from the control point, such that no modification of the repulsive vector is needed. Vector $\boldsymbol{n}$ is normal to the plane to which both $\boldsymbol{V}_R(\boldsymbol{P})$ and $\dot{\boldsymbol{V}}_R(\boldsymbol{P})$ belong, while vector $\boldsymbol{v}$ is normal to $\boldsymbol{n}$ and $\dot{\boldsymbol{V}}_R(\boldsymbol{P})$. The orthonormal base specified by the unitary vectors $(\boldsymbol{a}, \boldsymbol{n}, \boldsymbol{v})$ is used to modify the orientation of $\boldsymbol{V}_R(\boldsymbol{P})$ on the plane . The new angle γ of the repulsive vector on the plane is defined as a function (shaped by the positive scalar c) of the magnitude of the variation of the repulsive vector $\|\dot{\boldsymbol{V}}_R(\boldsymbol{P})\|$, and it is equal to β if the variation of the repulsive vector is zero while it converges to $\pi/2$ if the variation of the repulsive vector tends to the maximum allowed variation of the repulsive vector $\dot{V}_{R_{max}}$. When β is very close to zero (i.e., $\boldsymbol{a}$ and $\boldsymbol{r}$ are almost orthogonal), a small perturbation to the orientation of $\dot{\boldsymbol{V}}_R(\boldsymbol{P})$ is needed in order to apply the algorithm in a robust fashion. Figure 7 shows two simple simulations of a point-wise obstacle moving close to the control point. In the first case the repulsive vector is used directly, while in the second our pivot algorithm is applied with success.

B. *Using Multiple Control Points*

The repulsive vector for obstacle avoidance introduced so far is computed for and acts on a specific point of interest. When this point of interest is associated with the robot manipulator, it is usually referred to as control point. We typically consider multiple control points distributed along the manipulator structure. From the current known configuration $\boldsymbol{q}$ of the robot, it is possible in principle to obtain

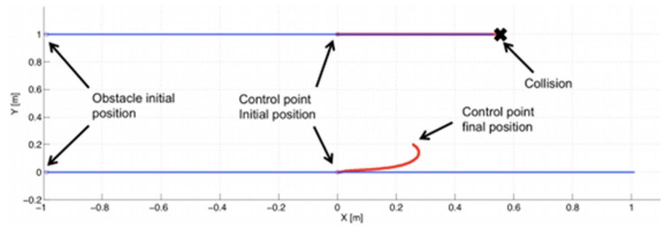

Fig. 7. Two simulations of collision avoidance using obstacle velocity information: The obstacle (trace in blue, moving to the right) has a speed of 1 [m/s], while the control point (in red) can move with a maximum speed of 0.8 [m/s]. The repulsive vector is either used directly as a repulsive velocity (top) or is processed by the pivot algorithm (bottom): In the first case, collision cannot be avoided

the projection in the depth space of every point belonging to the manipulator body by using the robot kinematic model to express the point in the reference frame, and then applying eqs. (2) and (3) to obtain its projection in the depth space. Obviously, this is quite cumbersome since not all manipulator points are needed or useful for collision avoidance. In a more efficient algorithm the manipulator body is overbounded by a sequence of spheres, each characterized by its center and radius. The centers of the spheres will be used as control points, and their radius will be subtracted in the distance evaluation of eq. (4) in order to consider the sphere size.

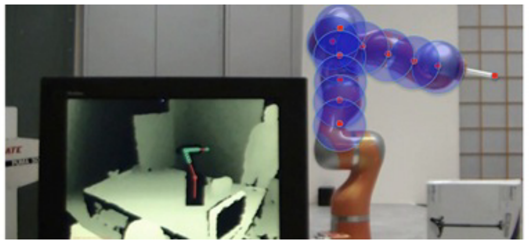

Fig. 8. The control points on the manipulator are the red centers of the blue spheres containing the robot body. The image on the screen below left visualizes the depth map sensed by the sensor in which the manipulator projection has been removed

Furthermore, the image captured by the depth sensor will contain also points that belong to the manipulator. Indeed, these points should not be considered as obstacles (otherwise, the minimum obstacle-robot distance would always be zero). To avoid this condition, the image of the manipulator is removed from the depth space as in the screen image shown in Fig. 8.

