6th International Conference
ðComputational Mechanics and Virtual Engineering ð
COMEC 2015
15-16 October 2015, Braşov, Romania

NEW STRATEGY OF THE RECIPROCAL COLLISION AVOIDANCE OF COLLABORATIVE ROBOTS

Aurel Fratu[1]
[1] "Transilvania" University of Brasov, ROMANIA, e-mail fratu@unitbv.ro

Abstract: *This paper, presents a formal approach that addresses the reciprocal robots collision avoidance, where collaborative robots need to avoid collisions one with each other while moving in a common workspace. Based on our formulation, each physical robot acts fully independently, communicating with corresponding virtual prototype and imitating her behavior. Each physical robot reproduces the pathway of her virtual homonymous prototype. With a view to collision avoidance, it is necessary to detect a possible collision. This action includes the potentially intersecting regions test of the corresponding virtual prototypes. The estimation of the collision-free actions on the virtual robots and the collaborative work of the physical robots are the original ideas. Based on potentially intersecting regions of the virtual robots, we identified a collision-free motion corridor for each collaborative robot.*
Keywords **:** collision detection, motion corridor, reciprocal collision avoidance

1. INTRODUCTION

Collaborative robots systems are designed to achieve tasks by collaboration. Collaborative robots, we see being deployed nowadays in research or industry, are permanently in danger to be in collision. Therefore installations with multiple robots in real world, such as collaborative work and maintenance the good state of the line production, require collision avoidance methods, which take into account the mutual constraints of the robots.
The types of tasks that can be automated using collaborative robotics are essentially unstructured tasks where it is hard or too expensive to use traditional industrial robotics. Collaborative robots and classical industrial robots are not competitors. Collaborative robotics is a new kind of robotics that is complementary of industrial robots. Collaborative robots are easy to use with the method proposed in this paper.. This is a revolution; there is no need for expert programmers that is expensive. Any worker without knowledge of robotics or computer science can program the robot.
Collision avoidance is a fundamental problem in robotics. Specifically accounting for the reactive nature of the other robots is called reciprocal collision avoidance. In this paper we present a formal approach to reciprocal collision avoidance for multiple robots sharing a common 2-D or 3-D workspace whose dynamics are subject to linear constraints. Our approach defines a protocol for robots to select their control path in coordination with other robots, while guaranteeing collision-free motion for all robots, assuming the robots can perfectly imitate her virtual homonymous. Robots are programmed by imitation of their virtual homonyms, just by moving its arm and demonstrate the movement that is supposed to do. A key requirement for their efficient operation is good coordination of the virtual prototypes and reciprocal collision avoidance of the physical robots.
Collision avoidance is a highly advanced robot control option that automatically detects collisions and quickly causes the robot to stop and back up to release the pressure. Not only does it reduces the force of the collision, but also prevents the robot and its tooling from being pressed against an object after a collision.
The contact of the robot with an obstacle must bee detected and it will cause the robot to stop quickly and thereafter back off to reduce forces between the robot and environment. The problem of the contact with obstacle imposes the null velocity in the moment of impact. The problem of the contact detection is better analyzed on the virtual prototypes in the virtual environment, where the virtual objects can be intersected and there no exist the risk to be destroyed. The problem of the contact detection in the virtual environment using virtual robots is important for the reason that this

built-in function is proven superior to mechanical collision detection devices. It detects collisions in all directions, protecting not only the end-effectors but also the work pieces and the robot itself.
In the case that the real (physical) robot will imitate her virtual prototype, it has no mechanical parts which give it higher reliability and more cost efficiency. Also, since there is no device attached to the real robot tool, one not extends the tool offset distance, which allows bigger maximum tool weight and better reorientation performance.
The problem of the collision avoidance can generally be defined in the context of an autonomous robot operating in an environment with obstacles, and/or other moving entities, where the robot employs a continuous cycle of acting.
For each cycle, an action for the robot must be computed based on local observations of the environment, such that, the robot stays free of collisions with the obstacles and the other moving entities, while making progress towards a goal.
The problem of local collision-avoidance differs from motion planning, where the global environment of the robot is considered to be known and a complete pathway towards a goal configuration is planned at once. So the collision detection simply determines if two geometric objects are intersecting or not. The intersecting of two objects is possible in the virtual world, where one may predict there behavior.
The ability of predicting of the behavior of cooperative manipulators is important for several reasons: for example, in design the designers want to know whether the manipulator will be able to perform a certain typical task in a given time frame; in creating feedback control schemes, where stability is a major problem, the control engineer cannot risk a valuable piece of equipment by exposing it to untested control strategies. Therefore, a facile strategy for collision avoidance, capable of predicting the behavior of a robotic manipulator or of a system at whole, becomes imperative.
In a real world, like collision detection, where the robots need to interact with their surrounding, it is important that the computer can simulate the interactions of the cooperative participants, with the passive or active changing environment, using virtual prototyping.

