Deusto
Facultad de Ingeniería
Ingeniaritzako Fakultatea

Proyecto Robótica y automatización inteligente: Hito 1

Gorka Larrea
Estiven Valencia Castrillón
Gorka Celaya
Meryan Sallembien
Unai Arévalo

Introducción

En este proyecto, combinamos los conocimientos adquiridos de visión por computador y robótica para solucionar dos problemas. El primero es detectar por visión una serie de piezas geométricas que están esparcidas por la mesa y llevarlas a su posición del tablero correspondiente.

Por otro lado, el segundo es resolver el problema de las torres de Hanoi utilizando también visión para detectar la posición de las piezas.

Nodos ROS

Los nodos de nuestro proyecto están organizados de la siguiente manera:

Nodos compartidos

- **TakePicture**: Es el nodo inicial del programa. Captura una imagen del entorno utilizando la cámara y determina si debe resolver el puzle de Hanoi o el de Shapes.
- **control_robot:** Este nodo se encarga de recibir datos provenientes de los caminos descritos para cada tipo de puzle y establecer la conexión con el brazo robótico, permitiendo ejecutar los movimientos necesarios.
- **Main:** Actúa como el nodo central del sistema, encargado de supervisar el control de flujo. Gestiona y publica en los distintos topics para determinar si una tarea en un nodo ha sido completada con éxito. En caso afirmativo, envía órdenes sobre los nodos de procesamiento posterior.

Dependiendo de si el puzle es Shapes o Hanoi, se toma alguno de los siguientes caminos:

Shapes

- **DetectPiecesShapes:** Procesa la imagen utilizando OpenCV para detectar las posiciones de las piezas y obtener una serie de coordenadas.
- **AnalyzePiecesShape:** Toma las coordenadas generadas y calcula los movimientos requeridos para resolver el puzle.

Hanoi

- **DetectPiecesHanoi:** Procesa la imagen de manera similar, obteniendo datos clave sobre las piezas del juego.
- **AnalysePiecesHanoi:** Nodo intermedio que recibe los datos del juego, generando una lista de movimientos necesarios para resolver el puzle y preparando información útil para el siguiente nodo.
- **MovementForHano:** Último nodo de este camino. Recibe las instrucciones detalladas y las transforma en cálculos de trayectorias para su ejecución.

Comunicación entre Nodos y Tipos de Datos

Este proyecto implementa topics para la comunicación entre nodos.

- El topic **/TakeImage** es un Booleano que indica la necesidad de sacar una nueva imágen de la cámara. Empieza en True para dar marcha al proceso. La rama de Shapes pide imágenes constantemente, por lo que publica Trues y Falses en este topic. La rama de Hanoi sólo necesita una imágen, así que no publica nada.
- Después de sacar la imagen, el nodo la pone en el topic **/CurrentImage** para que los nodos DetectPiecesShapes/Hanoi la procesen.
- Los topics **/PositionSceneryShape/Hanoi** reciben la información extraída de la imágen y la comunican con AnalysePiecesShape/Hanoi**.** Se puede observar en la siguiente Figura.

- El nodo AnalyzePiecesHanoi publica en el tema **/HanoiMvmnt,** que hace de intermediario para el nodo MovementForHanoi. Este topic recibe una serie de mensajes con la información de los movimientos.

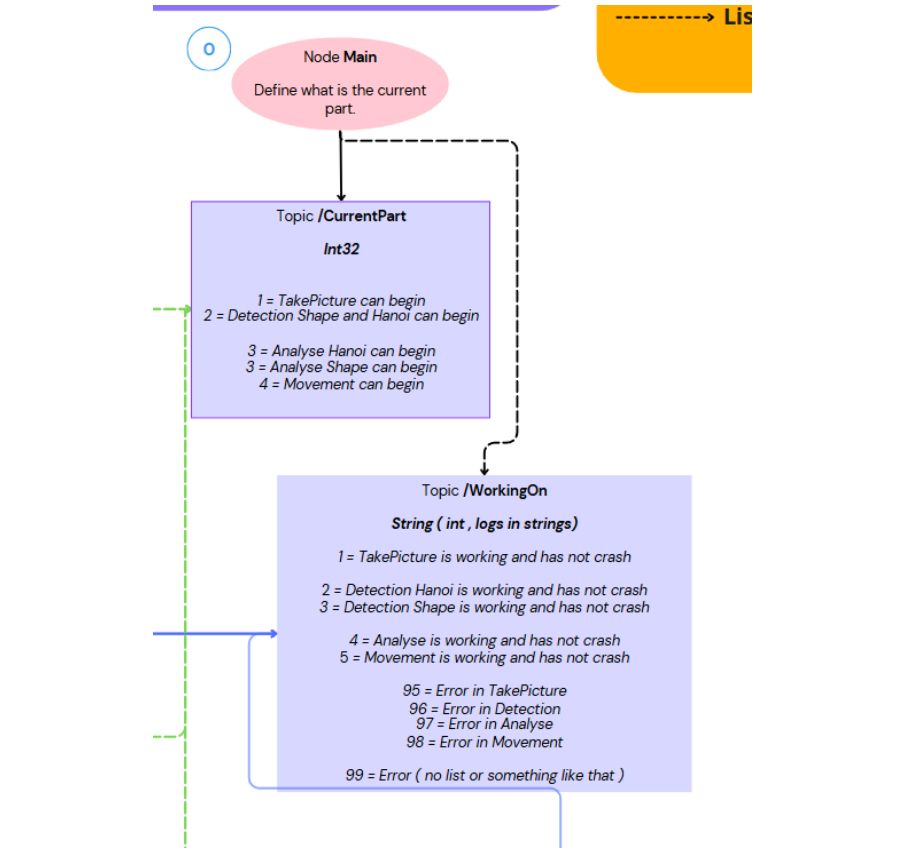
- Estas dos líneas publican a **/MovementToDo** y escuchan a **/MovementDone.** Estos dos topics sirven para manejar el flujo de movimientos que entra al nodo control_robot. ToDo recibe la pose que el robot va a recibir, mientras que Done contiene la información de si está el robot ocupado moviéndose.

