PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
IRB2001- FUNDAMENTOS DE ROBÓTICA

# TAREA ROBÓTICA COGNITIVA

**Fecha Máxima de Entrega: Viernes 30/11**

Muchach@s, en esta tarea tendrán la oportunidad de dotar a un robot con capacidades para encontrar su posición utilizando un mapa de celdas o un sistema de reconocimiento visual de ambientes.

## 1 Localización usando mapa de grid de evidencia

Para las siguientes actividades asuma que conoce el mapa del mundo, el cual consiste de un laberinto formado por celdas de 10x10 cm. El robot tiene forma circular con un diámetro de *5* cm. Los mapas son cargados automáticamente por la librería entregada y están almacenados en archivos .csv que contienen los cuatro posibles valores que puede tomar cada celda: tilewall, tileground, tilestart y tilefinish. Estos representan las murallas, espacios libres, posición inicial y posición final respectivamente. Para esto utilizarán una librería para *Python*, la cual se encuentra en la web del curso bajo el nombre de *LibreriaMapas.zip*.

Programe una función que permita al robot encontrar su posición en un mapa conocido utilizando la técnica State-Set Tracking-Automata. La función recibirá como entrada un mapa del ambiente en el formato descrito anteriormente. Luego, siga los siguientes pasos:

- Programe la función WhatDoISee(x,y,dir), que recibe como entrada las coordenadas (x,y) de una potencial casilla en la que se encuentra el robot y la dirección que se está observando, y retorna un valor asociado a la distancia a la que encuentra el primer muro que ve (puede asumir que el mapa está rodeado externamente por murallas, por lo que nunca debe entregar un valor infinito). Si así lo desea, puede implementar una función que realice las mediciones de las 4 orientaciones posibles (N, S, E, W) y almacenarlas en una lista.

- Usando la función WhatDoISee(x,y,dir) implemente la técnica State-Set Tracking Automata para localizar el robot. Para esto, inicialice el conjunto de posibles estados (posiciones y direcciones) del robot utilizando una lista que incluya todas las posibles celdas vacias con sus respectivas cuatro orientaciones del mapa de entrada. Luego, diseñe una heurística para el movimiento del robot, y según ésta actualice en cada movimiento el conjunto de estados (state-set) con las posibles posiciones del robot en el mapa. Continúe iterando hasta que no sea posible seguir reduciendo el conjunto de posibles posiciones del robot.

- El mapa de entrada contiene la información de la posición inicial del robot, use esta posición para inicializar el robot, sin embargo, tenga presente que el robot no conoce esta posición y que mediante el uso de la técnica de State-Set Tracking Automata debe localizarse. Adicionalmente, dado que en esta tarea el objetivo es sólo localizar el robot y no alcanzar una meta, la información del nodo final no es relevante.

- Reporte el código resultante y una explicación de la heurística utilizada para elegir la siguiente acción del robot. Asuma que la ejecución de cada acción seleccionada es ejecutada en forma correcta, es decir, no hay incerteza en el modelo de movimiento del robot. Eso sí, dado que el robot no conoce su dirección, no sabrá en qué dirección (N,S,W,E) se está moviendo, pero sí sabrá que es relativo hacia donde está mirando (el robot recibe información de sus sensores indicando lo que ve hacia el frente, izquierda, detrás y derecha).

# 2 Localización usando percepción visual

## 2.1 Espacio de Características (Feature space)

Como comentamos en clase, redes neuronales de aprendizaje profundo (deep learning) han surgido recientemente como una poderosa herramienta que permite aprender características (features) que permiten aumentar significativamente el rendimiento de diversas tareas de clasificación, tales como reconocimiento de voz, texto, o imágenes. Como también discutimos en clases, en general, el entrenamiento de estas redes requiere ajustar millones de parámetros, lo cual es un proceso lento y altamente demandante de recursos computacionales. Afortunadamente, existen redes pre-entrenadas en base a grandes volúmenes de datos, las cuales pueden ser usadas para obtener un vector de caraterísticas de alta discriminatividad. En el caso de imágenes, un modelo muy conocido es la red neuronal profunda Alex's Net, que en esta tarea usaremos para llevar a espacio de características la información visual de áreas locales de cada imagen, según se detalla a continuación.

