Boxing Humanoid Robot

Jin Han Lee, Carlos Nieto, Michael Novitzky

***Abstract*—In 2008, Nao robots played soccer for the first time in Robocup in Suzhou, China. This robots are developed to play soccer and for entertainment (e.g. dancing, interacting with children). This report presents that using different motion planning methods (e.g. inverse kinematics, RRTs, potential fields), allows this robot to do more complicated tasks like boxing. The main challenge is to deal with the constraints of the robot's hardware to accomplish fighting tasks such as punching as faster the robot can to his opponent.**

***Index Terms*—motion planning, boxing, Nao robot.**

I. Introduction

BOXING is one of the simplest and oldest combat sports where two participants fight each other with their fists until one of them is knocked unconscious. Is supervised by a referee and is typically engaged in during a series of one to three-minute intervals called rounds. Victory is achieved if the opponent is knocked down and unable to get up before the referee counts to ten seconds (a Knockout, or KO) or if the opponent is deemed too injured to continue (a Technical Knockout, or TKO). If there is no stoppage of the fight before an agreed number of rounds, a winner is determined either by the referee's decision or by judges' scorecards. [1]

Over the last years, In the last 6 years, fighting robots has become one of the most popular activities for hobbyist roboticist in the world. This work addresses the problem to change a robot designed to play soccer into a boxing robot.

A. Physics behind boxing: Contact with the Goal

The fist has its maximum velocity when it hits something[2]. This collision causes the fist to slow down, and eventually when the fighter begins applying a force to retract his arm, the fist stops and returns to the fighter. This speed is calculated using; $Velocity = \frac{distance}{time}$. As you can see, there are two ways to make a fist go faster:

1) By lengthening the distance or shortening the time. The distance can be lengthened to a maximum of the fighter's arm length, but the time will depend on training, and the acceleration ($acceleration = \frac{velocity}{time}$) of the arm.
2) Calculating the momentum and impulse of the arm and fist of the fighter where the momentum *(P)* can be seen as an object's tendency to resist change in acceleration $P = m * v$. An impulse (I) is the change in momentum of a certain object.

$$I = \int F dt \tag{1}$$

$$I = \int dp \tag{2}$$

$$I = \triangle p \tag{3}$$

B. Punch steps

1) Before the fist makes contact with the face, it has a certain momentum, and a stationary head would have zero momentum.
2) During the contact, there is a transfer of momentum from the fist and arm to the head of the opponent.
3) Although momentum is conserved when looking at both boxers, just looking at the person taking hit, his/her momentum has changed from zero, to what ever momentum was transferred from the fist.

A faster punch can be more effective because with the mass of the fist being constant, by increasing velocity, the momentum that the punch carries is larger, hence, but change in momentum that the opponent's head experiences increases.

II. Related Work

In [3], Moissec et al. proposed software that be used to generate an optimal trajectory kicking motion for the Humanoid robot HPR-2. Using a dynamic model of the robot incluiding static friction and the hardware constraints of the motors and the reduction ratio, they calculate an optimal motion obtained and some characteristics of the process of motion generation.

III. Motion planning for robot boxing

A. Collision Detection

Collision detection was performed by the V-COLLIDE: Accelerated Collision Detection for VRML library and implemented in the RST framework. With each iteration, the planner is given an object collision list. A function was designed to detect either a collision with the defending head or the defending torso and one of either the attacking limb types, either the arm of the robot starting from the elbow to the tip or the leg consisting of the shin and foot.

B. Forward Kinematics

Forward kinematics was calculated using Mathematica for each joint starting from the base of where each limb attached to the torso. Thus, each arm began at the shoulder joint and each leg began at the hip joint. The forward kinematic transformation matrices were hard coded into the planner using the full simplify equations. The planner then only required the current joint angle configuration in order to estimate the current cartesian coordinates.

College of Computing
School of Interactive Computing
Georgia Institute of Technology

C. Inverse Kinematics

The inverse kinematics [4], [5]were calculated using an underconstrained Jacobian method. Underconstrained because only the cartesian location was calculated and not the orientation of the end effector. Utilizing Mathematica[1], each Jacobian matrix was calculated by differentiating each x,y and z equation with respect to each joint from the forward kinematic transformation matrices. The Jacobian matrix was then hard coded into the system and only required input of the angle configuration for each end effector to begin the inverse kinematics. Because of system limitations the psuedo-inverse was used calculated using Single Value Decomposition utilizing the GSL[2] mathematics library.