2. OVERVIEW OF THE RECIPROCAL COLLISION AVOIDANCE BASED ON NEW STRATEGY

Collision avoidance is a fundamental problem in many areas such as robotics and animation. The problem of local collision-avoidance differs from motion planning, where the global environment of the robot is considered to be known and a complete pathway towards a goal configuration is planned at once. So the collision detection simply determines if two geometric objects are intersecting or not. The intersecting of two objects is possible in the virtual world, where one may predict there behavior. Optimization of the real robots behavior is performed in the low dimensional virtual space using the virtual robot prototypes.
In this paper, we present a study that provides the guarantee of the collisions avoiding for virtual collaborative robots. The virtual collaborative robots operate in a complex virtual environment, containing both static and moving obstacles. Our approach treats the virtual prototypes of collaborative robots that operate while guaranteeing collision-free motion. We transfer the trajectory of the virtual robots to the physical robots, in the real environment, assuming that each physical robot can perfectly imitate the movement/behavior of her virtual prototype.
We present the hybrid reciprocal velocity obstacle and optimal reciprocal collision avoidance methods for reciprocal collision avoidance and navigation in video scenes and described their implementations in C++ and Object Delphi as Hybrid Reciprocal Velocity Obstacle (HRVO) Library. The librarie can efficiently simulate groups of virtual agents in dense conditions and around moving and static obstacles.
In the virtual space one simulate even the intersecting of the virtual robot and her environment. The intersecting of two virtual objects is possible in the virtual world, where the virtual objects can be even intersected and there no exist the risk to be destroyed. The contact detection between the virtual objects is responsible for determining the distance between two points in a space and for optimization of the trajectories of motion. This strategy may use simple CAD methods or may be extended to be more complex and take into account invalid areas of the space.
In a real world, like collision detection, where the robots need to interact with their surrounding, it is important that the computer can simulate the interactions of the collaborative participants, with the passive or active changing environment, using virtual prototyping.
Conventional programming strategies are hardly applicable to robotic manipulators that joints must be coordinated concurrently and must assure a stable dynamics [5]. Therefore, exploiting virtual robots' potential in programming tasks for the reciprocal collision avoidance remains an atypical objective.
Virtual robots interactions studies are very promising and they allow to run free a lot of advances in collaborative tasks.
We propose an original idea that allows us to transfer the joint trajectory of each virtual robot to the corresponding joint of the real (physical) robot. With other worlds we prepare the free-trajectories for the pair of the virtual cooperative robots; these trajectories can be transferred to their corresponding pair of the physical cooperative robots.

In the real world, the programming pathway of the physical robots can be realized using the pathway of the virtual robot prototypes [22]. Therefore, the virtual robot behavior must be specifically taken into account in order to guarantee that collisions are avoided between corresponding physical robots. Each real robot may be able to communicate with her virtual corresponding entity and may imitate their intents. We call this problem *reciprocal collision avoidance using the virtual robots prototypes*, and it is the focus of this paper.
The strategy that we realized in this paper, for the virtual robots, is formally defined as follows. Let there be a set of virtual partners-robots sharing a virtual environment. Each robot has a current position and a current velocity. These parameters are part of the robot's external state, i.e. we assume that they can be estimate in the virtual environment.
Furthermore, each virtual robot has a maximum speed and a preferred velocity, which is the velocity the robot would assume had no other robot / object been in its way. We consider these parameters part of the internal state of the robot, and can therefore not be observed by the other robot.
The task is for each virtual robot to independently (and simultaneously) select a new velocity for itself such that both virtual robots are guaranteed to be collision-free for at least a fixed amount of time, when they would continue to move at their new velocity. To prevent nearby virtual agents from collisions, velocity-based methods use the current velocity of each virtual agent in the group and then extrapolate the position of each virtual agent for some short time interval under the assumption that the virtual agent will maintain almost a constant velocity over some short time interval. Based on predicting the future positions of other virtual agents, each virtual agent tends to choose an avoiding new velocity based on some optimization.
As a secondary objective, the virtual robots should select their new velocity as close as possible to their preferred velocity. The virtual robots are not allowed to communicate with each other, and can only use observations of the partner-robot's current position and velocity. The virtual robot prototypes are used mainly as an intermediate result for calculating the "nearest neighbors" and the potentially intersecting areas in an eventually collision.