V. Motion Control

In this section, we discuss how repulsive vectors obtained for the control points chosen on the manipulator can be used in a simple way to modify the robot motion within an on-line trajectory generation architecture. The primary task for a robot is typically to control its end-effector motion. For this reason, we used a different approach for the end-effector and for other manipulator points.

. *Collision voidance for the End-Effector*

Without loss of generality, we consider to command the manipulator at the joint velocity level. The given motion task for the robot is specified by a desired end-effector velocity $\dot{\boldsymbol{x}}_d$ in the Cartesian space. For the obstacle avoidance by the end-effector control point $\boldsymbol{P}_{EE}$, we simply consider the repulsive vector as a repulsive velocity. Thus, the commanded end-effector velocity will be

$$\dot{\boldsymbol{x}}_c = \dot{\boldsymbol{x}}_d + \boldsymbol{V}_R(\boldsymbol{P}_{EE}). \qquad (10)$$

The joint velocity is obtained by (pseudo)inversion as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}\ \ (\boldsymbol{q})\,\dot{\boldsymbol{x}}_c, \qquad (11)$$

which is used as target velocity command for the control algorithm.

This is indeed a simple form of the classical artificial potential field method, which has been chosen to prove the effectiveness of repulsive vectors. The main drawback of this approach is the presence of local minima, which are not considered. Note that from a safety point of view it is acceptable that the robot stops when it is not able to avoid the obstacles. Starting from this simple algorithm, more complex ones can be developed (see, e.g., [14]).

B. *On-Line Trajectory Generation*

The repulsive action has to be very reactive in order to avoid fast obstacles. The result of this requirement is a jerky end-effector motion, which can both exceed the robot capabilities and give to the human a feeling of an unsafe motion. To overcome this behavior, we have used an intermediate layer with an on-line trajectory generation algorithm [21], [22] as interface between the proposed repulsive method and the low-level motion controller. Thanks to this intermediate layer, a number of advantages are achieved:

- Jerk-limited and continuous motions are guaranteed independently of image processing signals.
- cceleration and velocity constraints due to limited dynamic robot capabilities can be *directly* considered.
- Physical and/or artificial workspace limits can be explicitly applied.
- In case of sensor failures or inappropriate image processing results, deterministic and safe reactions and continuous robot motions are guaranteed.
- The image processing hard- and software does *not* necessarily have to be real-time capable.
- High performance due to low latencies, because motion trajectories are computed within one low-level control cycle (typically, 1 msec or less).
- The proposed architecture is of a very simple nature and can be integrated in many existing robot motion control systems.

Figure 9 shows the input and output parameters of the corresponding algorithm. Because the underlying concept of this framework is based on motion states only, all input parameters may change arbitrarily based on image processing signals, and a steady jerk-limited, executable motion trajectory is always generated as output.

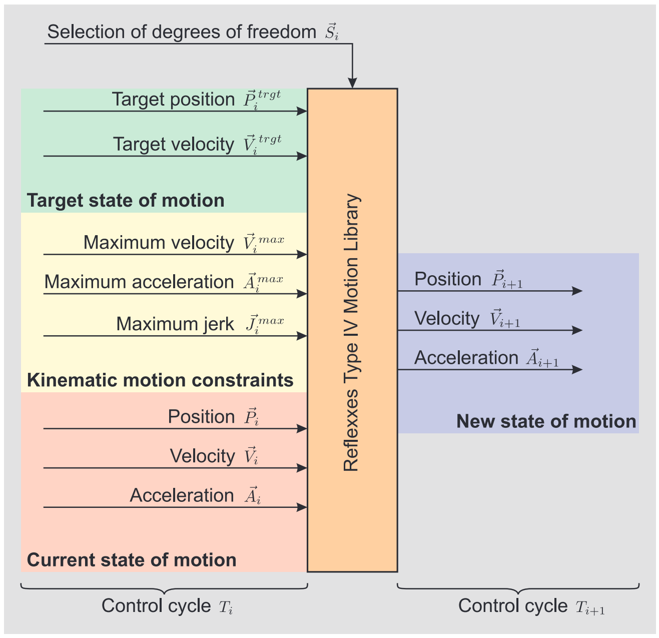
Selection of degrees of freedom $\vec{S}_i$
Target position $\vec{P}_i^{trgt}$
Target velocity $\vec{V}_i^{trgt}$
Target state of motion
Maximum velocity $\vec{V}_i^{max}$
Maximum acceleration $\vec{A}_i^{max}$
Maximum jerk $\vec{J}_i^{max}$
Kinematic motion constraints
Position $\vec{P}_i$
Velocity $\vec{V}_i$
Acceleration $\vec{A}_i$
Current state of motion
Reflexxes Type IV Motion Library
Position $\vec{P}_{i+1}$
Velocity $\vec{V}_{i+1}$
Acceleration $\vec{A}_{i+1}$
New state of motion
Control cycle T_i
Control cycle T_{i+1}

