Detección de obstáculos usando visión monocular

Integrante:
Nicolás Furquez
Tutores:
Mercedes Marzoa
Gonzalo Tejera

January 2, 2018

1 Introducción

Uno de los problemas mas comunes dentro de la población ciega o de baja visión que se da al momento de transitar por la vía publica es el no poder 'ver' aquellos obstáculos que pueden estar por sobre la cintura, dado que no están dentro del rango del barrido del bastón que usan como se puede ver en la Figura 1.

Figure 1: Problemas a los que se enfrentan los ciegos en la vía publica

Dentro de esta problemática se plantea la posibilidad de generar un dispositivo que permita dar un aviso al portador, de que se aproxima a un obstáculo y él mismo pueda tomar las medidas necesarias para esquivarlo.

2 Requerimientos

- Portabilidad: Se considera necesario que el dispositivo tenga la capacidad de ser portado de manera cómoda por el usuario, por lo que tiene que ser de un tamaño y peso acordes, para que sea fácil de llevar.
- Accesibilidad: Es deseable que el dispositivo sea accesible económica y tecnológicamente. Una realidad es que si el dispositivo es caro, no seria factible para el usuario tenerlo o mantenerlo, así como también debería de ser accesible tecnológicamente, esto es que o se pueda desarrollar con materiales encontrado en el mercado Uruguayo o que no sea complicado de replicar dado que necesitaría de mano de obra de un tercero para poderlo tener.
- Robustez: Debe de poder ser capaz de detectar obstáculo en la mayor cantidad de escenarios posibles.
- Confiabilidad: Debe de poder ser fiable a la hora de detectar obstáculos y tener baja taza de falsos positivos.

3 Opciones

Para crear al dispositivo es posible de contar con las siguientes tecnologías, dentro de cada etapa del procesos de detección:

- *Sensado*
  - Ultrasonido
  - Sharp
  - Láser
  - Visión
    - Cámara
    - Cámara Estéreo
    - Cámara RGB + sensor de profundidad (ej. Kinect)

- *Computo*
  - SBC
  - Smartphones
- *Actuación*
  - Sonido (pitido)
  - Vibración
  - Auriculares de conducción Ósea.

4 Camino elegido

Una posibilidad de desarrollar este dispositivo es el de usar un teléfono celular o Smartphone. El mismo cuenta con gran poder de calculo, es relativamente barato, y algo importante es que la mayoría de los impedidos visuales cuentan con uno y lo manejan en menor o mayor medida. Quizás las desventajas mas destacables son que, el general de los celulares modernos no cuenta con gran duración de batería y que solo poseen una cámara.

5 Alcance dentro de Taller de sistemas Ciber-Físicos

Se planteo como alcance dentro del taller y teniendo en cuenta la realidad planteada, investigar la posibilidad de detectar obstáculos para el usuario usando visión monocular, o sea con una cámara sola, como la que tendría usando un celular como unidad de computo.

Se pretende también generar una salida que pueda ser utilizada dentro del contexto planteado.

6 Detección usando visión monocular

Se encontraron varias publicaciones dentro del área de asistencia a discapacitados visuales en el que se plantea el uso de Smartphones como unidades de computo, entre las consultadas [4],[6], [8] y [9] . En particular se encuentran dos publicaciones *"A Smartphone-Based Obstacle Detection and Classification System for Assisting Visually Impaired People"*[9] y *"When ultrasonic sensors and computer vision join forces for efficient obstacle detection and recognition"*[8] publicados por el mismo grupo de investigadores, donde se plantea el uso de la cámara del celular como fuente de información del ambiente para la detección de obstáculos. Dada la información encontrada en estas publicaciones se pretende replicar lo presentado en las mismas, dado que: corre dentro de un Smartphone y no depende de internet para hacer el proceso de las imágenes, no usa métodos de aprendizaje de alto computo para la detección, no hay etapa previa de entrenamiento.

7 Ambiente de desarrollo

El ambiente elegido para desarrollar es ROS en un PC usando como fuente de entrada vídeos o secuencias de imágenes, se comenzó con una secuencia de imágenes tomadas del Dataset encontrado en [2], a su vez se filmaron recorridas en ambientes locales. Con este conjunto de imágenes se abarcaba gran cantidad de escenarios.

Como lenguaje de programación se utiliza Python y para el proceso de imágenes la biblioteca OpenCv [1], dado que se contaba con experiencia en ambos puntos y en conjunto con ROS, se puede trabajar mas rápidamente al programar en Python.

8 Flujo de detección

El proceso se divide en etapas, las cuales se pueden ver en la figura 2.

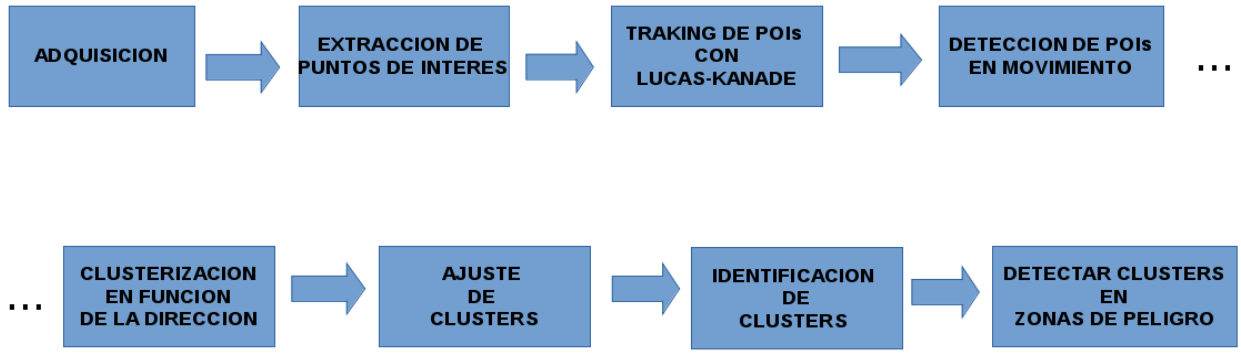

Figure 2: Flujo del proceso de detección de obstáculos

A continuación se describen las etapas mas importantes del flujo.

