

Practica 2: Mapas Topológicos visuales.

Robots Autónomos

Alejandro Muñoz Navarro
Fernando Planes Ruiz

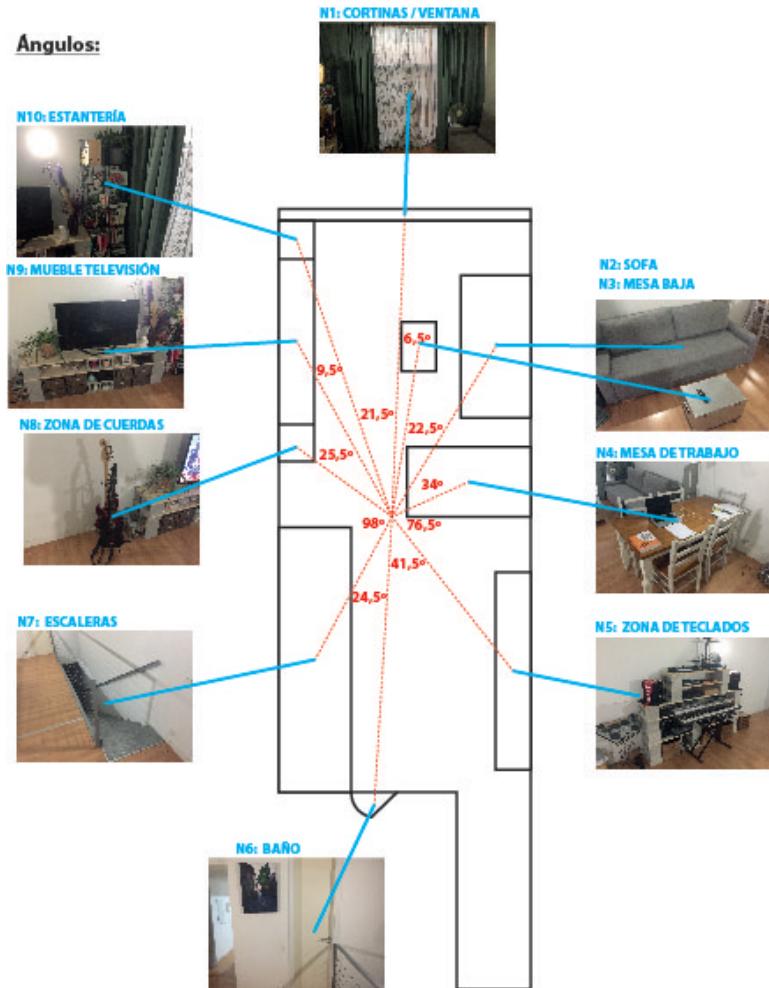
1 Introducción

Esta práctica consiste en el desarrollo y pruebas(estáticas y dinámicas) de un navegador mediante mapas topológicos visuales. Aunque inicialmente se planteaba construir el mapa a partir de múltiples espacios por los que se desplazaría un robot, en nuestro caso lo hicimos en una única estancia y con un robot con capacidad de giro panorámica.

Para ello en primer lugar definimos el mapa topológico del espacio en que estaría el robot. Pensamos que simular el robot usando únicamente una cámara nos daría muchos problemas al ser poco preciso, además de que sería muy difícil repetir el experimento, por lo que lo siguiente que hicimos fue construir el robot. Con el construimos un dataset de 60 imágenes por nodo con las que entrenamos a nuestros clasificadores. Por último usamos el dataset para las pruebas estáticas y el robot para las pruebas dinámicas de los clasificadores. Desarrollamos la práctica con 2 tipos de cámaras, una normal y otra de visión nocturna.

