Procesamiento de imágenes digitales en dispositivos móviles

Buzzoni Ariel Nicolás, Corno Ezequiel,
Izaguirre Alan, Ramos Micaela, Secchi Lucas

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
aribuzz98@gmail.com, ezequiel.corno1996@gmail.com, alan.iza96@outlook.es,
ramosmicaela@outlook.es, luucass.secchi@gmail.com

Resumen. El reconocimiento de imágenes o reconocimiento de objetos, es una de las aplicaciones más importantes de la visión por computador y aunque existe una gran cantidad de literatura sobre el tema, aún no existe un algoritmo cien por ciento capaz de responder de manera precisa la gama de variaciones que puede afectar a una imagen, como cambios de iluminación, oclusión o puntos de vista diferentes. Esta investigación se basa en el uso del descriptor local ORB (por su velocidad de cómputo) para el reconocimiento de imágenes de las bebidas creadas en barman-iot y a partir de este procesamiento obtener información acerca de la bebida, utilizando la cámara de cualquier dispositivo móvil.

Palabras claves: Android,gpu,cuda,ORB.

1 Introducción

- Durante los últimos años, ha existido un creciente interés en el enfoque basado en la descripción de un objeto, utilizando descriptores locales. Los descriptores locales permiten detectar estructuras o puntos significativos (keypoint detection) en la imagen y obtener una descripción discriminante de estas estructuras a partir de sus alrededores.
- Uno de los descriptores locales que más influencia ha tenido en el campo de la visión por computador es ScaleInvariantFeatureTransform. Aunque el detector

y descriptor de puntos clave SIFT ha sido puesto a prueba en varias investigaciones y es ampliamente utilizado; el alto procesamiento computacional que requiere este descriptor ha sido el punto de partida para la creación de nuevos descriptores tales como Oriented FAST and Rotated BRIEF (ORB) los cuales se enfocan en mejorar la eficiencia y la precisión de correspondencias entre puntos clave con respecto a SIFT.

- El propósito principal de la investigación es presentar un estudio en el cual se utilizará el algoritmo ORB para la detección de objetos, mediante este obtendremos datos de la imagen tales como umbral, nitidez, contraste, luminosidad, puntos significativos de la imagen, entre otros y de esta forma se obtendrá información útil de la bebida que acaba de realizar.
- Actualmente en el mercado existen varias aplicaciones de reconocimiento de imágenes siendo una de las más conocidas Google Lens, otro ejemplo es CamFind que con solo tomar una foto obtienes información a partir de búsquedas en internet.

2 Desarrollo

Para el procesamiento de imágenes se pueden utilizar múltiples algoritmos. El que utilizaremos en nuestra aplicación será ORB. La manera de proceder será la siguiente, una vez finalizada la creación de la bebida se tomará una imagen de esta con la cámara del dispositivo móvil y se le aplicará el algoritmo antes mencionado.
ORB utiliza el algoritmo FeaturesfromAcceleratedSegment Test(FAST) para la detección de puntos claves y BinaryRobustIndependentElementaryFeatures(BRIEF) para la extracción de los descriptores. FAST y BRIEF son algoritmos que mejoran notablemente la velocidad de cómputo (lo cual es muy útil ya que estamos utilizando un dispositivo móvil), sin embargo carecen de un operador de orientación e invariancia a la rotación respectivamente. Para paralelizar el trabajo en el procesamiento de la imagen se utilizó cuda (Compute Unified Device Architecture), de esta forma obtenemos un compilar y un conjunto de herramientas que nos permite codificar algoritmos en una Unidad de Procesamiento Gráfico (GPU, de sus siglas en inglés) usando una variación del lenguaje de programación C. Utilizando de esta forma un elevado número nodos de procesamiento para realizar operaciones en paralelo sobre un gran volumen de datos bajo la arquitectura SIMT (Sigle Instruction Multiple Thread), similar al paradigma SIMD (Single Instruction Multiple Data), obteniendo una considerable reducción del tiempo de procesamiento y un gran aumento de las prestaciones. Además se utilizará la API OpenCV, biblioteca libre de visión artificial, esta api contiene wrappers, En el caso de OpenCV para Android, el wrapper se lo denomina JNI que nos facilitara la obtención de keypoints.

3 Explicación del algoritmo.

Lo fundamental del algoritmo es la extracción de keypoints, mediante el procesamiento de estos obtendremos información valiosa de la bebida. Para facilitar el uso del algoritmo ORB utilizaremos la clase cv::Features2D.

