COLLISION AVOIDANCE CONTROL USING NULL SPACE CRITERIA

Mariana FRATU, Aurel FRATU
Transilvania University of Brasov, Romania

Abstract. A new strategy to on-line collision-avoidance of the redundant robots with obstacles is presented. The strategy allows the use of redundant degrees of freedom such that a manipulator can avoid obstacles while tracking the desired end-effectors trajectory. The effectiveness of the proposed strategy based on null space criteria is discussed by theoretical considerations.

The strategy is based on the redundant inverse kinematics and leads to a favourable ability of redundant robots to avoid the collisions with obstacles. This strategy has the advantage that the configuration of the manipulator can be influenced by further requirements such as joint limits and so the self-collision is avoided.

Keywords: collision avoidance method, obstacles avoid task, self-collision

1. Introduction

Robotic systems which are capable of motion in more degrees of freedom than required for the programmed task are known as redundant robots.

Conventional obstacle avoidance, during end-effector positioning in robotic systems, is included in the high level or task programming for the system. Real time obstacle avoidance is difficult to achieve in part because of the vast amounts of data that would have to be communicated between the high level processing unit and the robot servo systems.

The problem of obstacle avoidance is to ensure that robot’s links do not collide with obstacles in the workplace, while robotic arm system moves her end-effector along a desired, preplanned trajectory to perform a task.

The obstacle avoidance strategy may be formulated as a set of kinematic inequality constraints in the tasks space, that these inequality constraints are satisfied while the desired trajectory for end-effector is tracked.

Obstacle avoidance is one of the key issues to successful applications of robot systems. All robots feature some kind of collision avoidance, ranging from primitive algorithms that detect an obstacle and stop the robot short of it in order to avoid a collision, through sophisticated algorithms, that enable the robot to detour obstacles. Other algorithms are much more complex, since they involve not only the detection of an obstacle, but also some kind of quantitative measurements concerning the dimensions of the obstacle. Once these have been determined, the obstacle avoidance algorithm needs to turn the robot around the obstacle and proceed toward the original target.

Usually, this procedure requires the robot to stop in front of the obstacle, take the measurements, and only then continue motion. Obstacle avoidance (also called reflexive obstacle avoidance or local path planning) may result in non-optimal paths [1], since no prior knowledge about the environment is used.

In this paper the authors provides a method of operating a redundant robot system to position an end-effector in a workspace by changing joint angles between links connected to the end-effector. For the first step it must be determined the location of an obstacle in the workspace and map a sphere of influence, having a fixed radius surrounds the obstacle. In second step, it must determine a critical point, on a link, closest to the sphere of influence. After this, it must determine a critical distance between the critical point and the obstacle. Finally, the robot system acts to stop motion of the critical point toward the obstacle, when the critical distance equals the radius of the sphere of influence.

In this paper the problem of redundant inverse kinematics is reviewed and obstacles avoid subtask for exploiting the self motion are defined.

In order to accomplish a task, an accurate joint motion must be commanded to the manipulator. For that reason, it is necessary to obtain mathematical relations which allow computing joint-space variables corresponding to the assigned task-space variables. This is the objective of the inverse kinematics problem. Numerous different authors dealt with the solution of the redundant inverse kinematics problem.

In this paper the authors address the problem of local obstacle avoidance for manipulator robots operating in unknown, or partially known, environments. While this problem has been studied by several other researchers, there exist a number of desiderata that necessitate a new approach to the problem. Several desiderata are common to most existing methods:

- The robot should navigate safely, even in the face of noisy sensors and dead-reckoning error;
- The robot should be goal-directed while trying to avoid obstacles;
- The method must be computationally efficient, to run in real-time on-board the robot;
- The method should explicitly try to maximize forward progress of the robot;
- The method should simultaneously control both the direction and speed of the robot.

The main distinctions of the method, presented in this approach is that it operates in the velocity space of the robot, rather than Cartesian or configuration space, and it chooses commands by maximizing an objective function that trades off robot safety, speed and goal-directedness. The method presumes that the robot can control both translational and rotational velocities, but cannot turn instantaneously.

2. Related works

Several well-known local obstacle avoidance methods work by computing a direction for the robot to head in, but do not take robot dynamics into account. For example, Potential Field approaches [2] use vector sums of repulsive and attractive features to compute a desired robot route.

Speed control is sometimes handled by choosing velocity proportional to the magnitude of the potential vector. The method proposed in this approach [3], suppose the computing a one-dimensional parameter, which are then processed to detect open areas for the robot to move through.

