Reconocimiento de patrones

De Rito Micaela, Foglia Julieta, Monardo Javier, Perez Lautaro, Reynoso Thomas

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
micaeladerito@gmail.com, juli.foglia@gmail.com, javorengero2000@gmail.com,
laautaperez@gmail.com, thomi.reynoso@gmail.com

Resumen. Este trabajo de investigación intenta comprobar si es posible utilizar aprendizaje automático para que TransporMate reconozca un patrón de movimientos a través de imágenes captadas por su cámara integrada, y que lo pueda reproducir por sí solo, utilizando procesamiento en paralelo mediante GPU.

Palabras claves: TransporMate, HPC, GPU, Machine Learning, Reconocimiento de Patrones.

1 Introducción

Esta investigación se basa en añadir a nuestro proyecto TransporMate[1], que consiste en movilizar el Mate sobre una plataforma con ruedas controlada por una aplicación Android que se comunica vía Bluetooth con el dispositivo que lo transportará al destino, una funcionalidad extra que permitirá activar un modo “Automático”, en el cual el carro realizará el camino que recorrió en rondas pasadas, de forma sistematizada.

Para esto, haremos uso del módulo GPU integrado en nuestro dispositivo Android, que a través del procesamiento de imágenes e información de forma paralela nos permitirá recolectar los datos necesarios para memorizar el camino recorrido, y a su vez, brinde al usuario la posibilidad de seguir controlando el funcionamiento convencional del carro.

GPU, a diferencia de la secuencialidad que manejan las CPUs, es considerablemente más poderoso, sobre todo cuando se trata de operaciones que se pueden realizar en paralelo. Para sacar provecho de esto, los algoritmos deben presentar alto grado de paralelismo o grandes requerimientos computacionales, como por ejemplo algoritmos de aprendizaje automático.

Actualmente y en mayor escala a lo que es nuestro proyecto, existen vehículos de conducción autónoma en diferentes niveles: sin automatización en la conducción, con asistencia en la conducción, automatización parcial (sin detección ni respuesta ante objetos, se precisa conductor), automatización condicionada (autonomía más elevada pero aun así se

1 https://github.com/julieta-foglia/TransporMate

necesita conductor), automatización elevada (no se precisa la intervención humana) y automatización completa. Ejemplos de esto son el vehículo Waymo[2] de Google o el Modelo X [3]de Tesla.

2 Desarrollo

Para aplicar la funcionalidad propuesta en esta investigación, como primera medida, necesitaremos integrar al auto una cámara como medio de obtención de información, para que, de esta manera, paralelamente se pueda procesar la misma, permitiendo almacenar el camino realizado. La cámara enviará las imágenes captadas a la placa Arduino, y ésta, a su vez, al dispositivo Android para que sean procesadas.

Una vez recolectada cierta cantidad de información, el dispositivo mostrará la opción de manejo “Automático”. La misma consistirá en avanzar hacia el destino correspondiente una vez que el mate sea depositado en el carro. Esta detección se realizará mediante los sensores de detección de infrarrojo (PIR) y el pulsador Switch Reed, ya presentes en el dispositivo, que nos indicarán cuando el mate se encuentre apoyado en la plataforma y si ha sido tomado o no.

Para lograr la implementación utilizaremos la librería OpenCV[4], la cual contiene algoritmos de *Computer Vision* (nos da la posibilidad de enseñarle a la computadora a que interprete el significado del mundo físico a través de la visión) y de aprendizaje automático. Estos algoritmos pueden usarse para reconocimiento facial, identificación de objetos, seguir los movimientos de una cámara, seguir objetos en movimiento, seguir movimiento ocular entre muchas otras cosas.

La librería se basa en la API OpenCV 2.x, la cual es esencialmente una API C++. Tiene una estructura modular, de la cual utilizaremos varios de ellos: la funcionalidad núcleo (que define estructuras de datos básicas y funciones utilizadas por otros módulos), el módulo de procesamiento de imágenes, el módulo de video (incluye algoritmos de seguimiento de objetos entre otras cosas), el módulo GPU (algoritmos acelerados mediante GPU de distintos módulos).

3 Explicación del algoritmo.

El input del sistema de navegación es una imagen en escala de grises de 64x64, obtenida a través de la cámara a incluirse en el carro. Luego la imagen es analizada por dos procesos: el primero define la **extensión espacial de regiones** "diferentes" o no esperadas de la imagen, utilizando un algoritmo de detección de similitudes, que calcula el valor de cada pixel

2 https://waymo.com
3 https://www.tesla.com/modelx
4 https://opencv.org

comparando con los pixeles que lo rodean; el segundo es un **proceso de reducción de resolución** a un mapa de representación de 10x10.
Luego, esta imagen representada se pasa a una **red neuronal**, acompañado de la orientación/acción que pretende realizar el carro. El objetivo de la red es formar una representación espacial de la relación entre cada imagen capturada por el carro y la orientación o acción tomada en el momento en que se captura esa imagen.