It is known that this is not the most efficient method to calculate the inverse Jacobian yet the researchers felt that continuing progress into other domains for the planner was the most effecient use of the limited time. In order to minimize singularities the end effectors were limited from reaching joint values that would cause problems by simply not executing those joint commands. Additionally, if the inverse Jacobian method resulted in joint commands that went beyond their limits they were also ignored yet those commands that did not violate joint limitations were allowed to pass. This ensured that forward progress continued and the end effector would not get stuck in a singularity and kept the system more realistic. The inverse kinematics were used for all of the motions of the Nao humanoid.

The most basic implementation of our planner simply required the end effector location, determined with forward kinematics, and the target limb centroid such as the head or the torso. The planner would then iterate in small deltas in cartesian coordinates, 0.0005, in a straight line from end effector start location to goal location until either the end effector ran out of 5,000 iterations, or a collision between the end effector and the target limb was detected.

$$xdiff = target_x - attackLimb_x \tag{4}$$

$$ydiff = target_y - attackLimb_y \tag{5}$$

$$zdiff = target_z - attackLimb_z \tag{6}$$

$$distance = \sqrt{xdiff^2 + ydiff^2 + zdiff^2} \tag{7}$$

$$\triangle_x = (xdiff/distance) * jacobianStepSize \tag{8}$$

$$\triangle_y = (ydiff/distance) * jacobianStepSize \tag{9}$$

$$\triangle_z = (zdiff/distance) * jacobianStepSize \tag{10}$$

[1]Wolfram Research, Inc., Mathematica, Version 6.1, Champaign, IL (2008).

[2]The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.

D. Potential Fields

A goal of this project is to have an attacking robot score points against a defending robot, obstacles were determined to be the arms of the defending robot Thus, potential fields were utilized for obstacle avoidance. This method allows the planner to avoid a defender's blocking arms. In order to simplify the problem a potential field is created only using the XZ-plane (Figure 1) of the world using the Y value of the attacking end effector to make a slice of the world.

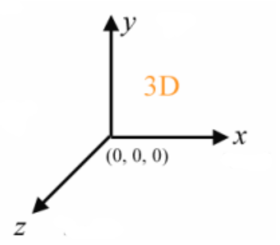


Figure 1. 3D World Axis.

Each end effector was determined to be a 3D line in the world starting from the elbow joint to the end effector. If a plane intersects with a defending arm's line segment then an obstacle point is created. An obstacle point has a radius of 0.08 and the repelling force from within the radius of $-cos\Theta$ or $-\sin\Theta$ depending whether it was the z or x axis calculation.

This first radius (Figure 2), is used to repel an end effector with a great amount of force becuse if a blocking and attacking end effector were within 0.08 of their centroids then a collision is sure to happen as it is an estimate of the widest part of two arms side by side.

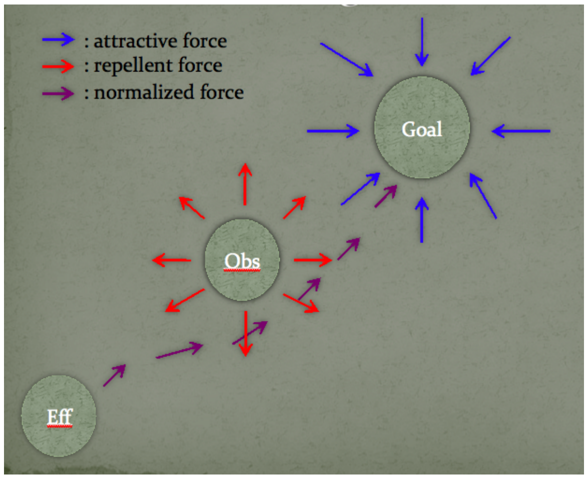


Figure 2. Potential Field

A second sphere around an obstacle is created to gently guide an attacking end effector around a blocking arm. Thus, if an attacking arm was within 0.10 or 0.08 then a repelling force R as calculated in Eq 11 is placed on the end effector trajectory.