2 Generación del mapa topológico del entorno

Ya que inicialmente se planteo aprovechar no solo el dataset, sino también los ángulos de los nodos con respecto al robot, construimos un primer mapa aproximado del espacio en el que definimos los nodos a reconocer:



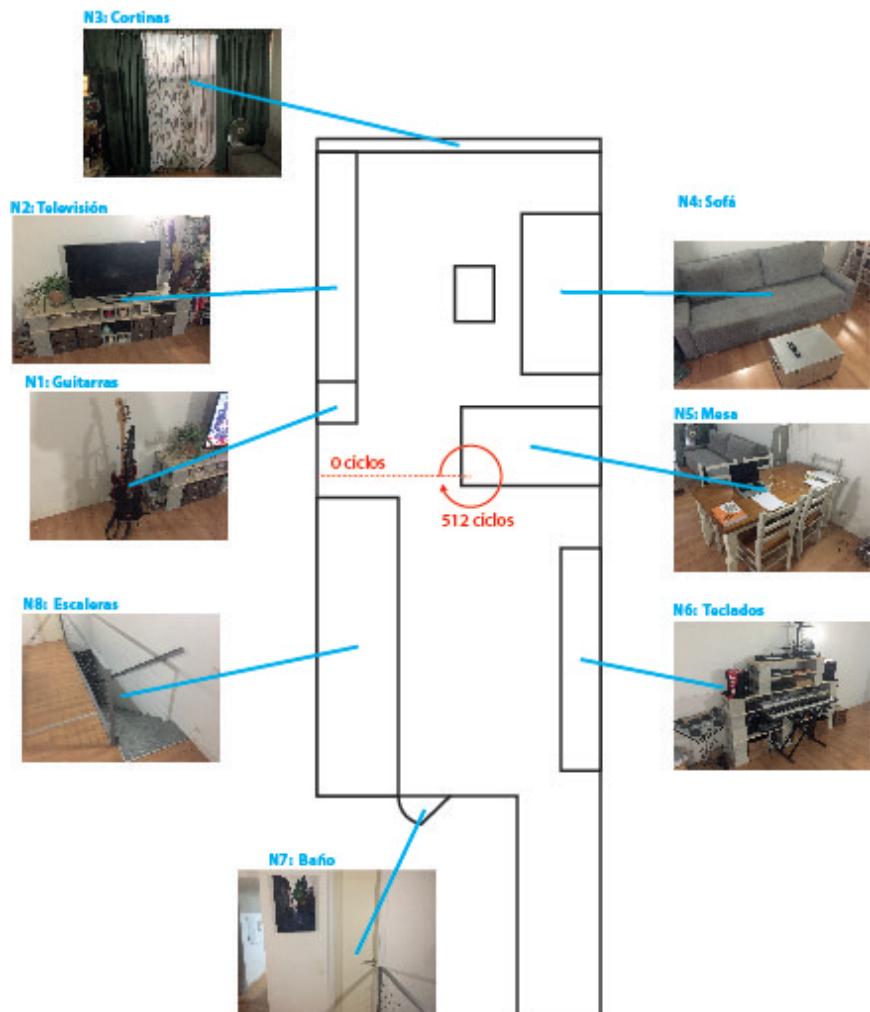
Primer mapa

Este planteamiento inicial tenía varios problemas. Por ejemplo, algunos de los nodos definidos serían barridos en un ángulo de giro muy pequeño y además muy pegados a los nodos vecinos por lo que podríamos tener algunos datasets con imágenes muy similares por lo que optamos por reducir el número de nodos.

Además el robot utiliza motores paso a paso, dando una vuelta en 512 ciclos de 8 medios pasos($1^\circ = \frac{512}{360} ciclos$) por lo que para facilitar el trabajo con buena precisión, en lugar de usar grados nuestra unidad de base será el

ciclo de 8 medios pasos. De esta manera evitamos la aparición de decimales. El robot está programado para que al acabar la ejecución de un programa regrese a la posición inicial (ciclo 0) tal que en cada ejecución del programa el sistema de referencia se mantiene constante.

El mapa final, ya sin nodos problemáticos quedaría de la siguiente forma:



Segundo mapa

Incluimos también unas imágenes que muestran los puntos que definimos como inicio y fin de cada nodo (las 60 fotos de cada nodo estarán entre esos extremos):



Inicio



Fin

Nodo 1: Guitarras [15, 48]



Inicio



Fin

Nodo 2: Televisión [51, 93]



