# Reflexive Halting for a Mobile Industrial Robot

A key safety feature of many robots is reflexive halting. Reflexive halting uses sensors to detect obstacles in the robot's path and prevent motion that would result in a collision. Reflexive halting is particularly important for robots that operate in shared environments with people, because people can suddenly move into a previously clear robot path, endangering themselves and the robot. Whereas a planner will create safe paths for a robot around static obstacles, paths may be invalidated by moving obstacles such as people, and the planner may react too slowly to prevent a collision. Reflexive halting operates more quickly and on a lower level than planning and trajectory generation, and can override the velocity commands from the trajectory generator.

## Reflexive Halt Methods for Mobile Bases

Mobile robots often implement a reflexive halt as shown in ALGORITHM. For example, if a measurement source such as a LIDAR reports that there is an obstacle in front of the robot at close range, the robot's velocity would be limited to turning and reversing. This approach is effective, but can cause difficulties in navigating tight areas, where there may be enough space for a robot to navigate through obstacles if it does so carefully.

Given:  
 Sensor measurements, M  
Do:  
 for each measurement m in M:  
 if(dangerous(m)):  
 prevent motion in direction of m

Another approach to reflexive halting was described by Chad Rockey in his masters thesis [CITE ROCKEY]. This approach, called Reflexive Avoidance Plus, was developed for a smart wheelchair, which must operate in crowded areas around people. Reflexive Avoidance Plus uses velocity limiting rather than preventing motion altogether. When a sensor detects an obstacle in the robot's path, it limits the maximum velocity in that direction using a scaling function based on the distance of the obstacle from the robot. This approach prevents the robot from colliding with an obstacle (the maximum velocity is zero below a certain threshold distance), but allows low speed progress toward an obstacle. Because the robot approaches the obstacle at lower speed, it can safely get closer to obstacles.

# Reflexive Halting for Manipulators

In addition to the mobile base, ABBY's robotic arm poses a risk for collisions. In industrial situations, manipulators are kept inside safety cages to prevent people from interfering with them or getting injured. A safety cage is not a possibility for a mobile manipulator, so another solution is necessary. A reflexive halt for the manipulator allows it to operate safely in the presence of people and obstacles without a safety cage.

Like the mobile base planner, the planner for the arm generates collision-free paths. However, the planner for the arm does not replan at all once it commits to a trajectory. The trajectory is generated and sent to the IRC5 for execution, and then ROS waits for the trajectory to be executed. If something enters the path of the trajectory, the robot does not alter the current trajectory. This makes the robot unsafe to operate around humans above a certain joint speed.

Rethink Robotics, with their robot Baxter, [CITE BAXTER] solved the problem of operating an industrial robot without a safety cage with a mechanical and software solution that relied on force feedback and serial-elastic actuators. Because all of Baxter's joints are elastic and its arms are so light, it can safely collide with people and obstacles. It also uses force-feedback in its joints to detect these collisions and become passive, allowing people to push it around. Because ABBY's robotic arm does not have serial elastic actuators or force feedback in the joints, this solution is not possible.

Using the Kinect sensor, it would be possible to implement one of several possible reflexive collision avoidance methods on ABBY. One method would be to halt arm motion when an obstacle enters the arm's work envelope and suspend arm motion until the obstacle leaves the work envelope. Although this is arguably the safest solution, it can cause the robot to get stuck. If an inanimate obstacle is brought into the robot's work envelope and left there, the robot will never re-enable the arm.

To resolve this problem, the reflexive halt behavior can be augmented as shown in ALGORITHM. In this algorithm, the currently planned path is repeatedly checked for collisions until execution is completed. If an object enters the path, the robot stops execution of the trajectory and waits. If the obstacle leaves the arm's path before a timeout is reached, the robot resumes execution of the trajectory. If the obstacle does not move, the robot will attempt to retry planning to move around the obstacle to accomplish its goal. This reflexive halt algorithm has two advantages over the naïve algorithm described in the previous paragraph. First, it does not stop for obstacles that enter the work envelope but do not interfere with the planned motion. This allows humans to work alongside the robot and interact with it by giving it objects or taking objects from it. Second, it will plan around stationary objects that enter the work envelope, allowing a person to leave an object in the work envelope without stalling the robot.

Given:  
 Current Plan P, Measurements M  
Do:  
 for each state s in P, measurement m in M:  
 if(dangerous(m, s)):  
 halt  
 wait for obstacle to move or replan

# Recommendations for This Robot