$$R = -1/distance^2 * obstacle\Theta \tag{11}$$

As an attacking arm gets closer and closer to the defending arm the repelling force grows more and more until it actually makes contact with the blocking arm and is repelled with a great force. A target limb centroid is projected onto the XZ-plane regardless of its Y height in the world which means the end effector is constantly drawn to the target with a force of $cos(goalTheta)$ or $sin(goalTheta)$.

$$\triangle X = \frac{\triangle X_{obstacle} + \triangle X_{goal}}{\sqrt{\triangle X^2 + \triangle Z^2}} \qquad (12)$$

$$\triangle Z = \frac{\triangle Z_{obstacle} + \triangle Z_{goal}}{\sqrt{\triangle X^2 + \triangle Z^2}} \qquad (13)$$

The $\triangle Y$ is calculated exaclty the same as in the methods above for the straight line jacobian method. Thus, at each time step, obstacle and goal points are created on an XZ-plane slice. The forces are summed as vectors and these x and z deltas are then input into the Jacobian to perform inverse kinematics. This allows an attacking robot limb to avoid a blocking arm and still score a point.

E. Maximizing Impact of Punching

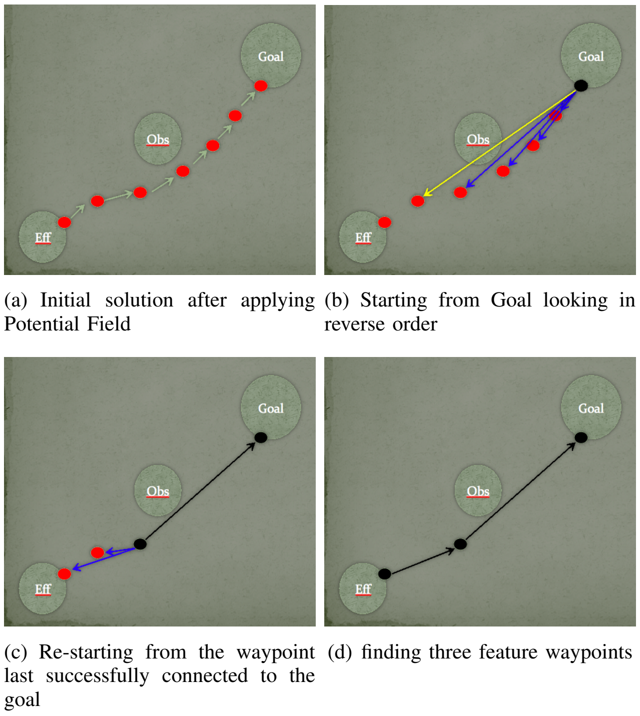
As one of our goal through this paper is to deliver possible maximum force to an opponent through maximizing momentum of a moving end effector at the contact to the opponent. Solutions generated from the previous section using IK alone and IK with Potential Field contains too many waypoints due to small jacobian step size Δx. Executing such a solution fails to maximize the impact of motion on a target. In this section, we propose to meet the need of our goal a post processing step taking two steps: *extracting feature waypoints among many waypoints* and *interpolating between feature waypoints*.

Reducing the number of waypoints is important due to two main factors. Firstly, it is not efficient for robot controllers to follow fine grained waypoints. Controlling with high accuracy is costly expensive. Secondly, there is no much of space to increase the momentum of the end effector. A controller should handle a few waypoints along the path trajectory so that it can accelerate the end effector by properly interpolating between waypoints.

F. Extracting Feature Waypoints

Under the constraint that the end effector should avoid obstacles such as an opponent's blocking arms and legs, we define feature waypoints as the minimum number of waypoints that should be kept to avoid obstacles along the path trajectory. The method for extracting feature waypoints works as shown in Figure 3.