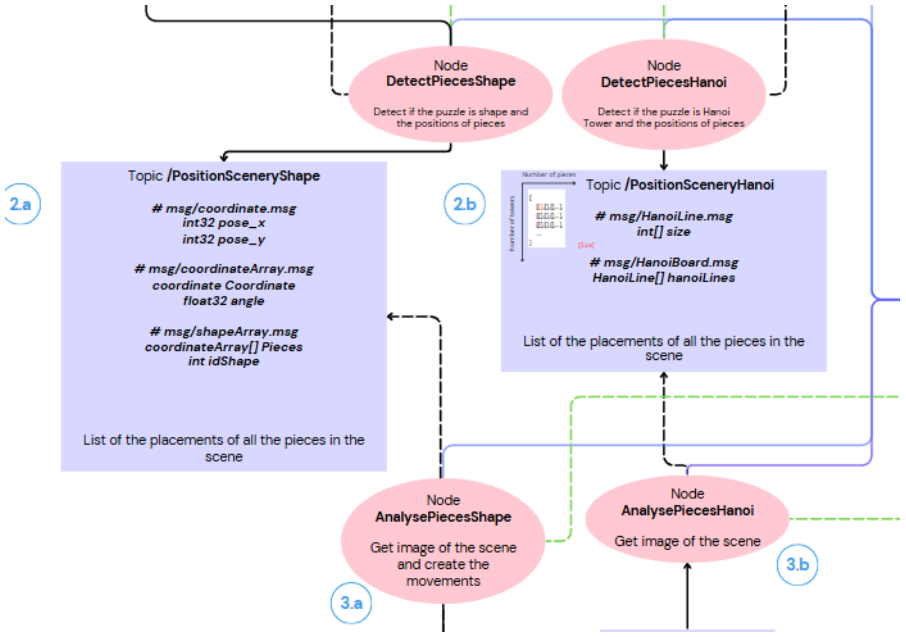
Este proyecto utiliza *topics* para la comunicación entre nodos. A continuación, se describen los *topics* implementados:

1. **/TakeImage**: Es un topic booleano que indica si es necesario capturar una nueva imagen con la cámara. Comienza en True para iniciar el proceso.
    - La rama de Shapes publica constantemente valores True y False en este topic, ya que requiere imágenes de manera continua.
    - Por otro lado, la rama de Hanoi solo necesita una imagen inicial y, por lo tanto, no realiza publicaciones adicionales en este topic.
2. **/CurrentImage**: Una vez que se captura una imagen, esta se publica en este *topic*. Los nodos **DetectPiecesShapes** y **DetectPiecesHanoi** procesan la imagen obtenida desde aquí.
3. **/PositionSceneryShape** y **/PositionSceneryHanoi**: Estos *topics* reciben la información procesada a partir de la imagen y la transmiten a los nodos **AnalyzePiecesShape** y **AnalyzePiecesHanoi**, respectivamente.
4. **/HanoiMvmnt**: El nodo **AnalyzePiecesHanoi** publica los datos generados en este *topic*, que actúa como intermediario hacia el nodo **MovementForHanoi**. Aquí se envía una serie de mensajes con la información detallada de los movimientos necesarios para resolver el puzle.
5. **/MovementToDo** y **/MovementDone**:
    - **/MovementToDo**: Recibe las posiciones o poses que el brazo robótico debe ejecutar.
    - **/MovementDone**: Indica si el brazo robótico está ocupado realizando un movimiento.

    Estos dos *topics* coordinan el flujo de tareas en el nodo **control_robot**, asegurando que los movimientos se gestionen de forma ordenada.

Todos los nodos además escuchan y publican a **/Currentpart** y **/WorkingOn**. Estos dos son topics que manejan la concurrencia en el sistema. Para visualizar la relación entre topics y nodos (diagrama de flujo completo enlace). A continuación se visualizan las relaciones más importantes.


0
Node Main
Define what is the current part.
Topic /CurrentPart
Int32
1 = TakePicture can begin
2 = Detection Shape and Hanoi can begin
3 = Analyse Hanoi can begin
3 = Analyse Shape can begin
4 = Movement can begin
Topic /WorkingOn
String (int , logs in strings)
1 = TakePicture is working and has not crash
2 = Detection Hanoi is working and has not crash
3 = Detection Shape is working and has not crash
4 = Analyse is working and has not crash
5 = Movement is working and has not crash
95 = Error in TakePicture
96 = Error in Detection
97 = Error in Analyse
98 = Error in Movement
99 = Error (no list or something like that)


Node DetectPiecesShape
Detect if the puzzle is shape and the positions of pieces
Node DetectPiecesHanoi
Detect if the puzzle is Hanoi Tower and the positions of pieces
2.a
Topic /PositionSceneryShape
msg/coordinate.msg
int32 pose_x
int32 pose_y
msg/coordinateArray.msg
coordinate Coordinate
float32 angle
msg/shapeArray.msg
coordinateArray[] Pieces
int idShape
List of the placements of all the pieces in the scene
2.b
Topic /PositionSceneryHanoi
msg/HanoiLine.msg
int[] size
msg/HanoiBoard.msg
HanoiLine[] hanoiLines
List of the placements of all the pieces in the scene
Node AnalysePiecesShape
Get image of the scene and create the movements
3.a
Node AnalysePiecesHanoi
Get image of the scene
3.b