1. qt_vision_turtlebot3_tracing

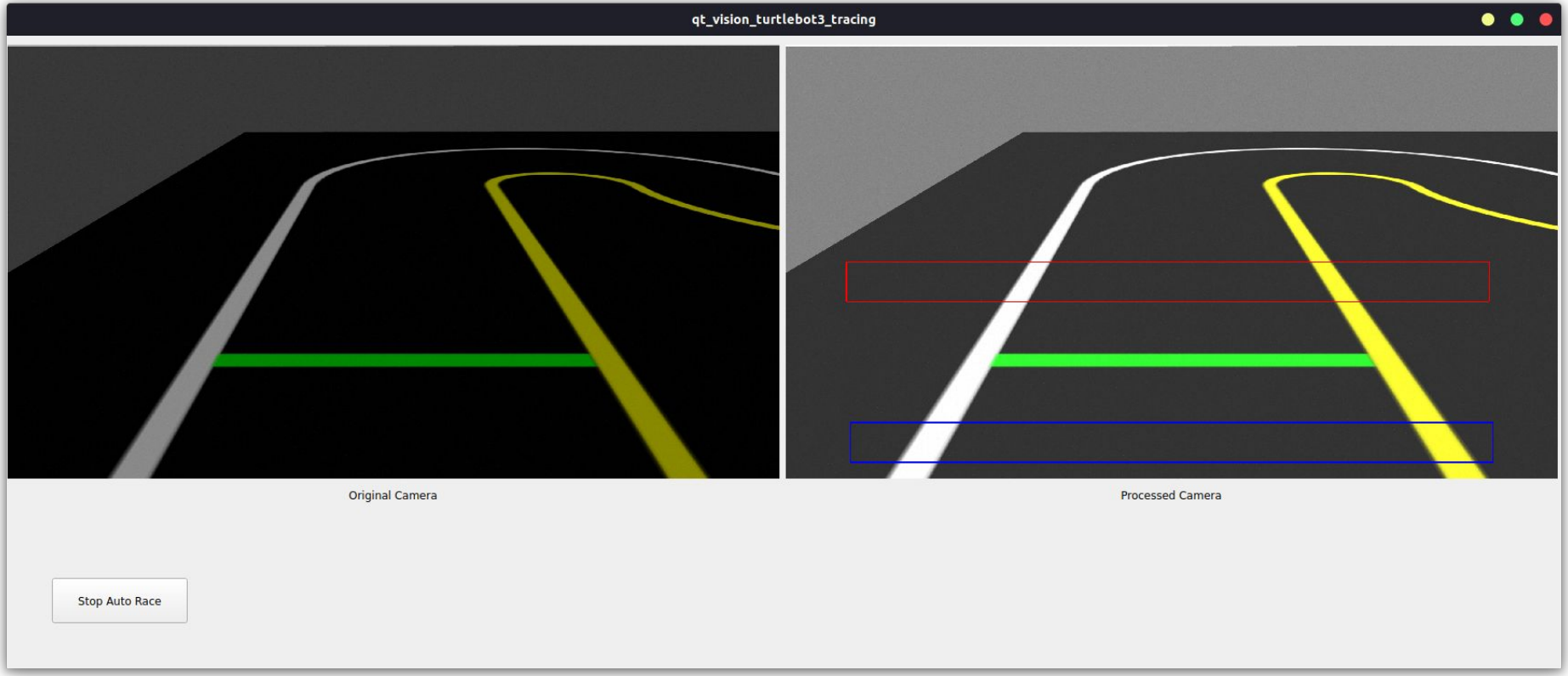
Main Algorithm

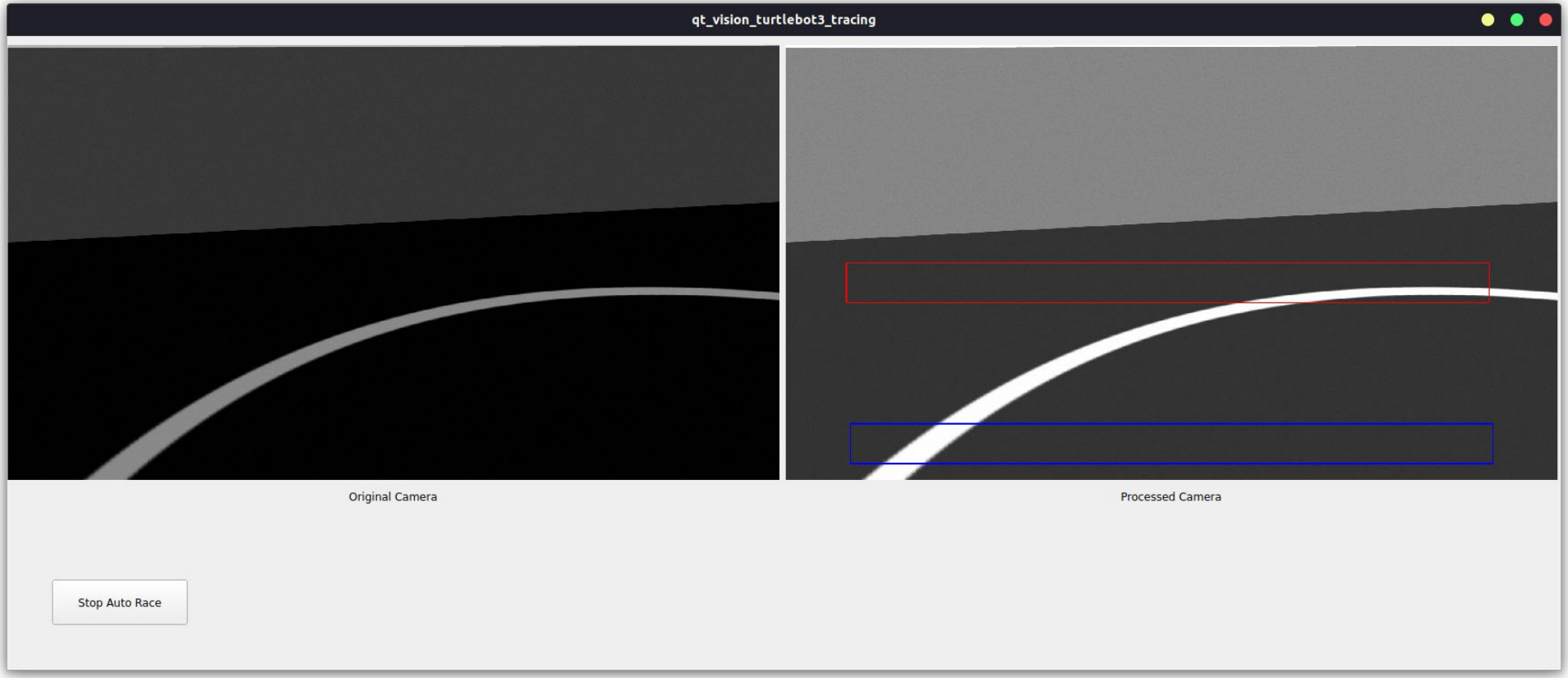
1. /camera/image_raw 토픽으로 이미지 불러오기
2. QPixmap 처리로 Qt GUI에 출력하기
3. 주행 버튼 클릭 시 -> 감지 영역 설정하기(main_detect, sub_detect)