Inicio

Fin

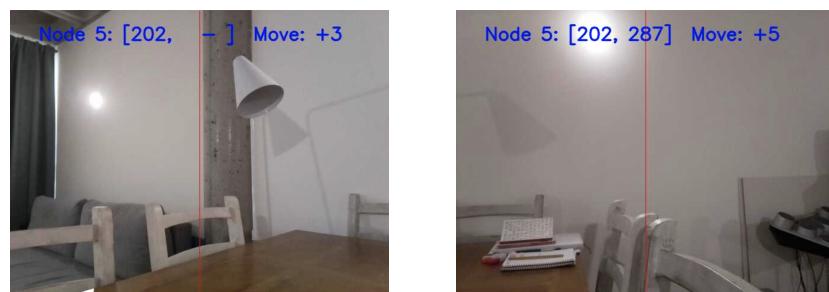
Nodo 3: Cortinas [96, 156]



Inicio

Fin

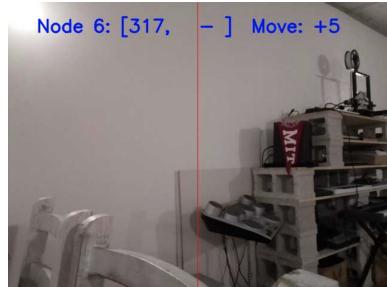
Nodo 4: Sofá [159, 199]



Inicio

Fin

Nodo 5: Mesa [202, 287]



Inicio



Fin

Nodo 6: Teclados [317, 367]



Inicio



Fin

Nodo 7: Baño [382, 407]



Inicio



Fin

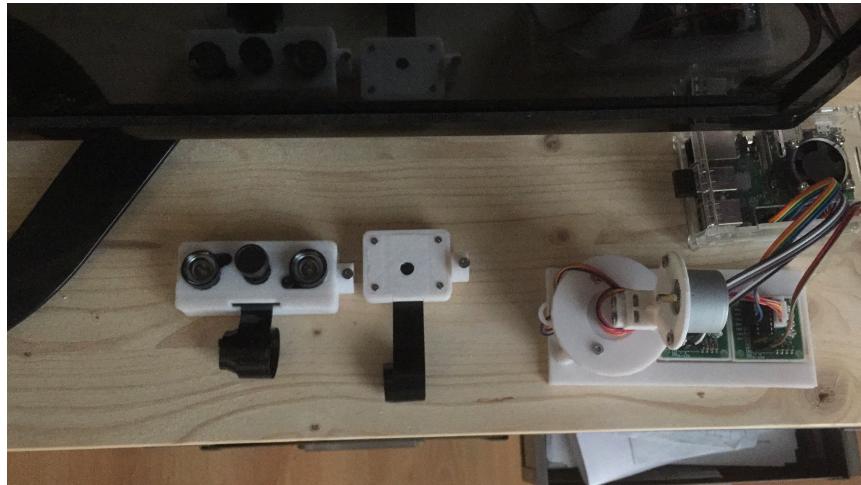
Nodo 8: Escaleras [410, 490]

3 Construcción del robot

Para construir el robot nos apoyamos en el diseño [aquí enlazado](#). Este robot permite giros de tipo pan y tilt, aunque para la práctica nos limitaremos a los giros panorámicos.

Hemos utilizado una Raspberry Pi 3B+ con sistema operativo Raspbian a la que podremos conectarnos por escritorio remoto. Además podemos conectar el dispositivo a una batería portátil, lo que facilita colocar el robot a funcionar en cualquier sitio y controlarlo desde otro ordenador o incluso un teléfono móvil.

Realizamos la práctica con 2 cámaras distintas. Una normal y otra de visión nocturna (cámara de infrarrojos). No solo cambia entre una y otra el espectro de luz, sino que además el ángulo de visión es distinto, teniendo la cámara de visión nocturna mayor distorsión hacia los bordes de la imagen.



Robot con las 2 cámaras utilizadas.

Como ya hemos comentado, los motores que lleva el robot son motores paso a paso y que dan una vuelta entera tras 512 ciclos de 8 medios pasos. En la programación del robot para el movimiento nos hemos restringido a los ciclos, lo que nos permite giros por debajo de 1° , pero cuando pedimos al robot que saque 60 imágenes dentro de un rango de ciclos, trabajaremos con una resolución de medios pasos, por lo que en 1 ciclo podríamos sacar hasta 8 fotos. Así mientras el rango sea como mínimo de 8 ciclos (64 medios pasos = $5,625^\circ$) el robot podrá captar las 60 imágenes distintas para el dataset.



