Papelera con seguimiento inteligente

Escobar Fabián, Estigarribia Emmanuel, Ferrarese Martín,
Tejerina Ezequiel, Vera Cristian

Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
fabiannorbertoescobar@gmail.com, emmanuelestigarribia@hotmail.com,
martin.ferrarese@gmail.com, ezequiel.tejerina95@gmail.com, crygvera@gmail.com

Resumen. El presente trabajo de investigación tiene la finalidad de añadir la funcionalidad de detección de movimiento y seguimiento de personas a nuestro proyecto *SmartTrash*, mediante el análisis de imágenes con visión artificial, haciendo uso del procesamiento en paralelo.

Palabras clave: HPC, SmartTrash, OpenCV, Detección de movimiento.

1 Introducción

El proyecto *SmartTrash* consta de la implementación de un sistema embebido a una papelera, con el objetivo de hacer más cómodo su uso, proveyéndole de funcionalidades tales como la apertura automática, detección y aviso de llenado, iluminación en base al estado del ambiente, entre otras cosas.

El objetivo de esta investigación es añadir a nuestro embebido la posibilidad de reconocer una persona en movimiento y seguirla en su trayecto, permitiéndole usar la papelera en dondequiera que lo necesite. A dicha funcionalidad la nombramos “Modo limpieza”.

Para esto, haremos uso del modulo 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 enviar los comandos a la papelera y que esta comience la persecución.

La GPU, a diferencia de la CPU, tiene como pilar el procesamiento en paralelo de datos[1], lo cual nos permite sacar ventaja de situaciones donde los algoritmos presentan gran complejidad o que manejan grandes cantidades de datos.

Podemos encontrar ejemplos de detección de movimiento con visión artificial en varios sistemas de seguridad, o sistemas que necesitan hacer un conteo de las personas que entran o salen de algún lugar, entre otras aplicaciones.

2 Desarrollo

Para aplicar la funcionalidad propuesta en esta investigación, como primera medida, necesitaremos integrar al tacho un juego de cuatro ruedas con sus correspondientes ejes y motores de corriente continua, y una cámara para obtener información de lo que se encuentra entrente de la papelera, la cual será procesada paralelamente permitiendo la generación de comandos para que nuestra papelera avance siguiendo a la persona detectada. La cámara enviará las imágenes captadas a la placa Arduino, y ésta, a su vez, al dispositivo Android para que sean procesadas.

Una vez activado el modo "Limpieza", en la aplicación del dispositivo Android, la cámara comenzara el envío de imágenes hacia la placa Arduino.

Para lograr la implementación utilizaremos la librería OpenCV[2], la cual contiene algoritmos de vision artificial o "computer vision"[3]. Estos algoritmos pueden usarse para reconocimiento facial, identificación de objetos o seguimiento de objetos en movimiento, entre otras cosas.

El proceso de detección de movimiento que se propone implementar está compuesto por 3 tareas: conversión a escala de grises y eliminación de ruido; sustracción entre el segundo y primer plano; y aplicar un umbral a la imagen resultado de la resta.

Dicho proceso se explica en la siguiente sección.

3 Explicación del algoritmo.

Empezaremos leyendo lo que capta la camara del embebido, y guardando cada frame en una matriz Mat.

```
// Leer el dispositivo de video, y leer cuadro por cuadro
VideoCapture video = cvProc.VideoCapture(camara);
Mat frame;
video.read(frame);
```

Teniendo ya cargada previamente la matriz que representa al frame del "fondo" (lo que será la imagen base para encontrar la persona en movimiento), empezamos con el tratamiento de cada frame, pasando por el proceso de las 3 tareas que anteriormente comentábamos.

Conversión a escala de grises y eliminación de ruido.