We looked through the generated path in reverse order from the goal. Initially, the goal waypoint is pushed onto a stack and is labeled as the reference waypoints. We retrieve the next waypoint and check if a robot can move from the reference waypoint to the retrieved waypoint without collisions. If so, then we retrieve the next one and perform the same reference to retrieved waypoint collision checking. We continue doing this procedure until a collision occurs, at which point we push back on the stack the last retrieved waypoint, which was



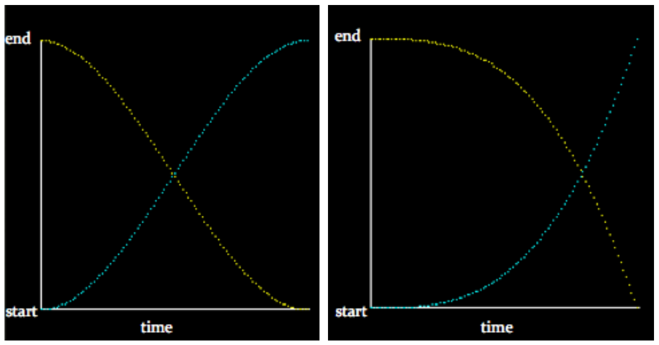
(a) Initial solution after applying Potential Field
(b) Starting from Goal looking in reverse order
(c) Re-starting from the waypoint last successfully connected to the goal
(d) finding three feature waypoints

Figure 3. Feature Waypoints Extraction

successfully extended from the reference waypoint without collisions. At this point, we treat the last waypoint pushed onto the stack as our reference waypoint. We keep iterating this procedure until the retrieved waypoint is the initial waypoint. After finishing this procedure, waypoints in the stack is actually feature waypoints.

G. Interpolating between Waypoints

Once obtaining feature waypoints through the previous section, interpolation between those waypoints are necessary to control momentum of the moving end effector. We use two different cubic Bezier curves, curve type A and curve type B, as shown in Figure 4, for interpolation between two waypoints.



(a) Type A. Accelerate fast and slow down at end
(b) Type B. Accelerate slowly and maximize at end

Figure 4. Cubic Bezier Curves

We chooses a simple strategy to choose which curve type is used for interpolation. Curve type A is used to reach from one waypoint to other in such a stable manner that it reduces the momentum of the end effector. Curve type B is used to cover the opposite case, maximizing the momentum. We uses curve type B for every interpolation except for interpolation between last two waypoints. The final planned motion is as shown in Figure 5. We emprically found two control points for each curve type.

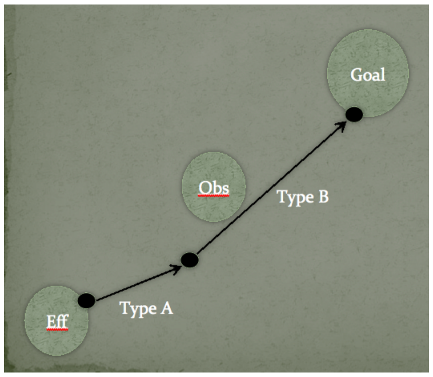


Figure 5. Final Planned Motion

Once deciding a curve type for interpolation between feature waypoints, we calculate how much time should be to taken to execute each interpolated motion between two waypoints. To make our interpolation realistic, we put a constraint that robot's each joint motor has the maximum velocity limitation, 10 degree per 20ms in our case. This means that if any of robot joints tries to exceed that limitation, it need to be sub-interpolated. We make sure that this does not happen by doing linearly re-interpolated.

H. Step and Strike

The ultimate goal of this project was to optimize the momentum of an attacker's punch. A key idea in this approach is that if we can increase the mass and velocity that we will increase the momentum. The next step after accelerating just an end effector during the motion plan is to include the movement of the whole body during a strike. This was partially achieved through keypoint positions of the Center of Mass (COM) of the torso throughout a stepping motion and a punch being executed as the robot transferred its COM from one leg to another in a forward motion. Unfortunately, due to time constraints the stepping motion followed by a punch was only achieved using the inverse Jacobian method without acceleration. Thus, testing of increased acceleration was not possible and left as future work.

IV. Experiments

All of the implementation and testing of our system was performed in the RST framework provided by Dr. Stilman. In order to test the effectiveness of our acceleration methodology we tested the straight punch and the obstacle avoidance punch using the potential field without any acceleration. The next step was then to test both the straight punch and the obstacle avoiding punch using our acceleration method which removed the unnecessary waypoints and interpolated the keyway points.

As shown in the table I, it is clear that only using the Jacobian method and a very small step size of 0.0005 results in very weak momentum upon impact with the straight unaccelerated punch producing 6.194 g*m/s and the unaccelerated potential field punch producing 6.538 g*m/s. This small step size significantly reduces the velocity of the arm links from one step to another.