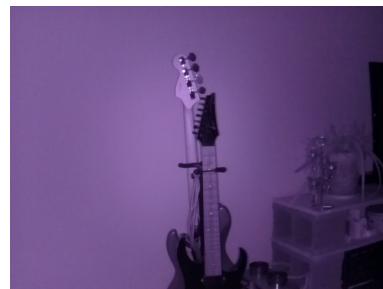
Robot en la posición final del experimento.

4 Obtención del dataset

Nuestro código permite definir de manera interactiva el inicio y final de cada nodo y tomar 60 imágenes dentro de ese rango de manera automática. Con ello obtuvimos 2 datasets (cámara normal y de visión nocturna) de 60 imágenes para cada uno de los 8 nodos.



Cámara normal



C. de visión nocturna

Ejemplo de imágenes de los datasets

Se puede acceder a los 2 datasets y el código utilizados en el siguiente [enlace](#).

5 Reconocedores

Construimos 3 reconocedores (el código puede encontrarse al final de la memoria). En concreto todos son de tipo 1-NN- δ , utilizando el primero la distancia entre histogramas de la imagen en blanco y negro, el segundo la distancia entre supervectores obtenidos de la combinación de los 3 histogramas de cada canal de color y por último calculando la distancia entre las imágenes guardadas como matrices.

Dado que para nuestro problema es muy frecuente que se vean varios nodos a la vez, veremos como los que trabajan con histogramas tenderán a etiquetar los huecos que no hemos definido como parte de los nodos como si fuesen el nodo más cercano al centro de la imagen, mientras que el que emplea matrices generará más "unknowns" en estos huecos.

Ambos resultados podrían ser deseables según el tipo de aplicación que se tenga en mente. No obstante tendremos que recordar esto a la hora de interpretar resultados, pues si únicamente miramos los porcentajes de error, veremos como el de matrices funciona aparentemente mucho mejor y sin embargo al mirar los videos veremos como los de histogramas son capaces de llenar los huecos que nosotros no hemos definido con resultados que también tienen sentido.

6 Pruebas estáticas

Al haber construido el dataset con imágenes tomadas de manera secuencial para cada nodo y de manera equidistante, los resultados que obtenemos en las pruebas estáticas son bastante buenos ya que las imágenes de cada nodo son muy similares entre sí.

En concreto obtuvimos los siguientes resultados:

- Cámara normal:
 - 1-NN- δ Hist. Grises: True error rate % = 0.208334
 - 1-NN- δ Hist. RGB: True error rate % = 0.208334
 - 1-NN- δ Matrices: True error rate % = 2.5
- Cámara de visión nocturna:
 - 1-NN- δ Hist. Grises: True error rate % = 0.208334

- 1-NN- δ Hist. RGB: True error rate % = 0.208334
- 1-NN- δ Matrices: True error rate % = 6.041667

7 Pruebas dinámicas

Como ya mencionamos anteriormente, los errores los estamos midiendo tomando como referencia los barridos que definimos para cada nodo, pero que no se corresponden con si los nodos aparecen o no en la imagen. Por tanto, aunque los errores porcentuales que obtenemos indican unos malos reconocedores, conviene mirar los videos para interpretar adecuadamente los resultados.

Los reconocedores de matrices dan buenos resultados porcentuales ya que se ajustan mucho mejor a como hemos definido el problema (coinciden en mas "unknowns"/no-definido), sin embargo, si preguntamos a una persona mirando los videos cual funciona mejor, probablemente se decantará por los que usan histogramas, ya que llenan los huecos donde aparecen nodos pero que no se incluyeron en el dataset pues entraban en un terreno cercano al de la lógica borrosa.

Los resultados y videos obtenidos son los siguientes:

- Cámara normal:
 - 1-NN- δ Hist. Grises: True error rate % = 31.3439 [video](#).
 - 1-NN- δ Hist. RGB: True error rate % = 30.5509 [video](#).
 - 1-NN- δ Matrices: True error rate % = 12.1452 [video](#).
- Cámara de visión nocturna:
 - 1-NN- δ Hist. Grises: True error rate % = 19.7470 [video](#).
 - 1-NN- δ Hist. RGB: True error rate % = 28.7421 [video](#).
 - 1-NN- δ Matrices: True error rate % = 22.4174 [video](#).