Antes de realizar ninguna operación con las imágenes, es conveniente convertir a escala de grises. Resulta menos complejo y más óptimo trabajar con este tipo de imágenes. El código de ejemplo está en lenguaje Java.

```
// Convertir la imagen a escala de grises
cvProc.cvtColor(frame, frameBN, cvProc.COLOR_BGR2GRAY)
```

Por otro lado hay que minimizar el ruido provocado por la propia cámara y por la iluminación. Esto se hace a través de promediar cada píxel con sus vecinos. Se conoce comúnmente como suavizado.

```
// Suavizado de la imagen
Size size = new Size(21, 21);
cvProc.GaussianBlur(frame, frameBN, size, 0);
```

Sustracción entre el segundo plano y el primer plano.

Lo que haremos será restar y obtener los valores absolutos de cada píxel, para no tener ningún número negativo.

```
//Resta en valor absoluto
Mat resta;
cvProc.(frameBN, fondo, resta);
```

Aplicación del umbral.

En esta parte del proceso lo que hacemos es quedarnos con aquellos píxeles que superen un umbral. El objetivo es binarizar la imagen es decir, tener dos posibles valores. Todos aquellos que superen el umbral (significa que no forman parte de la imagen de segundo plano, o fondo) serán píxeles blancos y los que no lo superen serán píxeles negros. Esto nos servirá para seleccionar el objeto en movimiento.

```
// Aplicamos el umbral a la imagen
Mat umbral;
cvProc.threshold(resta, umbral, 25, 255, cvProc.THRESH_BINARY);
```

Luego de estos tres pasos, se obtiene una imagen en blanco y negro, que muestra en color negro el fondo, y en blanco el objeto en movimiento.

Esto nos permite luego, en función de la posición de la silueta blanca representando a la persona en movimiento, procesar la dirección hacia dónde se deberá mover el sistema automatizado y enviarlo al sistema embebido para movilizar el sistema de ruedas en la dirección deseada.

La limitación en la primera fase de nuestra investigación es que como la cámara toma una sola imagen como valor de fondo, la persona en movimiento no deberá salirse del plano estático que puede capturar la cámara del embebido, y, como se puede esperar, este solo avanzará en línea recta (siempre que encuentre movimiento enfrente).

4 Pruebas que pueden realizarse

En primera instancia, la prueba que puede realizarse es la esperada para su correcto funcionamiento, con la persona caminando medianamente en línea recta, sin salirse del plano que puede captar la cámara del embebido.

Otra prueba posible sería empezar con el camino recto, y desviarse del plano que puede captar la cámara, para confirmar que efectivamente el embebido no se moverá porque no encontrará movimiento o "alguien a quien seguir".

5 Conclusiones

El resultado de la investigación y utilización de OpenCV en nuestro proyecto nos permite añadir una funcionalidad interesante, el seguimiento de una persona, haciendo uso del procesamiento paralelo que es posible gracias a la GPU de un dispositivo móvil.

Como lección aprendida tenemos el aprendizaje y medianamente adentramiento en lo que se conoce como visión artificial por computadora o "Computer Vision", tecnología que va encontrando cada vez más participación en la actualidad, principalmente en el campo de la videovigilancia.

En un futuro, lo que quisiéramos agregar al proyecto es la capacidad de reconocer movimiento haciendo un uso no estático de la cámara del embebido, lo que supone considerar muchos más factores que los que cubre esta investigación.

6 Referencias

1. John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, y James C. Phillips, "GPU Computing" (2008), p. 879-899, Proveniente de IEEE.
   https://ieeexplore.ieee.org/abstract/document/4490127

2. Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, Victor Eruhimov. “Real-time computer vision with OpenCV” (2012).
   https://dl.acm.org/citation.cfm?id=2184337

3. R. Szeliski. Computer Vision: Algorithms and Applications. Springer (2011).
   https://books.google.com.ar/books?hl=es&lr=&id=bXzAlkODwa8C&oi=fnd&pg=PR4&dq=R.+Szeliski.+Computer+Vision:+Algorithms+and+Applications.+Springer+2011&ots=g_562pzAJ&sig=uxSQddEBJNWOu2qtCl9g0Pik2A0#v=onepage&q=R.%20Szeliski.%20Computer%20Vision%3A%20Algorithms%20and%20Applications.%20Springer%202011&f=false