Fig. 9. The interface of the on-line trajectory generation algorithm. Based on the current state of motion and the kinematic motion constraints, a new state of motion is calculated which lies exactly on the time-optimal trajectory reaching the desired target state of motion (see [22] for details)

C. Collision voidance for the Robot Body

For the other control points along the robot, we use a slightly different approach. Obstacles points do not generate repulsive velocities on these control points, but they are rather treated as Cartesian constraints with artificial forces that are translated into joint velocity constraints as detailed in [23]. Our approach, based on generating and eventually imposing joint velocity constraints while exploiting kinematic redundancy, will preserve the desired end-effector task as long as possible. If we had considered instead repulsive velocities, as for the end-effector, we would have needed to work with multiple tasks and manage these tasks using the magnitudes of the repulsive vectors as associated priorities. While this approach is indeed feasible, if the end-effector task has always the highest priority then collision avoidance for the robot links would not be guaranteed. On the other hand, if the end-effector task is not privileged then its trajectory could be arbitrarily modified even when there is no risk of end-effector collisions.

Let be one of the control points belonging to a generic robot link and $\boldsymbol{J}_C$ be its associated (partial) Jacobian. The minimum distance between the control point and all obstacle points $\in \boldsymbol{S}(\ \ )$ is $D_{\min}(\ \ ) = \|\boldsymbol{D}(\ \ ,\ \ _{\min})\|$. The risk of collision is defined by the function

$$f\left(D_{\min}(\ \ )\right) = \frac{1}{1 + e^{(D_{\min}(\ \ )(2/\rho)\ \ 1)}}\ , \tag{12}$$

where ρ and have been introduced in eqs. (5) and (7), respectively. When projected in the joint space, this collision risk function generates a vector

$$\boldsymbol{s} = \boldsymbol{J}_C^T\, \frac{\boldsymbol{D}(\ \ ,\ \ _{\min})}{\|\boldsymbol{D}(\ \ ,\ \ _{\min})\|}\, f\left(D_{\min}(\ \ )\right). \tag{13}$$

The component s_i of $\boldsymbol{s}$ represents the 'degree of influence' of the Cartesian constraint on the ith joint, for $i = 1 \ldots n$. From these, we reshape the admissible limits of the velocity of all joints that are influenced by the Cartesian constraint by the risk of collision function as

$$\begin{array}{lll} \text{if } s_i \geq 0, & \dot{q}_{\max,i} = V_{\max,i}\ \left(1 \quad f\left(D_{\min}(\ \ )\right)\right) \\ \text{else} & \dot{q}_{\min,i} = \quad V_{\max,i}\ \left(1 \quad f\left(D_{\min}(\ \ )\right)\right), \end{array} \tag{14}$$

where $V_{\max,i}$ is the original bound on the ith joint velocity, i.e., $|\dot{q}_i| \leq V_{\max,i}$, for $i = 1, \ldots, n$. In practice, joint motions that are in contrast with the Cartesian constraint are scaled down. When the constraint is too close, all joint motions that are not compatible with the constraint will be denied. Multiple Cartesian constraints are taken into account by considering, for each joint i, the minimum scaling factor obtained for all the constraints. With this approach, collision avoidance for the robot body has always the highest priority, while the end-effector task will continue to be correctly executed until it is compatible with the Cartesian constraints. Otherwise, the manipulator will stop and a recovery method should be applied.

VI. Experiments

. Experimental Setup

The scenario is composed by a manipulator that executes positional only (i.e., of dimension $m = 3$) motion tasks through a sequence of desired Cartesian points, while unknown obstacles enter its workspace (see Fig. 1). Experiments have been performed on the KUK LWR IV manipulator having $n = 7$ revolute joints, with a control cycle of 1 ms. For the primary Cartesian task, this robot has degree of redundancy $n \quad m = 4$. The robot workspace is monitored by a Microsoft Kinect$^{\text{TM}}$ depth sensor, positioned at a horizontal distance of 2 meters and at a height of 1.2 meters w.r.t. the robot base frame. The Kinect captures 640×480 depth images at a frequency of 30 Hz. The implementation of our new collision avoidance approach is executed on an eight-core CPU. Four processors execute the repulsive velocity computation, and the other four enable visualization and robot motion control.