3. COLLISION DETECTION

Collision detection frequently arises in various applications including virtual prototyping, dynamic simulation, interaction and motion planning. Collision detection has been exhaustively researched for more than four decades. Most of the commonly used algorithms are based on spatial partitioning or Bounding Volume Hierarchies (BVHs).
Typically, for a simulated environment consisting of multiple moving virtual objects collision problems consist of two phases: the "broad phase" where collision is performed to reduce the number of pair intelligent tests, and the "narrow phase" where the pairs of objects in proximity are checked for collision.
In this section, we present a study based on collision detection algorithm for computing all the contacts between multiple moving virtual objects in a large virtual environment. It uses the visibility reducing algorithm described in [3]. The overall algorithm is general and applicable to all environments. We also highlight many optimizations and the visibility queries used to accelerate the performance of the algorithm.
Algorithms for narrow phase can be further subdivided into efficient algorithms for convex objects and general purpose algorithms based on spatial partitioning and BVHs, for polygonal models [2].However, these algorithms often involve pre-computation and are mainly designed for rigid models.

3.1. Potentially Colliding Set

We compute a Potentially Colliding Set **(**PCS) of virtual objects that are either overlapping or are in close proximity [2]. If an object does not belong to the PCS, it implies that this object does not collide with any object. We initially compute the PCS of virtual objects based on the above algorithm. As an alternative of testing each object pair in the PCS for exact partly cover, we more than use the visibility formulation to identify the potentially intersecting virtual areas among the objects.
Based on this property, we realized a programming platform for a pair real (physical) robots based on cooperative virtual robots pair, which need to be checked for exact collision detection. This platform needs to compute the exact overlapping area of the virtual cooperative robots, as collision response.
If an object O_i does not belong to the PCS, it implies that O_i does not collide with any object in the PCS. Based on this property, we can reduce the number of virtual object pairs that need to be checked for exact collision. This is similar to the concept of computing the potentially visible set (PVS) of primitives from a viewpoint for spatial relation. We

perform visibility computations between the objects in image space to check whether they are potentially colliding or not.

Given a set *S* of virtual objects, we test the relative visibility of an object *O* with respect to set *S* using an image space visibility query. The query checks whether any part of *O* is spatial intersected by *S*, rasterizing all the objects belonging to set *S*.

The object *O* is considered fully-visible if all the fragments generated by the rasterization of *O* have a depth value less than the corresponding pixels in the frame buffer. We do not consider self masking of an virtual object *O* in checking its visibility status.

If an object does not intersect in either of the two passes, then it does not belong to the PCS. Each pass requires the object representation for an object to be rendered twice.

We can either render all the triangles used to represent an object or a bounding box of the object. Initially, the PCS consists of all the objects in the scene.

We perform these two passes to reduce objects from the PCS. Furthermore, we repeat the process by using each coordinate axis as the axis of projection to further reduce the PCS. We'll check if an object potentially intersects with a set of objects or not.

It should be noted that our method based on reducing algorithm [6] is quite different from algorithms that reduce PCS using 2D overlap tests.

Our method does not perform frame-buffer read backs and computes a PCS that is less conservative an algorithm based on 2D overlap tests.

Many applications need to compute the exact overlapping features (e.g. triangles) for collision response. We initially compute the PCS of objects based on the algorithm highlighted above. Instead of testing each object pair in the PCS for exact overlap, we again use the visibility formulation to identify the potentially intersecting virtual regions among the objects in the PCS.

Specifically, we use a fast global reducing algorithm to localize these regions of interest. We perform object level reducing to compute the PCS of objects.

Initially, all the objects belong to the PCS. Firstly we perform reducing along each coordinate axis by using the axis aligned bounding boxes as the object's representation for collision detection. The reducing is performed till the PCS does not change between successive iterations.

We also use the object's triangulated representation for further reducing the PCS. The size of the resulting set is expected to be small and we use all-pair bounding box overlap tests to compute the potentially intersecting pairs.

If the size of this set is large, then we use sweep-and- reduce technique [5] to reduce this set instead of performing all-pair tests. We decompose each object into sub-objects. A sub-object can be a bounding box, a group of *k* triangles (say a constant *k*), or a single triangle. We have extended this approach to sub-object level and compute the potentially intersecting areas.

More recently, velocity-based methods have exhibited improvements in terms of local collision avoidance and behavior of virtual agents, and improved computational performance, over force-based collision avoidance methods.