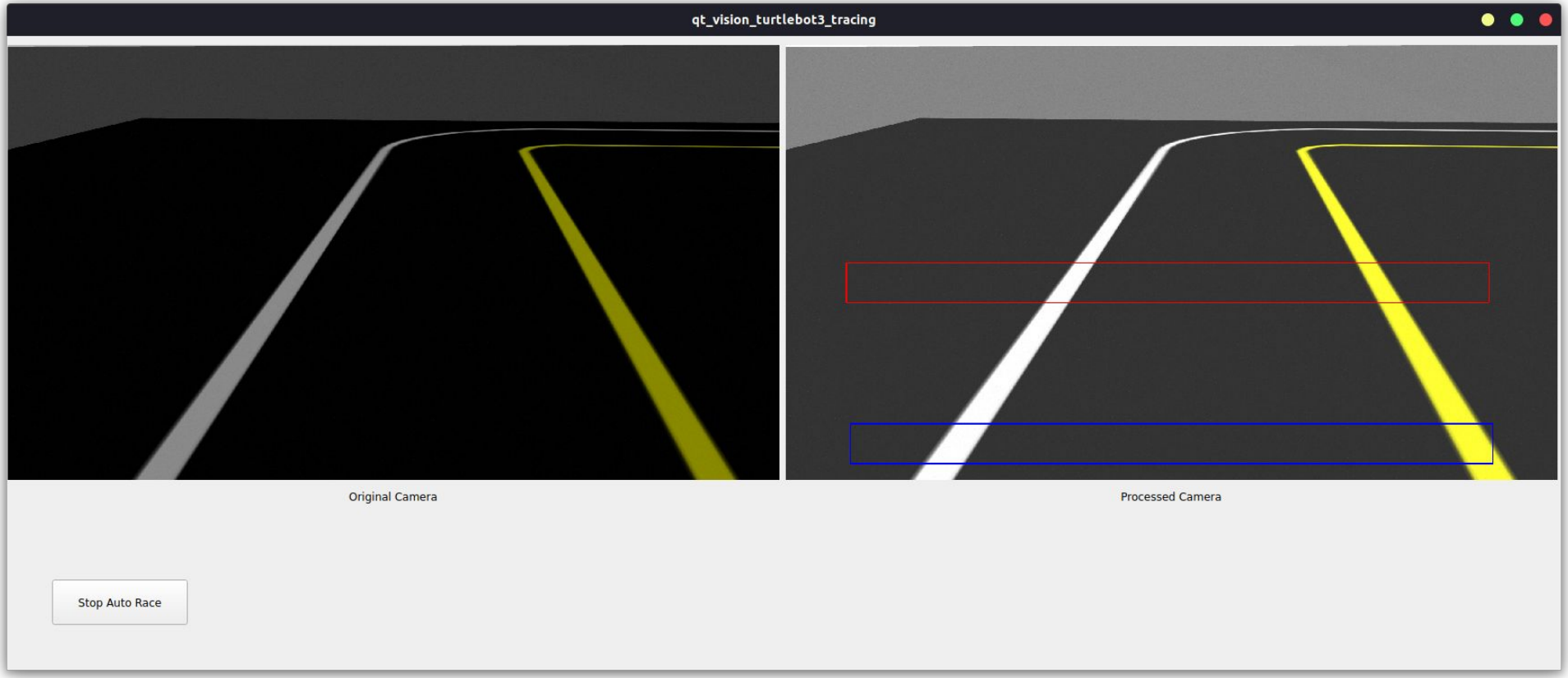
   main_detect: 직진, 좌회전, 우회전 담당

   sub_detect: 코너 담당

4. 바운딩 박스 처리해서 디버깅하기
5. HSV 설정으로 주행할 차선(흰색, 노란색) 및 기믹(빨강, 초록, 파랑) 차선 검출하기
6. 각 detect 내에서 선 감지하기
7. 각 detect 내에 선이 존재하면 true, 선이 존재 하지 않으면 false로 설정하기
8. 7를 통해 얻은 선의 유무를 가지고 주행하기


qt_vision_turtlebot3_tracing
Original Camera
Processed Camera
Stop Auto Race


qt_vision_turtlebot3_tracing
Original Camera
Processed Camera
Stop Auto Race


qt_vision_turtlebot3_tracing
Original Camera
Processed Camera
Stop Auto Race

2. qt_vision_turtlebot3_maze

Main Algorithm

1. /scan, /turtlebot3_status_manager/number, /cmd_vel 토픽으로 LiDAR 센서 받아 오기, 선택된 Maze Exit number에 대한 value 가져오기, turtlebot3 제어하기
2. 시도한 알고리즘
   a. 오른쪽 벽만 감지해서 제어하기(라인트레이서처럼)
   b. LiDAR 센서의 모든 값(0~359)중 방향을 지정하고, 평균을 내서 가장 벽과 멀리 있는 방향으로 이동 제어하기 (전방향(315 ~ 45), 우측(46 ~ 135), 후방(136 ~ 225), 좌측(226 ~ 314)으로 구역 나눠서 평균 내기)
   c. 우선순위를 정해서 제어하기(현재)

```cpp
void QNode::runEscape()
{

  geometry_msgs::msg::Twist cmd_msg;

  float speedX = 0.05;
  // 직진 처리
  if (distance_front_current_ > 0.2) {
    // 직진 도중 오른쪽에 벽이 없을 때
    if (distance_right_current_ > 0.25) {
      // 벽이 없지만, 값이 최댓값(3.5)을 넘어설 때: 첫 번째 코너에 대한 예외처리
      if (distance_right_current_ > 3.5) {
        cmd_msg.linear.x = speedX;
        cmd_msg.angular.z = 0.0;
        RCLCPP_WARN(rclcpp::get_logger("QNode"), "우! 회! 전! 예외처리11111");
      } else {
        // 벽이 없고, 왼쪽 벽과의 거리가 가까울 때: 왼쪽 벽 트래킹
        if (distance_left_current_ < 0.18) {
          cmd_msg.linear.x = speedX;
          cmd_msg.angular.z = 0.0;
          RCLCPP_WARN(rclcpp::get_logger("QNode"), "우! 회! 전! 예외처리22222");
        } else {
          cmd_msg.linear.x = speedX;
          cmd_msg.angular.z = -0.5;
          RCLCPP_WARN(rclcpp::get_logger("QNode"), "우! 회! 전!");
        }
      }
    } else if (distance_right_current_ > 0.25) {
      cmd_msg.linear.x = speedX;
      cmd_msg.angular.z = 0.5;
      RCLCPP_WARN(rclcpp::get_logger("QNode"), "좌! 회! 전!");
    } else {
      cmd_msg.linear.x = 0.2;
      cmd_msg.angular.z = 0.0;
      RCLCPP_WARN(rclcpp::get_logger("QNode"), "직! 진!");
    }
  } else {
    RCLCPP_WARN(rclcpp::get_logger("QNode"), "멈! 춤!");
  }


  publisher_cmd_vel_->publish(cmd_msg);
}
```