Note that three different run-time processes coexist, working at three different frequencies:

1) The vision process captures the depth image and removes the manipulator from the image each time a new image is captured at the sensor frequency (30 Hz).
2) The on-line trajectory generation algorithm of [21], [22] produces a joint velocity command at the same cycle time of the robot controller (1 kHz).
3) The obstacle avoidance process computes a repulsive vector at a frequency lying between those of the vision and the control processes; in fact, even if a new depth image is available only at 30 Hz, the manipulator is

moving during this interval and the repulsive vector changes.

B. Results

The following experimental results are shown in the accompanying video clip.

In Experiment 1, the goal is to keep the robot in the constant (initial) configuration while avoiding any collision between a human and the end-effector only. The parameters used for the end-effector repulsive action are $\rho = 0.4$ [m], $V_{\max} = 2$ [m/s], and $= 6$. For the pivot algorithm, we used $\dot{V}_{R_{m\ x}} = 0.5$ [m/s] and $c = 5$. Two different modalities are tested; in the former, repulsive velocities are generates using only distance information, while in the latter also obstacle velocities are taken into account. Figure 10 shows a human who tries to touch the end-effector. The benefits of using obstacle velocity estimation is visible in the corresponding plots of Fig. 11.

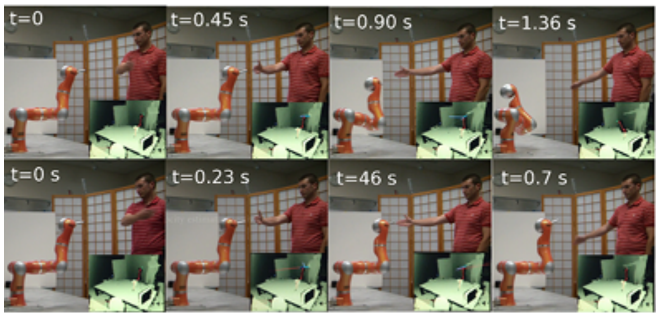

Fig. 10. Experiment 1: Image flows when repulsive velocity due only to distances is used (top) and when also obstacle velocities are considered (bottom)

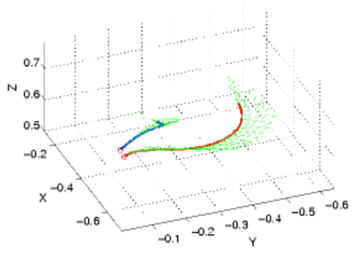

Fig. 11. Experiment 1: End-effector trajectories. The red line refers to when repulsive velocity is due only to distances and the blue line when also obstacle velocity is considered. The green lines are the commanded Cartesian velocities

Figures 12–13 show similarly the results for Experiment 2, in which the manipulator task is to move the end-effector through three Cartesian points without and with dynamic obstacles. We used here $\rho = 0.3$ [m] and $V_{\max} = 0.5$ [m/s]. In this case, the human does not try to collide intentionally with the robot, so that a less reactive action is obtained.

Experiment 3 considers collision avoidance also for other control points on the manipulator, see Fig. 14. The arm and forearm of the robot are both covered with five control points as shown in Fig. 7, while the base link is not considered

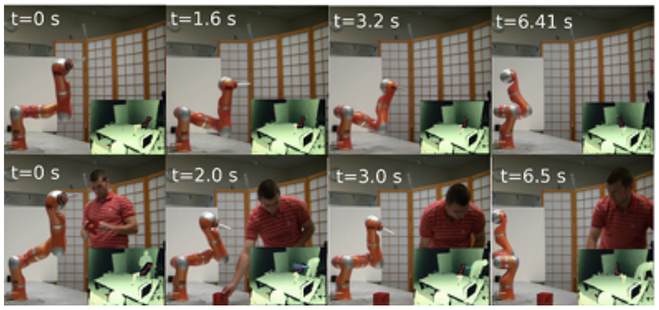

Fig. 12. Experiment 2: Image flows for a motion through three points in the absence of dynamic obstacles (top) and simultaneous collision avoidance of a human entering the workspace (bottom)