Robot velocity, chosen after the direction has been selected, is proportional to the distance to obstacles ahead. While this method produces smoother move and can handle both narrow and wide openings, it like the Potential Field approach, does not account for the fact that when robots turn they typically move along arcs, rather than in straight lines. In cluttered environments, this neglect of robot dynamics can be critical.

The methods that take robot dynamics and constraints into account have been studied in the context of off-line path planning; such methods are generally too computationally expensive for fast local obstacle avoidance.

Recently, several local obstacle avoidance methods have been reported that do incorporate robot dynamics, choosing steering commands rather than move direction. The Steering Angle Field method [4] uses the curvatures tangent to obstacles to constrain a continuous space (in their case, the one-dimensional space of steering angles). The curvatures and associated arc distances are used to prohibit move over ranges of steering angles. The method calculates constraints for several distance thresholds, and tries to move along the freest dimension. Speed control is an iterative “negotiation” process between the associated pilot servo-system and the associated local obstacle avoidance servo-system.

3. Problem of redundant inverse kinematics

Usually, the Cartesian position and orientation for the end-effectors task vector $\boldsymbol{x}_t$ can be described as a function of the vector of joint variables**,** $\boldsymbol{q}$ of the manipulator:

$$\boldsymbol{x}_{\mathrm{t}} = f(\boldsymbol{q}) \tag{1}$$

While equation (1) can be obtained easily with the help of Denawit-Hartenberg operators, the inverse problem is crucial. In the redundant case it is generally not possible to find an inverse mapping, f^{-1}**.** Instead of constructing an inverse function $g(\boldsymbol{x}_{\mathrm{t}})$ with $f(g(\boldsymbol{x}_{\mathrm{t}})) = \boldsymbol{x}_{\mathrm{t}}$.

As an alternative of constructing an inverse mapping, the problem is often reformulated in the velocities, utilizing the partial derivation of $f(\boldsymbol{q})$.

The end-effectors task velocity vector $\dot{\boldsymbol{x}}_{\mathrm{t}}$ is:

$$\dot{\boldsymbol{x}}_{\mathrm{t}} = \boldsymbol{J}_{\mathrm{t}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2}$$

with Jacobean matrix:

$$\boldsymbol{J}_{\mathrm{t}} = \partial\, \boldsymbol{f}(\boldsymbol{q}) / \partial \boldsymbol{q} \tag{3}$$

Due to the fact that the inverse of the non-square (analytical) Jacobean $\boldsymbol{J}_{\mathrm{t}}(q)$ does not exist in the redundant case, the well known generalized Moore–Penrose pseudo inverse $\boldsymbol{J}_{\mathrm{t}}^{+}(q)$ is utilized. This proposed strategy often employs a special solution of equation (2).

Optimization criteria for the redundant self motion can be supplementary by e.g. null-space projection, which leads to the relation:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\mathrm{t}}^{+}\dot{\boldsymbol{x}}_{\mathrm{t}} + (\boldsymbol{I} - \boldsymbol{J}_{\mathrm{t}}^{+}\boldsymbol{J}_{\mathrm{t}})\dot{\boldsymbol{q}}_0 \tag{4}$$

Here, $(\boldsymbol{I} - \boldsymbol{J}_t^{+}\boldsymbol{J}_t)$ represents the orthogonal projection matrix in the null space of $\boldsymbol{J}_{\mathrm{t}}$, and $\dot{\boldsymbol{q}}_0$ is an arbitrary joint-space velocity; the second part of the solution is therefore a null-space velocity.

The vector $\dot{\boldsymbol{q}}_0$ can be chosen arbitrary and is used to force a desired behaviour for the null space motion. The null space of $\boldsymbol{J}_{\mathrm{t}}(q)$ is defined as:

$$N(\boldsymbol{J}_{\mathrm{t}}(\boldsymbol{q}))=\{\dot{\boldsymbol{q}}\in\dot{\boldsymbol{Q}}: 0=\boldsymbol{J}_{\mathrm{t}}(\boldsymbol{q})\dot{\boldsymbol{q}}\} \tag{5}$$

In redundant directions joint velocities causes no motion at the end-effectors level. These are internal motions of the manipulator. Redundant joint velocities satisfy the equation:

$$\boldsymbol{J}_t(\boldsymbol{q})\dot{\boldsymbol{q}} = 0 \tag{6}$$

At each configuration, the null space of $\boldsymbol{J}_{\mathrm{t}}$ is the set of joint-space velocities which yield zero task velocity; these are thus called null-space velocities.