For two robot agents *A* and *B*, the velocity obstacle, $VO^{\tau}_{A|B}$ (read: the velocity obstacle for *A* induced by *B* for time window τ) is the set of all relative velocities of *A* with respect to *B* that will result in a collision between *A* and *B* at some moment before time τ [9]. It is formally defined as follows.

Let be an *A* robot agent with radius r_A , positioned at $\boldsymbol{p}_A$ on a horizontal disc. For a configuration of two robot agents *A* and *B*, the horizontal disc of radius ($r_A + r_B$) is centered at ($\boldsymbol{p}_B - \boldsymbol{p}_A$) in the Cartesian space.

Let *S* ($\boldsymbol{p}$, *r*) denote an open horizontal disc of radius *r* centered at vector position $\boldsymbol{p}$ and defined by the (1).

$$S(\boldsymbol{p},r)=\{\boldsymbol{s}\in S,\|\boldsymbol{s}-\boldsymbol{p}\|\le r\} \qquad (1)$$

Then the velocity obstacle is defined as:

$$VO^{\tau}_{A|B}=\{v \mid \exists t\in[0,\tau], vt\in S(\boldsymbol{p}_B-\boldsymbol{p}_A, r_A+r_B)\} \qquad (2)$$

Let $\boldsymbol{v}_{tA}$ and $\boldsymbol{v}_{tB}$ be current the task (operational) velocities of the robot agents *A* and *B*, respectively. The definition of the velocity obstacle implies that if ($\boldsymbol{v}_A$ - $\boldsymbol{v}_B$) $\in$ $VO^{\tau}_{A|B}$, or equivalently if ($\boldsymbol{v}_B$ -$\boldsymbol{v}_A$) $\in VO^{\tau}_{B|A}$, then *A* and *B* will collide at some moment before time τ if they continue moving at their current velocity.

On the other hand, if ($\boldsymbol{v}_A$-$\boldsymbol{v}_B$) ¢ $VO^{\tau}_{A|B}$, the two robot agents *A* and *B* are guaranteed to be collision-free for at least τ time. Robot agent *A* will collide with robot agent *B* within τ time if its velocity $\boldsymbol{v}_A$ is inside $VO^{\tau}_{A|B}$ and it will be collision-free for at least τ time if its velocity is outside the velocity obstacle.
For an articulated robot arm, the robot end-effectors velocity vector is calculate as, $v = \dot{X}_t$ where $\boldsymbol{X}_t$ is the Cartesian position vector of the robot end-effector. The vector $\boldsymbol{X}_t$ can be described as a function of robot joints variables vector, $\boldsymbol{q}$:

$$X_t = \mathrm{f}(\boldsymbol{q}) \qquad (3)$$

Equation (3) can be obtained easily with the help of the Denawit - Hartenberg operators.
The robot end-effectors velocity vector **v** can be obtained as:

$$v = \dot{X}_t = \boldsymbol{J}_t(\boldsymbol{q})\dot{\boldsymbol{q}} \qquad (4)$$

where $\boldsymbol{J}$ is Jacobian matrix, and $\dot{\boldsymbol{q}}$ is the robot joints velocities vector.
Given a trajectory that each moving robot end-effector will travel, we can determine the exact collision time. Please refer to [4] for more details. If the path that each robot end-effector travels is not known in advance, then we can calculate a lower bound on collision time. This lower bound on collision time is calculated adaptively to speed up the performance of dynamic collision detection.

3.2 Planning scene formulation for collaborative tasks

In the collaborative tasks studies, the simulation is used to find whether it is possible to avoid the collision between a particular part of the robot arm and diverse objects in the work space and so to find one possible free path.
In this section we will describe how to render a planning scenario in the form of constraints for the constraint-based planning framework. Our visibility based on PCS computation algorithm is based on hardware visibility query which determines if a primitive is fully-visible or not.
By assuming that the geometry representing the robots and obstacles is given, as well as prescribed motion, simulated for the obstacles over time. Our system, then defines constraints that will restrict the motion of the robots to meet the design specifications, and also guide the robots to complete the planning tasks such as the collision will be avoided.
Our virtual system was implemented on a programming platform, using Delphi object-oriented programming language. We have tested our system of robots team for collision detection in the following scenes for the virtual prototyping applications. The collaborative tasks for each robot are studied in scene 1 (example seen in Fig. 1).