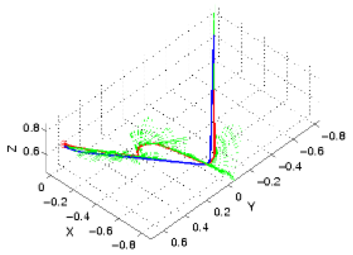

Fig. 13. Experiment 2: End-effector trajectories. The blue line refers to the absence of dynamic obstacles and the red line to the case of collision avoidance of a human entering the workspace. The green lines are the commanded Cartesian velocities

because it always occupies the same Cartesian area. The task and parameters are the same as in Experiment 2. The image sequence and the relative plot show how the elbow avoids the collision with the box.

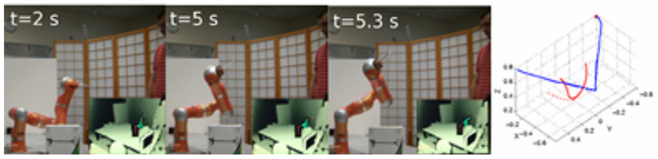

Fig. 14. Experiment 3: Image flow (on the left) and end-effector (in blue) and elbow (in red) trajectories (on the right) when all moving parts of the robot are considered for collision avoidance. The dashed lines in the plot represents the motion in the absence of obstacles

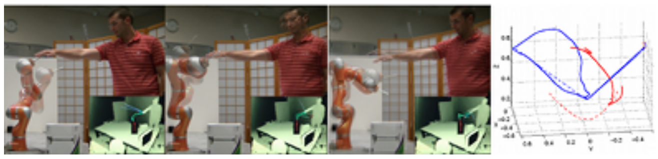

Fig. 15. Experiment 4: Same as for Experiment 3 but with multiple obstacles

Finally, Experiment 4 illustrates the robot behavior when multiple obstacles that occlude the end-effector and elbow trajectories are avoided simultaneously. In particular, Figure 15 shows a situation where the end-effector turns around the human arm. n exemplary evaluation of the execution frequency of the repulsive velocity computations is reported in Fig. 16. The average value is 689.41 Hz, which is much faster than other robot to obstacles distance evaluation methods, leading to no significant loss of robot motion performance.

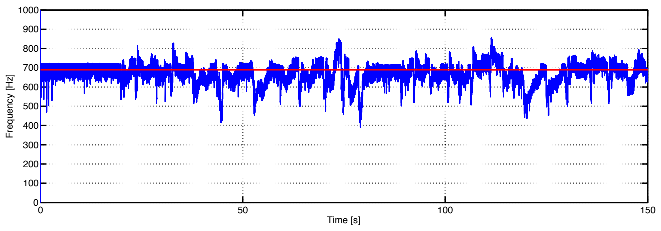

Fig. 16. Experiment 4: Instantaneous (blue) and average (red) computational frequency of the repulsive velocities

VII. Conclusions

We have presented a new collision avoidance method for robot manipulators equipped with an exteroceptive depth sensor. The core of the algorithm is an innovative approach to evaluate the distances between the robot and the dynamic obstacles in its workspace, which is based only on simple and efficient computations on depth space data. These distances are used to generate repulsive vectors which are processed so as to obtain smooth and feasible joint velocity commands that avoid obstacles. Further improvements in terms of natural robot behavior were obtained by using also an estimation of the obstacle velocity. different repulsive action has been designed for the end-effector and for the other control points on the manipulator in order to be able to avoid collisions while executing at best the original Cartesian motion task. series of experiments on the KUK LWR IV robot using the Kinect sensor confirmed the real-time effectiveness and good performance of the method.

Future work will address an even closer integration of human-robot coexistence and cooperation by monitoring the physical interaction by means of exteroceptive and proprioceptive sensors and by applying safer and reactive control methods. For instance, we would like to allow intentional contacts between human and robot while dangerous and undesired collisions should still be avoided. These and related problems are being addressed within the European FP7 project S PH RI (2011-15).

cknowledgements

This work was performed while the first author was visiting the rtificial Intelligence Laboratory of the Stanford University, whose hosting is gratefully acknowledged. Work supported by the European Community, within the FP7 ICT-287513 S PH RI project.