## 2.2 Preprocesamiento: Ventana deslizante y max-pooling

Una de las características relevantes del mundo visual es la coherencia local de diversos patrones visuales. Por ejemplo, al considerar imágenes de bares, existen estructuras espacialmente localizadas que se repiten, como pisos, mesones, áreas con botellas, etc. Para capturar esta información local se utiliza un estrategia denominada ventana deslizante (sliding window), que consiste en deslizar sobre la imagen una ventana espacial según un paso regular en la dirección horizontal y vertical. Así es posible obtener vectores de atributos de distintas áreas de cada imagen que denominaremos patches, tal como muestra la Figura 1.

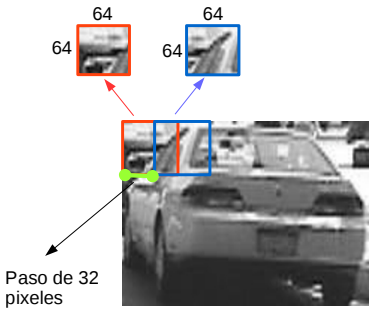


Figure 1: Ventana deslizante usando patches de 64x64 pixeles y un paso de 32 pixeles.

Las imágenes usadas en esta actividad tienen una resolución de 256x320 pixeles, por tanto, al utilizar una ventana deslizante de 64x64 pixeles y un paso de 32 pixeles, se obtiene en cada imagen un total de 7x9=63 patches. Cada uno de estos patches se ingresa a Alex's net para obtener un vector (feature vector) de 4096 dimensiones para cada patch. Luego, para obtener el vector de atributos de la imagen completa, se integran los vectores de cada patch usando una estrategia denominada max-pooling. Ésta consiste en tomar la máxima respuesta para cada dimensión del vector de atributos. Por ejemplo, si en lugar de 4096 cada vector fuera de 5 dimensiones y tuviéramos 3 patches, el resultado de max-pooling para los siguientes 3 vectores sería: maxpool {(**10**, 3, 4, 5, 6); (3, **21**, 4, 5, 6); (1, 3, **80**, **100**, **50**)}=(10, 21, 80, 100, 50). En este laboratorio, para cada imagen el proceso de max-pooling entrega un vector de 4096 dimensiones.

Para facilitar la implementación de esta tarea, ya hemos aplicado el procedimiento anterior a cada una de las imágenes que utilizarán, por ende, el set de datos proporcionado consistirá de la imagen original usada como referencia y también un vector de 4096 dimensiones (feature space) que será utilizado para implementar clasificadores del tipo SVM.

## 2.3 Actividades

La actividad la realizaremos utilizando códigos en lenguage de programación Python. Adicionalmente, usaremos la librería scikit-learn `http:\\www.scikit-learn.org`, la cual provee una gran gama de técnicas

de aprendizaje de máquina, tal como clasificadores del tipo SVM.

Para esta parte de la tarea descargue el siguiente material:

- Dataset Scenes-15-Classes.zip disponible en : `https://www.dropbox.com/s/fowbajpsf0w05hi/Scenes-15-Classes.zip?dl=0`.
- Archivos con código Python para ejecutar reconocimiento visual, disponibles en sitio web del curso en Siding.

Las siguientes actividades consistirán en ejecutar los códigos de Python provistos en el sitio web de la tarea. Específicamente, usaremos el código en el archivo classifier.py. Antes de ejecutar este código verifique que las rutas al set de datos de entrenamiento sean correctas.

Utilizando el set Scenes-15-Classes pruebe el rendimiento de un clasificador SVM. Para el clasificador utilice los valores por defecto.

- Documente sus resultados utilizando el nivel de exactitud reportado para el set de entrenamiento y test. ¿Nota diferencias entre los rendimientos obtenidos para estos sets?, comente fundamentando sus observaciones.
- Analice la matriz de confusión reportada, ¿Cuál es el error más común?, ¿Qué categoría obtiene el mejor rendimiento?, ¿Cuál es el más bajo?. Indique razones que puedan justificar estos resultados. Para esto mire las imágenes reales incluidas en el set de datos.
- Vuelva a ejecutar el clasificar utilizando distintos valores para el coeficiente de penalización de las variables slack. Para ello, en la definición del clasificador modifique el valor de C (por defecto *C=1.0*). Reporte y comente sus resultados.