La estructura de la red consiste en dos capas. La primera es una capa de representación (confeccionada a partir de las unidades de 10x10 tomadas por la cámara y luego procesadas). Cada una de esas unidades es mapeada a cualquiera de las 12 unidades de la segunda capa, que es la capa de orientación (representa la orientación del robot).

Durante el periodo de aprendizaje, se captura una secuencia de impresiones visuales del entorno del robot, lo que se usa como entrada de la red. Se identifican y procesan las imágenes, que se almacenan en la capa de representación al igual que la orientación (que puede ser obtenida por medio de algún sensor), que se almacena en la capa de orientación.

Si luego de realizar otra captura la orientación del robot sigue siendo la misma, pero no así la imagen capturada, ésta última se eligen representación de la posición actual. El robot continuará moviéndose en su dirección actual hasta la adquisición de una nueva captura.

En caso de que luego de una captura la orientación del robot no sea la misma que la actual, entonces la imagen actual en la capa de representación es eliminada y se procede a realizar una nueva captura.
Utilizaremos la librería OpenCV que nos permitirá correr en segundo plano el procesamiento de la información. A continuación pasaremos a explicar el funcionamiento interno de la misma:

OpenCv consiste en almacenar cada imagen en una una matriz que contiene todos los valores de intensidad de los puntos de píxeles. Esos píxeles pueden ser almacenados de distintas formas seleccionando el espacio de color (cómo combinamos los componentes de color para codificar un color determinado) y el tipo de datos utilizado. En nuestro caso, utilizaremos el sistema de color RGB, que se basa en los colores Rojo, Verde y Azul, y nos permitirá tener una mayor precisión a la hora de comparar imágenes.

Para crear nuestra matriz, debemos indicarle el tamaño en filas y columnas, y el tipo de imagen a guardar. Con la función “imread” guardaremos la imagen capturada por la cámara en nuestra matriz, y de esta forma podremos almacenar todo el camino recorrido por nuestro TransporMate, e ir comparándolo con las capturas en tiempo real.

```
Mat imagen = (20,20,CV_LOAD_IMAGE_COLOR);
//Creo la matriz de 20 x 20, para almacenar imágenes a color.

imagen = imread("//rutaImagenCapturada", CV_LOAD_IMAGE_COLOR);
//Le asigno a la matriz la imagen capturada por la cámara.
```

Pseudocódigo

A continuación, mostraremos una parte del pseudocódigo a implementar, con el cual obtendremos y guardaremos las imágenes, y por consiguiente, obtendremos los caminos realizados, y decidiremos si es posible activar la modalidad Automática.

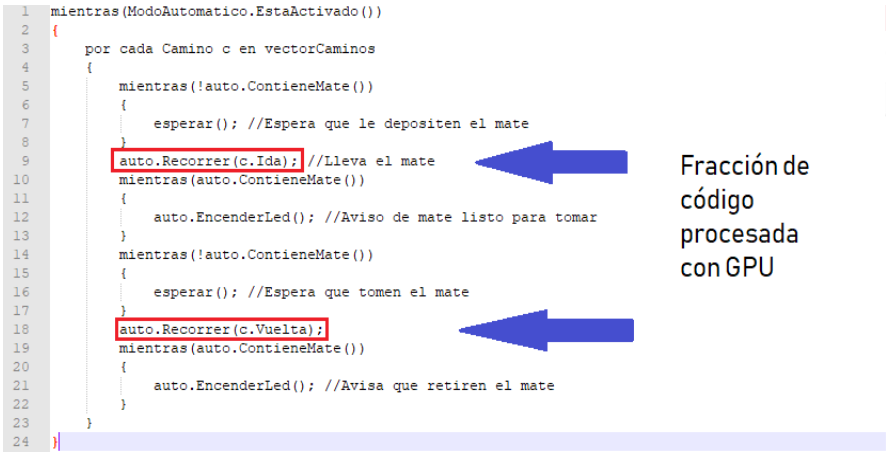
```
mientras(verdadero)
{
    List<Camino> vectorCaminos;
    Auto auto;
    Camara camara;
    mientras(auto.EnMovimiento() && auto.ContieneMate())
    {
        camara.CapturarImagen();
        nuevo.GuardarCamino(camara.Imagen);
    }
    si(vectorCaminos.Contiene(nuevo)
        vectorCaminos[nuevo].repeticiones ++;
    sino
        vectorCaminos.Agregar(nuevo);
    si(vectorCaminos.RepiteTodosLosCaminos())
        HabilitarModoAutomatico();

}
```

Fracción de código procesada con GPU

El GPU sería necesario para el procesamiento de la información y el manejo del transportador de forma paralela, pero principalmente será útil en el procesamiento de imágenes, materia en la cual es especialista.