References

[1] . Bicchi, M. Peshkin, and J. Colgate, "Safety for physical human-robot interaction," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008, pp. 1335–1348.

[2] . Bicchi and G. Tonietti, "[Fast and Soft rm Tactics: Dealing with the Safety-Performance Trade-Off in Robot rms Design and Control," *IEEE Robotics and utomation Mag.*, vol. 11, pp. 22–33, 2004.

[3] S. Haddadin, . lbu-Schaffer, . De Luca, and G. Hirzinger, "Collision detection and reaction: contribution to safe physical human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, F, September 2008, pp. 3356–3363.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[5] C. W. Warren, "Global path planning using artificial potential fields," in *Proc. of the IEEE International Conference on Robotics and utomation*, vol. 1, Scottsdale, Z, US , May 1989, pp. 316–321.

[6] P. Ögren, M. Egerstedt, and X. Hu, "Reactive mobile manipulation using dynamic trajectory tracking," in *Proc. of the IEEE International Conference on Robotics and utomation*, San Francisco, C , US , pr. 2000, pp. 3473–3478.

[7] O. Brock and O. Khatib, "Elastic strips: framework for motion generation in human environments." *Int. J. of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[8] S. Lindemann and S. LaValle, "Current issues in sampling-based motion planning," in *Proc. of the Eighth Int. Symp. on Robotics Research*, P. Dario and R. Chatila, Eds. Berlin, Germany: Springer, 2004, pp. 36–54.

[9] O. Brock, J. Kuffner, and J. Xiao, "Manipulation for robot tasks," in *Springer Handbook of Robotics*, 1st ed., B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg, Germany: Springer, 2008, ch. 26, pp. 615–645.

[10] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (R MP) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1199–1212, Oct. 2008.

[11] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments," in *Proc. of Robotics: Science and Systems*, Philadelphia, P , US , ug. 2006.

[12] ——, "Elastic roadmaps — motion generation for autonomous mobile manipulation," *utonomous Robots*, vol. 28, no. 1, pp. 113–130, Jan. 2010.

[13] L. Balan and G. Bone, "Real-time 3D collision avoidance method for safe human and robot coexistence," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, PRC, October 2006, pp. 276–282.

[14] S. Haddadin, S. Belder, and . lbu-Schaeffer, "Dynamic motion planning for robots in partially unknown environments," in *IF C World Congress (IF C2011)*, Milan, Italy, September 2011.

[15] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, . lbu-Schäffer, and G. Hirzinger, "Real-time reactive motion generation based on variable attractor dynamics and shaped velocities," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 3109–3116.

[16] S. Kuhn and D. Henrich, "Fast vision-based minimum distance determination between known and unknown objects," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, San Diego, C , US , November 2007, pp. 2186–2191.

[17] Microsoft Corporation, 1 Microsoft Way, Redmond, W 98052-7329, US , "Microsoft kinect homepage. http://xbox.com/Kinect (accessed: Mar. 28, 2011)," Internet, 2011.

[18] L. Bascetta, G. Magnani, P. Rocco, R. Migliorini, and M. Pelagatti, " nti-collision systems for robotic applications based on laser Time-of-Flight sensors," in *IEEE/ SME Int. Conf. on dvanced Intelligent Mechatronics*, July 2010, pp. 278–284.

[19] R. Schiavi, F. Flacco, and . Bicchi, "Integration of active and passive compliance control for safe human-robot coexistence," in *Proc. IEEE Int. Conf. on Robotics and utomation*, 2009, pp. 259–264.

[20] KUK Laboratories GmbH, Zugspitzstraße 140, D-86165 ugsburg, Germany, "Homepage. http://www.kuka-labs.com/en (accessed: ug. 22, 2011)," Internet, 2011.

[21] T. Kröger and F. M. Wahl, "On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. on Robotics*, vol. 26, no. 1, pp. 94–111, Feb. 2010.

[22] T. Kröger, "Opening the door to new sensor-based robot applications — The Reflexxes Motion Libraries," in *Proc. of the IEEE International Conference on Robotics and utomation*, Shanghai, China, May 2011.

[23] F. Flacco, . De Luca, and O. Khatib, "Motion control of redundant robots under constraints: Saturation in the null space," in *Proc. IEEE Int. Conf. on Robotics and utomation*, 2012.