While most non-redundant manipulators possess enough Degrees-of-Freedom (DoFs) to perform their main task(s), i.e., position and/or orientation tracking, it is known that their limited manipulability results in a reduction in the workspace due to mechanical limits on joint articulation and presence of obstacles in the workspace.

4. Problem of the collision avoidance with redundant robots

The redundant robots can operate in the Cartesian space with obstacles. The redundant self motion can satisfy both, the end-effector task and the additional constraint task cause of the obstacles, at the same time.

The concept of task-space augmentation introduces a constraint task to be fulfilled along with the end-effector task. In that case, an augmented Jacobean matrix is set-up whose inverse gives the required joint velocity solution.

Let us consider the p-dimensional vector $\boldsymbol{x}_{\mathrm{c}} = (x_{\mathrm{c},1} \, . \, . \, . \, x_{\mathrm{c},p})$ which describes the additional tasks to be fulfilled in addition the m-dimensional end-effector task vector $\boldsymbol{x}_{\mathrm{t}}$, i. e., the degree of redundancy $p = n - m$, whereas n is the number of joints (degrees of freedom-DoFs).

Redundant manipulators possess extra DoFs than those required to perform the main task(s). These additional DoFs can be used to fulfil user-defined additional task(s).

The additional task(s) can be represented as kinematic functions. The kinematic functions reflect some desirable kinematic characteristics of the manipulator such as posture control, joint limiting and obstacle avoidance. The kinematic functions can be extended to include dynamic measures of the performances.

The kinematic functions may be also defined as the robot configuration-dependent terms in the manipulator model.

The relation between the joint-space coordinate vector $\boldsymbol{q}$ and the constraint-task vector $\boldsymbol{x}_{\mathrm{c}}$ - due the presence of obstacles- can be considered as a direct kinematics equation:

$$\boldsymbol{x}_{\mathrm{c}} = \boldsymbol{k}_{\mathrm{c}}(\boldsymbol{q}) \tag{7}$$

where $\boldsymbol{k}_{\mathrm{c}}$ is a continuous nonlinear vector function.

By differentiating (7) one can be obtained (8):

$$\dot{\boldsymbol{x}}_{\mathrm{c}} = \boldsymbol{J}_{\mathrm{c}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{8}$$

In (8) $\dot{\boldsymbol{x}}_{\mathrm{c}}$ is the constraint-task velocity vector, and the mapping $\boldsymbol{J}_{\mathrm{c}}(\boldsymbol{q}) = \partial\boldsymbol{k}_{\mathrm{c}}/\partial\boldsymbol{q}$ is the (p x n) constraint-task Jacobean matrix. At this moment, an augmented-task vector $\boldsymbol{x}_{\mathrm{a}}$, can be defined by stacking the end-effector task vector with the constraint-task vector as:

$$\boldsymbol{x}_{\mathrm{a}} = [\,\boldsymbol{x}_{\mathrm{t}} \quad \boldsymbol{x}_{\mathrm{c}}\,]^{\mathrm{T}} = [\,\boldsymbol{k}_{\mathrm{t}}(\boldsymbol{q}) \quad \boldsymbol{k}_{\mathrm{c}}(\boldsymbol{q})\,]^{\mathrm{T}} \tag{9}$$

According to this definition [3], finding a joint configuration $\boldsymbol{q}$ that result, in some desired value for $\boldsymbol{x}_{\mathrm{a}}$, means satisfying both the end-effector task and the constraint task, at the same time.

At the differential level, Cartesian velocities are given by:

$$\dot{\boldsymbol{x}}_{\mathrm{a}} = \boldsymbol{J}_{\mathrm{a}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{10}$$

To finding a joint velocity $\dot{\boldsymbol{q}}$ in some desired value for Cartesian velocities $\dot{\boldsymbol{x}}_{\mathrm{a}}$ can be used. In this control method, each joint velocity is computed as:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_{\mathrm{a}}^{+}\dot{\boldsymbol{x}}_{\mathrm{a}} + (\boldsymbol{I} - \boldsymbol{J}_{\mathrm{a}}^{+}\boldsymbol{J}_{\mathrm{a}})\dot{\boldsymbol{q}}_0 \tag{11}$$

That means the inverting of the augmented Jacobean, $\boldsymbol{J}_{\mathrm{a}}(\boldsymbol{q}) = [\,\boldsymbol{J}_{\mathrm{t}}(\boldsymbol{q}) \quad \boldsymbol{J}_{\mathrm{c}}(\boldsymbol{q})\,]^{\mathrm{T}}$ to obtain the pseudo-inverse matrix, $\boldsymbol{J}_{\mathrm{a}}^{+}$.