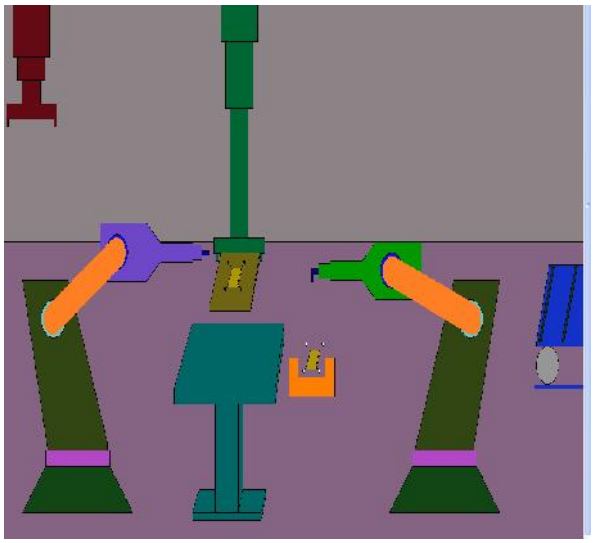

Figure 1 *Scene for collaborative robotic arms*

Each robot arms is composed of rigid components that are held together by constraints. For all of the components of the robot arms, the planner must compute paths to satisfy the joint constraints do not collide with the obstacles as the conveyors and lead the end-effectors along the prescribed path.

4. CONCLUSION

In this paper, we address the more involved and less studied problem of reciprocal robots collision avoidance, where collisions need to be avoided for multiple tasks. The ability of predicting of the behavior of collaborative manipulators is important for several reasons: for example, in design the designers want to know whether the manipulator will be able to perform a certain typical task in a given time frame; in creating feedback control schemes, where stability is a major problem, the control engineer cannot risk a valuable piece of equipment by exposing it to untested control strategies. Therefore, a facile strategy for collision avoidance, capable of predicting the behavior of a robotic manipulator, or of a system at whole, for that matter, becomes imperative.
This problem has important applications in many areas in robotics, such as multi-robot cooperation and coordination. It is also a key component in modeling and behavioral simulation of the robots for computer graphics and Virtual Reality.
In this paper, we propose a fast method that simultaneously determines actions for virtual robots that each may have different objectives. The actions are computed for each virtual robot and are transferred to corresponding physical robot, with a central coordination for the cooperative tasks. Yet, we prove that our method guarantees the collision-free motion for each of the robots. Collision avoidance and multi-agent navigation is an important component of modern robotic systems. Recent developments in commodity hardware, in particular the utilization of multi-core and many-core architectures in personal computers and consoles are allowing large numbers of virtual agents to be incorporated into scene in increasing numbers. Rather than using virtual forces to prevent nearby virtual agents from collisions, velocity-based methods use the current velocity of each virtual agent in the group and then extrapolate the position of each virtual agent for some short time interval under the assumption that the virtual agent will maintain almost a constant velocity over some short time interval. Based on predicting the future positions of other virtual agents, each virtual agent tends to choose an avoiding new velocity based on some optimization.
The libraries can efficiently simulate groups of virtual agents in constraint conditions and around moving and static obstacles.

REFERENCES

[1] Jamie Snape, Stephen J. Guy, and Dinesh Manocha, “Smooth coordination and navigation for multiple differential-drive robots,” in Oussama Khatib, Vijay Kumar, and Gaurav Sukhatme (eds.), *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, Springer Tracts in Advanced Robotics, vol. 79, Springer-Verlag, Berlin/Heidelberg, Germany, April 15, 2013, pp. 601-613.

[2] Sujeong Kim, Stephen J. Guy, Dinesh Manocha, and Ming C. Lin, “Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory,” in Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games, Costa Mesa, Calif., March 9-11, 2012, pp. 55-62.

[3] Stephen J. Guy, Sujeong Kim, Ming C. Lin, and Dinesh Manocha, “Simulating heterogeneous crowd behaviors using personality trait theory,” in Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation, Vancouver, B.C., Canada, August 5-7, 2011, pp. 43-52.

[4] Jur van den Berg, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha, “*Reciprocal n-body collision avoidance*,” in Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger (eds.), *Robotics Research: The 14th International Symposium ISRR*, Springer Tracts in Advanced Robotics, vol. 70, Springer-Verlag, Berlin/Heidelberg, Germany, May 7, 2011, pp. 3-19.

[5] Jamie Snape, Jur van den Berg, Stephen J. Guy, and Dinesh Manocha, “Smooth and collision-free navigation for multiple robots under differential-drive constraints,” in Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Taipei, Taiwan, October 18-22, 2010, pp. 4584-4589.

[6] Stephen J. Guy, Jatin Chhugani, Sean Curtis, Pradeep Dubey, Ming C. Lin, and Dinesh Manocha, “PLEdestrians: A least-effort approach to crowd simulation,” in Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation, Madrid, Spain, July 2-4, 2010, pp. 119-128.