Visión Artificial

# PRÁCTICA FINAL

Grado en Ingeniería de Robótica Software

**AUTOR:** Saúl Navajas Quijano

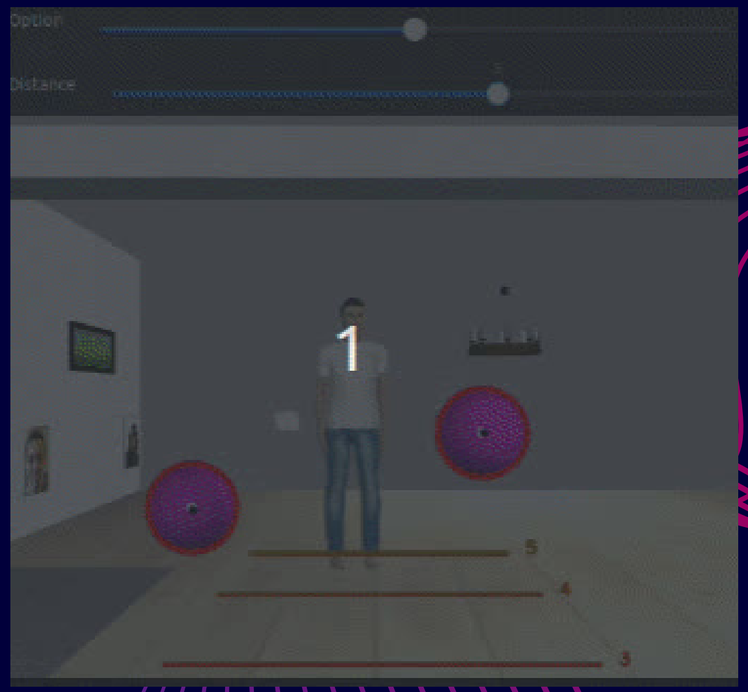
## Partes Implementadas

- Detección pelota 2D y proyección 3D
- Detección pelota 3D y proyección 2D
- Proyección líneas 2D y 3D
- (Extra) Proyección pelota 2D a 3D con el radio calculado en 2D y centro corregido

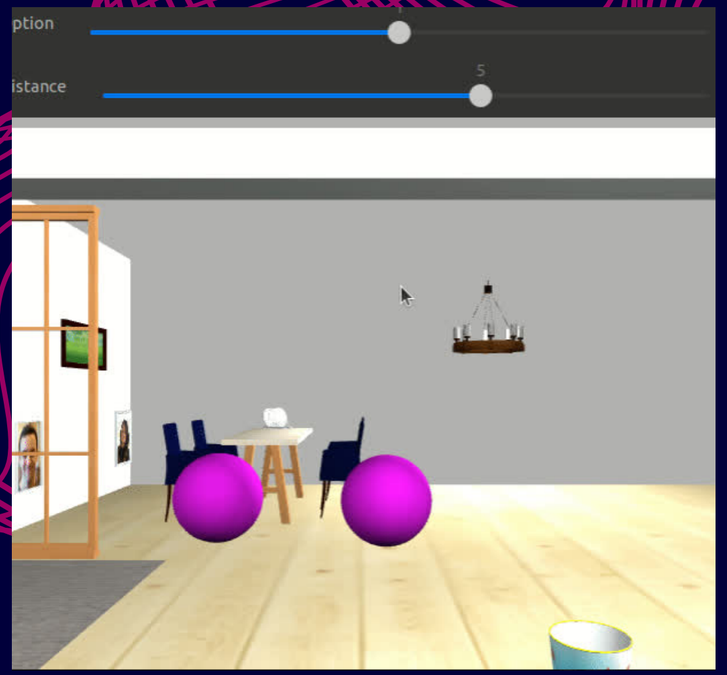
8 de Mayo de 2023

# Detección de una persona en la escena

```
// Load the network
net = cv::dnn::readNetFromDarknet(MODEL_CONFIG_PATH, MODEL_WEIGHTS_PATH
net.setPreferableBackend(cv::dnn::DNN_TARGET_CPU);

// Create a 4D blob from a frame.
cv::dnn::blobFromImage(
    image, blob, 1 / 255.0, cv::Size(INP_WIDTH, INP_HEIGHT), cv::Scalar
      0, 0,
      0), true, false);

//Sets the input to the network
net.setInput(blob);

// Runs the forward pass to get output of the output layers
net.forward(outs, getOutputsNames(net));

// Get the output layers
std::vector<int> outLayers = net.getUnconnectedOutLayers();
std::vector<std::string> layerNames = net.getLayerNames();
std::vector<cv::Rect> detections;

std::vector<int> classIds;
std::vector<float> confidences;
std::vector<cv::Rect> boxes;

/* .... */

// Check if any person is detected
for (int i = 0; i < (int)boxes.size(); ++i)
{
    int classId = classIds[i];
    float confidence = confidences[i];
    if (classId == 0 && confidence > CONF_THREESHOLD)
    {
        detections.push_back(boxes[i]);
        human_detected = true;
        return;
    }
}
human_detected = false;
```




PERSONA DETECTADA



PERSONA NO DETECTADA

# Extra: proyección pelota 2D a 3D teniendo en cuenta el radio en 2D y centro

```cpp
void filter_pink_balls(const cv::Mat image)
{
  cv::Mat pink_mask, gray, output_image = image.clone();
  cv::Vec3i c;
  cv::Point center;
  std::vector<cv::Vec3f> circles;
  size_t i;
  int radius;

  // Apply mask to isolate pink color (hsv image for a better filter)
  pink_mask = filter_pink(image);

  // Blur image for a better circle detection
  cv::medianBlur(pink_mask, gray, BLUR_APERTURE_SIZE);

  // Find circles
  cv::HoughCircles(gray, circles, cv::HOUGH_GRADIENT, INVERSE_RATIO_ACCUM,
                   gray.rows/16, PINK_BALL_LOWER_THEESHOLD, PINK_BALL_UPPER_THREESHOLD,
                   MIN_CIRCLE_RADIUS, MAX_CIRCLE_RADIUS);

  // Store circles info
  detected_balls2d = circles.size();
  for (i = 0; i < circles.size(); i++ ) {
      c = circles[i];

      center = cv::Point(c[0], c[1]);
      radius = c[2];

      balls2D_list[i].x = center.x;
      balls2D_list[i].y = center.y;
      balls2D_list[i].radius = radius;
  }
}
```

CV NODE → PCL NODE

```cpp
int i;
double radius;
cv::Scalar center, p1;
// Draw the 2D detect circles in 3D:
for (i = 0; i < detected_balls2d; i++) {

  // Draw centers
  center = from_2D_to_3D_point(balls2D_list[i].x, balls2D_list[i].y);
  p1 = from_2D_to_3D_point(balls2D_list[i].x+ balls2D_list[i].radius, balls2D_list[i].y);
  radius = std::abs(p1[0] - center[0]);
  out_pointcloud = draw_sphere(out_pointcloud, center[0], center[1], center[2]+radius,
                               COLOR_BLUE, radius);
```

# Demostración del funcionamiento