5. Additional tasks for exploiting the self motion in the joint axis direction

One supposes that, the collision avoidance takes over control for only one degree-of-freedom (specifically, in joint axis direction, as example y) in order to avoid the closest obstacle. This means that the collision avoidance uses one degree of freedom for the robotic arm.

Collision avoidance does not affect other degrees of freedom, which remains available for the task execution (e.g., target reaching motions related to the x and z direction).

The origin of the coordinate system is the closest point on the flanking obstacle. The y-axis is aligned to a line that connects the closest points so that the direction of avoidance is aligned to this axis. The *x*-axis is aligned to the vector that extends from the robot to the target position. The *z*-axis is an outer product of a unit vector of *x* direction and a unit vector in *y* direction. The collision avoidance joint moves only in the *y* direction on this coordinate system.

In this control method, each joint velocity is computed based on the row vector extracted from the collision avoidance pseudo inverse Jacobean matrix, $\boldsymbol{J}_{ca}(\boldsymbol{q})$ and which is defined in the joint axis (*y* axis for example) in the collision avoidance coordinate system:

$$\dot{\boldsymbol{q}}_{ca} = \boldsymbol{J}_{ca}{}^{+}\dot{\boldsymbol{y}}_{ca} + (\boldsymbol{I} - \boldsymbol{J}_{ca}{}^{+}\boldsymbol{J}_{ca})\dot{\boldsymbol{q}}_0 \,, \tag{12}$$

where $\boldsymbol{J}_{ca}(\boldsymbol{q})$ is the collision avoidance Jacobean between closest points.

The row vector is an avoidance velocity vector which is derived from a virtual force. The physical limitations corresponding of velocity-space constraints are maximum rotational and translational velocities.

A target (defined externally or by the robot itself) is supplied to a motion control method such as a whole robot motion (WRM) control method [5] and a collision avoidance (CA) method. The whole robot motion is to be understood as being simply an example for a motion in the task space.

The WBM control method yields a first joint velocity vector that is combined with a second joint velocity vector from the CA method. The blending control method yields a combined joint velocity vector based on which the robot's motion is controlled.

The combined joint velocity vector is furthermore provided to a distance computing method that calculates the two closest points of different segments of the robot connected to each other via at least one joint or a segment of the robot and another obstacle.

The distance computing method yields closest point data and distance information necessary to the CA control method. With the blending method will bee calculated the blending ratio between the first and the second joint velocity vector on the basis of the obstacle distance information provided.

6. Conclusions

In this paper the authors have considered a redundant inverse kinematics model based on the augmented Jacobean. This model enables to use of self motion of the manipulator to perform additional task for obstacles avoidance, based on null space criteria.

By operating of the obstacle avoidance strategy in velocity space one can simultaneously control the speed and route of the robot elements, and can come up with solutions that correspond directly to control the robot.

By treating the local obstacle avoidance problem as one of constrained optimization, one can easily incorporate constraints from the environment and robot dynamics, and can come up with formulations that trade off speed for safety.

Advantages of this formulation include the ability to simultaneously control the speed and route of the robot.

References

1. Yared, R. et al. (2007) *Locality-preserving distributed path reservation protocol for asynchronous cooperative mobile robots.* Proceeding of the 7th IEEE International Symposium on Autonomous Decentralized Systems (ISADS’07), ISBN: 0-7695-2804-X, p. 188-195
2. Siciliano, B., Khatib, O. (2008) *Handbook of Robotics*. ISBN: 978-3-540-23957-4, Springer-Verlag Berlin, Heidelberg
3. Van den Berg, J., Lin, M. Manocha, D. (2008) *Reciprocal Velocity Obstacles for real-time multi-agent navigation*. Proceeding of the IEEE Int. Conf. on Robotics and Automation, p. 1928-1935
4. Jiang, R., Tian, X., Xie, L., Chen, Y. (2008) *A Robot Collision Avoidance Scheme Based on the Moving Obstacle Motion Prediction.* Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management, ISBN: 978-0-7695-3290-5**,** Vol. 2, p.341-345, 2008
5. Sugiura, H., Janssen, H., Gienger, M. (2008) *Robots with collision avoidance functionality.* US Patent application 2008 / 0234864 A1

Received in February 2012