El siguiente pseudocódigo ejemplificará el uso de CUDA con el algoritmo ORB utilizando OpenCV.

```
#include <opencv2\opencv.hpp>

//obtenemos la imagen a procesar

Mat image = imread("../opencv-keypoints/bebida.jpg", IMREAD_GRAYSCALE);

//se sube la imagen a la gpu

cv::mat img = cv::imread("bebida.jpg");

cv::cuda::gpuMat gpu_img;

gpu_img.upload(bebida.jpg);

//reducimos el tamaño de la imagen

cuda::rezice(input,output,gpu_img..size(),25,30,AREA)

//obtenemos un puntero a un objeto de la clase ORB la cual implementa el algoritmo
del mismo nombre, lo hacemos a través del método

Ptr<Feature2D> detect = ORB::create();

/*luego usamos el método detect(...) para obtener el listado de los KeyPoint de
nuestra imagen de entrada, esta clase guarda toda la información correspondiente a
nuestro punto característico, coordenadas del punto, ángulo, etc.*/

vector<KeyPoint> kp;

detect->detect(image, kp);

/*finalmente para visualizar los puntos encontrados usaremos la función
cv::drawKeypoints(...) en la cual indicamos la imagen de entrada, el conjunto de
```

puntos que deseamos dibujar, la imagen de salida, el color a utilizar (-1 indica colores aleatorios)/*

Mat result;

drawKeypoints(image,kp,result,Scalar::all(-1),
DrawMatchesFlags::DRAW_RICH_KEYPOINTS);

*/*Una vez tenemos los puntos característicos de nuestra imagen podemos intentar localizar estos puntos en una segunda imagen, con ello podremos por ejemplo: combinar imágenes, realizar seguimiento y detección o en nuestro caso compararla a otra bebida.*

*/*para ello utilizamos BFMatcher (**BruteForceMatcher**) para comparar los descriptores de cada punto y obtener el pareo para cada uno de ellos, la clase base para los comparadores es DescriptorMatcher, al utilizar el método match(...) obtenemos una lista de DMatch. */*

vector<DMatch> matches;

Ptr<DescriptorMatcher> matcher = BFMatcher::create(NORM_HAMMING, true);

matcher->match(descA, descB, matches);

4 Pruebas que pueden realizarse

Las pruebas que pueden realizarse son todas referentes a sacar fotos de distintas bebidas, ya sea conocidas o mezclas que se acaban de inventar en el momento y a partir de estas que se obtenga información de relevancia para el usuario.

5 Conclusiones

- En esta investigación se incorporó el procesamiento de imágenes a nuestra aplicación barman-iot. Se investigó qué algoritmo era el más óptimo a utilizar teniendo en cuenta tiempo de ejecución, consumo de batería, uso de memoria y que todo este procesamiento iba a ser realizado en un dispositivo móvil. También como podemos paralelizar la información para hacerlo lo mas performante y rápido posible llegando a la conclusión que era fundamental utilizar tecnologías de GPU.
- La investigación nos deja como aprendizaje lo costosa que es la implementación de procesamiento de imágenes y cómo es sumamente necesario paralelizar la

información y conocer el algoritmo ideal y verificar donde este va a actuar para sacarle el mayor desempeño y provecho posible a las nuevas tecnologías que van surgiendo, sobre todo en el mundo de las tecnologías móviles..

- Para próximas investigaciones hay que mejorar el algoritmo ORB ya que, a pesar de que mejora notablemente la velocidad de cómputo para el procesamiento de imágenes, carece de un operador de orientación e invariancia a la rotación respectivamente lo que provoca que ciertos datos de la imagen no seas tomados correctamente. Por lo cual al algoritmo ORB deberíamos agregarle KeypointOrientation y Rotation-AwareBrief.

6 Referencias

1. Rublee. E, Rabaud. V, Konolige. K, Bradski, K. (2015) "ORB: anefficientalternativeto SIFT or SURF.
2. Mikolajczyk. K, Schmid.C, "A performance evaluation of local descriptors".IEEE Trans. Pattern Anal. Mach. Intell., 27:1615–1630, October 2017.
3. Batuhan Hangun, onder eyecioglu performance comparison between OpenCV built in CPU and GPU Functions on IMage Processing operations. International Journal Of Engineering Science and Application.Vol.1, No.2, 2017.