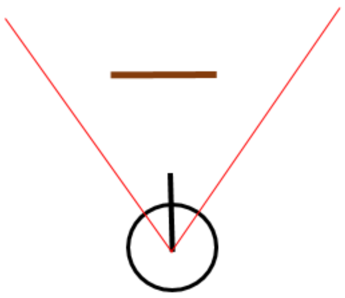
Report

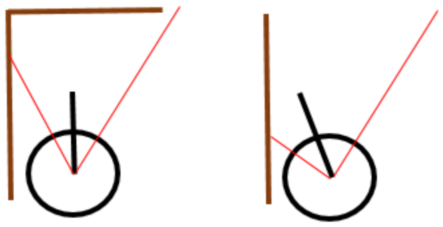
Strategy for Lidar_Alarm:

In lidar_alarm.cpp, I basically divided the situation into 3 normal cases and 2 special cases, which are:

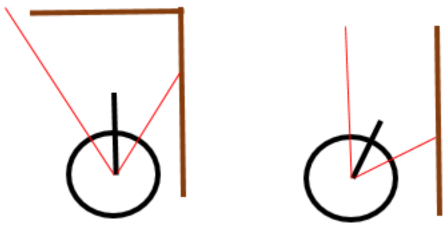
Front:



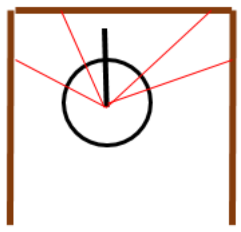
Left:



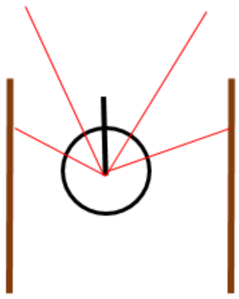
Right:



Surrounded:



Tunnel:



Hence, I group the pings in three parts:

```
ping_min = (int) (-1.00-angle_min_)/angle_increment_;        //first ping
ping_max = (int) (1.00 - angle_min_)/angle_increment_;        //last ping
ping_left = (int) (0.30-angle_min_)/angle_increment_; //left threshold
ping_right = (int) (-0.10-angle_min_)/angle_increment_; //right threshold
```

And then, for the front case, as long as there is any ping gives warning, the robot will get an alarm, which is published by “lidar_alarm_publisher_1”:

```
//front
        for(ping_index_=ping_right;ping_index_<ping_left;ping_index_++){

               ping_dist = laser_scan.ranges[ping_index_];
               if(ping_dist <SAFE_DISTANCE){

                    front = true;        //if there is one ping gives warning then count
                    break;      //obstacles in front
                    }
               else{
                    front = false;
                        }
             }
```

```
//case1
    if(front &&!left && !right){
        ROS_WARN("DANGER,SOMETHING IN FRONT,JUST TURN LEFT     ");
        laser_alarm_1=true;    // notice lidar and laser
        lidar_alarm_msg1.data = laser_alarm_1;
        lidar_alarm_publisher_1.publish(lidar_alarm_msg1);
    }
     else{
     laser_alarm_1 = false;
     lidar_alarm_msg1.data = laser_alarm_1;
     lidar_alarm_publisher_1.publish(lidar_alarm_msg1);
      }
```

For the left case, if the number of pings that give warnings is less than a certain amount, then it may be regarded as a misjudgment and the warnings will be dismissed:

```
//left
     for(ping_index_=ping_left;ping_index_<ping_max;ping_index_++){
          ping_dist = laser_scan.ranges[ping_index_];
          if(ping_dist<LEFT_SAFE_DISTANCE){
              counter+=1;
            }
        }

     if(counter>20){
        left = true;
       }
     else{
        left = false;
     }
     counter = 0;    //clear counter

//case2
    if(left && !right){
        ROS_WARN("DANGER,TRUN RIGHT");
        laser_alarm_2=true;
        lidar_alarm_msg2.data = laser_alarm_2;
        lidar_alarm_publisher_2.publish(lidar_alarm_msg2);
    }
    else{
        laser_alarm_2 = false;
```

```
            lidar_alarm_msg2.data = laser_alarm_2;
            lidar_alarm_publisher_2.publish(lidar_alarm_msg2);
         }
```

so is the right case.