Al habilitar el ModoAutomático, se implementará el siguiente pseudocódigo, que nos permitirá realizar el recorrido guardado anteriormente con ayuda del procesamiento de GPU.



La función “auto.Recorrer()” será procesada con GPU. La misma consistirá en la captura de imágenes en tiempo real de la posición actual del auto, y la comparación con las ya guardadas. Al encontrar coincidencia, irá avanzando y tomando nuevas imágenes para, de esta forma, poder completar tanto el recorrido de ida, como de vuelta.

4 Pruebas que pueden realizarse

Podemos realizar la prueba del algoritmo de varias formas, luego de hacer las respectivas implementaciones. Como primera prueba, podemos realizar un camino muy simple, por ejemplo, todo recto, de manera repetida, y comprobar luego de que la opción de manejo automático este activada, que el auto realiza el camino de la manera correcta y esperada.

Para la segunda prueba, podemos ir aumentando la dificultad, realizando un camino más complejo (pero que sea corto), incluyendo movimientos diagonales, y comprobar el correcto funcionamiento del modo automático.

Luego podemos realizar una prueba mucho más compleja: Primero, podemos mover el robot de forma manual a través de un camino preestablecido (más largo que la anterior prueba), ubicando imágenes bien definidas (por ejemplo, figuras geométricas) para comprobar que el procesamiento y reconocimiento de imágenes se realiza de manera correcta y se transmite a las respectivas capas de la red neuronal.

Luego, podemos realizar el mismo proceso de manera automática, para comprobar si el carro puede moverse de forma autónoma a través del camino aprendido.

Como prueba final, podemos realizar varios caminos complejos, objetivo final que tiene nuestra investigación, que generen un trayecto de “ronda” para repartir el mate, y que luego de

varias repeticiones de la misma, el modo automático funcione de la manera correcta, realizando los caminos preestablecidos, y transportando el mate sin ningún inconveniente.

5 Conclusiones

A través de esta investigación intentamos demostrar que se puede implementar el aprendizaje automático a TransporMate utilizando una cámara integrada y una red neuronal, de esta forma puede realizar recorridos de manera autónoma e incluso aprenderlos. Dicha implementación se puede considerar viable en cuanto a lo económico y tecnológico, ya que existen distintas herramientas que facilitan el desarrollo e integración de los componentes explicados previamente. Asimismo, muchas de las grandes empresas tecnológicas del mundo han profundizado y realizado prototipos que incluyen dichas funcionalidades, por lo que tendríamos una base firme de la cual apoyarnos a la hora de integrar algoritmos *Machine Learning* y *Pattern Recognition.*

Durante la investigación encontramos una gran diversidad de algoritmos para procesamiento de imágenes y *Machine Learning*, separados en grandes grupos: Aprendizaje Supervisado, Aprendizaje por Refuerzo, y Aprendizaje No Supervisado. Dentro de éste último grupo se puede ubicar el de Redes Neuronales, el cual elegimos para nuestra investigación.

6 Referencias

1. Jeong-Min Choi , Sang-Jin Lee, Mooncheol Won. “Self-learning navigation algorithm for vision-based mobile robots using machine learning algorithms” (2010). http://pdfs.semanticscholar.org/3aad/005a9f55dac9a755b00b6f006c8fac8a375d.pdf

2. Sirma Yavuz. “Determining shortest path and obstacle recognition of a map building robot” (2007). http://emo.org.tr/ekler/7a20639726a041d_ek.pdf

3. Jesús Pardal Garcés. “Aceleración de algoritmos de Visión hiperespectral mediante GPU” (2014) https://riunet.upv.es/bitstream/handle/10251/37033/PardalGarces_Jes%C3%BAs.pdf?sequence=1

4. OpenCv Library. https://docs.opencv.org/3.1.0/d5/df8/tutorial_dev_with_OCV_on_Android.html , https://docs.opencv.org/2.4/doc/tutorials/core/mat_the_basic_image_container/mat_the_basic_image_container.html