The small Jacobian step size is utilized because a criteria of the planner is that it produce stable waypoints along the path to the target. It is not difficult to predict that making the step size larger will increase the momentum yet it would lead to a greater chance of instability. However, removing unnecessary waypoints after a stable path has been created is a logical next step in increasing the momentum of an attacking end effector. Once unnecessary waypoints are removed, the final step in the procedure is the interpolation of the required waypoints so that the joint maximum velocity of 10^o every 20ms is not violated thus keeping the integrity of the system intact. As can be seen by the results, the removal of unnecessary waypoints and the interpolation of the trajectory in order to reach the maximum velocity does increase the momentum producing 33.1979 g*m/s for the straight punch and 41.6244 g*m/s for the potential field driven punch.

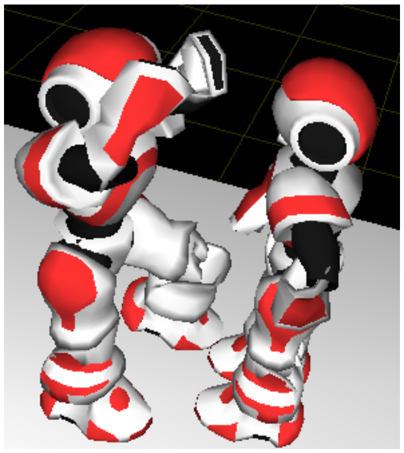


Figure 6. Nao Boxing Simulation

There remains much room for improvement of this planner. First of all, this planner is not complete because it has a maximum Jacobian iteration value of 1,000. This means that a solution may exist for an end effector to reach its target yet the planner will stop iterating and return a failure. It is difficult to determine if our algorithm is optimal. Based on our momentum maximization requirement, true momentum maximization will be achieved if more distance can be created by the planner between the attacking end effector and the target.

However, this implementation only searches for solutions decreasing the distance to the target thus making it not optimal. This algorithm is also not efficient in that it uses a very small jacobian step size of 0.0005 in order to increase the stability of the solutions. A more efficient manner would be to analyze the stability of the inverse Jacobian given a large step size and then reduce the step size until a stable inverse Jacobian is found thus reducing the number of steps required to find a solution. Additionally, using LU Decomposition would be a more efficient manner of solving the inverse Jacobian. Clearly, there is room for much improvement in the implementation of this planner.

Punching Motion	Momentum (kg*m/s)	# waypoints
Straight	0.00619446	136
Straight w/Bezier Curve	0.0331979	2
Straight w/PF	0.000650383	351
Straight w/Bezier Curve and PF	0.0416244	6

Table I
COMPARISON OF MOMENTUM

V. DISCUSSION

In order to implement to make a robot capable of boxing, capability of avoiding obstacles and increasing impact of punching are necessary. Benefits of using potential field for obstacle avoidance are many folds. Two dimensional potential field we implemented works in continuous working space so that there is no need of discretizing the space and computationally cheap. The well known problem of the potential field is being fallen into local minima. In our case, during punching the end effector does not move forward and backward between obstacles.

If we solved inverse kinematics in the analytical way, we could have used RRT (Rapidly Exploring Randomized Tree) to implement obstacle avoidance. A given target pose in the working space is transformed and represented in joint space using IK. By using RRTs we think that the solution to the target could be generated faster than our current method.

As a future work, we would like to attach any of physics engines to the simulation so that dynamics of a robot will be taken into account to generate punching and walking motions. Currently the user controls attacking limb and target. In addition, we would like to extend our implemention such that the user only commands a target and the planner chooses which attacking limb to use and which target point to hit.

REFERENCES

[1] "Boxing," 2008, encyclopaedia Britannica.

[2] "Physics behind boxing," http://class.phys.psu.edu/p001projects/.

[3] K. K. A. Miossec, S. Yokoi, "Development of a software for motion optimization of robots - application to the kick motion of the hrp-2 robot," in *IEEE International Conference on Robotics and Biomimetics*, Kunming, China, 2006.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, June 2005, iSBN 0-262-03327-5.

[5] M. V. Mark W. Spong, S. Hutchinson, *Robot Modeling and Control*. John Wiley and Sons, Inc., 2005.