For case 4, which is to warn the robot that it is being closely surrounded by obstacles and let the robot turn around:

```
//case4
        if(front && left && right){
            ROS_WARN("DANGER,TRUN AROUND");
            laser_alarm_4=true;
            lidar_alarm_msg4.data = laser_alarm_4;
            lidar_alarm_publisher_4.publish(lidar_alarm_msg4);
        }
        else{
            laser_alarm_4 = false;
            lidar_alarm_msg4.data = laser_alarm_4;
            lidar_alarm_publisher_4.publish(lidar_alarm_msg4);
        }
```

And finally, case 5 gives a warning of being in a tunnel, this is just to deal with the "! front && left &&right " situation, which will let the robot slow down:

```
//case5:false alarm
        if(!front && left && right){
            ROS_WARN("IN TUNNEL");
            laser_alarm_5=true;
            lidar_alarm_msg5.data = laser_alarm_5;
            lidar_alarm_publisher_5.publish(lidar_alarm_msg5);
        }
        else{
            laser_alarm_5 = false;
            lidar_alarm_msg5.data = laser_alarm_5;
            lidar_alarm_publisher_5.publish(lidar_alarm_msg5);
        }
}
```

Strategy for Reactive_commander:

There are 5 subscribers for the corresponding cases:

In case 1, when meeting an obstacle in front, I choose to let robot turn left, because turn back may not be nice choice. And in compensate of case 3, which also warns the robot to turn left, the index of “ping_right” (the threshold ping from case 3 to case 1) will increase.

```
// case1
twist_cmd.linear.x=0.0; //stop moving forward
twist_cmd.angular.z=yaw_rate; //and start spinning in place
timer=0.0; //reset the timer
while(g_lidar_alarm1 && !g_lidar_alarm2 && !g_lidar_alarm3 && !g_lidar_alarm4 && !g_lidar_alarm5) {
ROS_INFO("Under Case 1 !");
twist_commander.publish(twist_cmd);
timer+=sample_dt;
ros::spinOnce();
loop_timer.sleep();

            }
```

When receiving g_lidar_alarm2, the robot will turn right. However, in some situations, when the next alarm or the former alarm that the robot receive is case 1, because of the opposite turning direction and parameters of judgment, the robot will keep swinging and not easily get out of this status. I think the easiest way to do here is to give extra time for the turning so it will pass the next alarm and get rid of swinging.

```
twist_cmd.linear.x=0.0; //stop moving forward
twist_cmd.angular.z=-yaw_rate; //and start spinning in place
timer=0.0; //reset the timer
while(!g_lidar_alarm1 && g_lidar_alarm2 && !g_lidar_alarm3 && !g_lidar_alarm4 && !g_lidar_alarm5) { // case2
ROS_INFO("Under Case 2 !");
    while(timer<time_sec){
        twist_commander.publish(twist_cmd);
        timer+=sample_dt;
        loop_timer.sleep();
          }
   ros::spinOnce();
  }
```

And finally, for special case 4 the robot will turn around:

```
          twist_cmd.linear.x=0.0; //stop moving forward
          twist_cmd.angular.z=yaw_rate; //and start spinning in place
          timer=0.0; //reset the timer
          while(!g_lidar_alarm1 && !g_lidar_alarm2 && !g_lidar_alarm3 && g_lidar_alarm4
&& !g_lidar_alarm5) { // case4
            ROS_INFO("Under Case 4 !");
            while(timer<2.6){
            twist_commander.publish(twist_cmd);
            timer+=sample_dt;
            loop_timer.sleep();
              }
            ros::spinOnce();
            }
```

For case 5, the robot will slow down:

```
          twist_cmd.angular.z=0.0; // do not spin
          twist_cmd.linear.x=low_speed; //command to move forward slowly
          timer=0.0;
          while(!g_lidar_alarm1 && !g_lidar_alarm2 && !g_lidar_alarm3 && !g_lidar_alarm4
&& g_lidar_alarm5) {
            ROS_INFO("Under Case 5 !");
            twist_commander.publish(twist_cmd);
            timer+=sample_dt;
            ros::spinOnce();
            loop_timer.sleep();
            }
```

Some problems

Sometimes under the same parameters, the trajectory of the robot can be different which will make the tuning difficult, and sometimes the robot will turn more angles than the supposed angle.