8.1 Adquisición de imagen

La Adquisición de imágenes es en realidad un Nodo ROS, el cual toma una imagen de la secuencia seleccionada (o cuadro en el caso de un vídeo) y es enviado al nodo que hace el procesamiento.

Cabe destacar que previo al proceso del cuadro se le hace un filtro pasa bajos a la imagen para suavizarla y asi eliminar ruido, dicho filtro se realiza por un filtro Gaussiano.

8.2 Extracción de puntos de interés

Con punto de interés (POI de ahora en mas) nos referimos a píxeles en la imagen que poseen determinada característica que permite volverlo a encontrar ya sea que la imagen este en otra posición o solo se tenga una sección de ella.

Para la extracción de POI en [9] se plantea una grilla regular en función de las dimensiones de la imagen y la cantidad de puntos a encontrar, en ella se seleccionaba un píxel en función de un detector de esquinas de Harris [5]. En la practica este mecanismo no generaba buenos resultado en etapas posteriores del flujo, por lo que se descarto como detector de POI's.

El mecanismo que se selecciono de los tantos que existen para esta tarea, fue ORB [3] dado que según se encontró en diversas publicaciones, este algoritmo es el mas eficiente y mas rápido en este tipo de tareas. Por contra posición ORB (como otros) se basan en la composición global de la imagen, lo que da que los puntos se aglomeren en ciertas zonas de la imagen, el sistema de grilla por el contrario dispersa mas la ubicación de los puntos al hacer la búsqueda local a la zona de la grilla. En la figura 3, se puede ver la diferencia de ambos métodos.

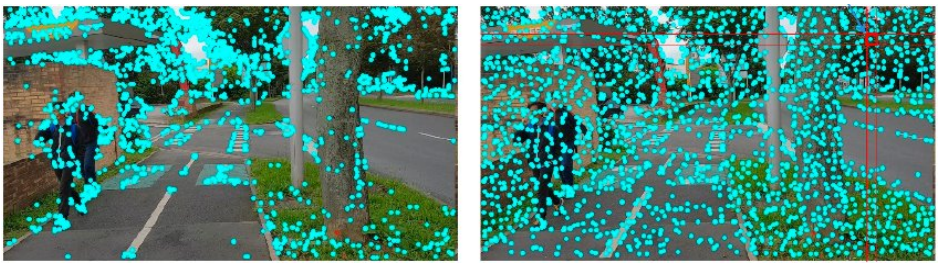

Figure 3: Diferencias entre un algoritmo como ORB y la grilla regular

El resultado de aplicar la detección de POI se puede ver en la figura 4, en donde los puntos verdes representan los POI's encontrados en la imagen. La cantidad a encontrar en cada iteración esta limitada a 1000 puntos, pero es ajustable.

Es importante destacar que la extracción de puntos, no se hace cada vez que se adquiere un cuadro, sino que se hace cada cierta cantidad de cuadros, siendo esta periodicidad ajustable. Esto ayuda a hacer mas rápido el sistema dado que, a pesar de ser rápido, es un proceso pesado y en pocos cuadros no cambiara mucho el contexto. La extracción también es disparada cuando no se encuentran puntos en movimiento (ver sección 8.4)

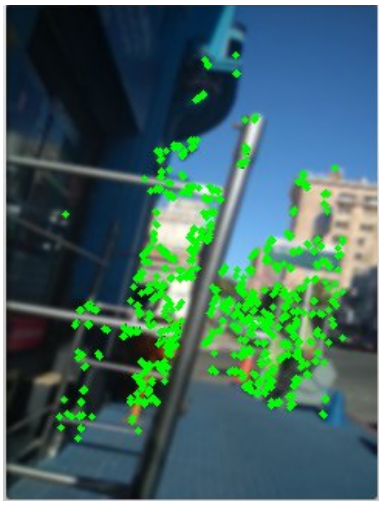

Figure 4: Posición de los POIs obtenidos en la imagen

8.3 Tracking de POI's

El tracking o seguimiento de los puntos de interés, hace referencia a la posibilidad de determinar como se movió un punto de una imagen a la siguiente, también llamado Flujo Óptico. El algoritmo usado es el de Lucas-Kanade [7]. Este algoritmo toma la imagen y la posición de los puntos en el cuadro anterior, junto con la imagen del cuadro actual y devuelve una lista de la posición de los puntos en la imagen actual.

Luego son eliminados los puntos que no pudieron ser encontrados haciendo el camino inverso, de esta manera se eliminan puntos que "salieron" de la imagen.

8.4 Detección de puntos en Movimiento

Con puntos en movimiento se hace referencia a los puntos que están relativamente en movimiento ya que la cámara se esta moviendo y por lo tanto el fondo u objetos estáticos están en moviendo aparente. Dado esto se identifican dos grupos de puntos:

- Los puntos que pertenecen al fondo y se están moviendo pero en realidad es a causa del movimiento de la cámara.
- Los puntos que pertenecen al primer plano y que son los objetos que se están acercando mas rápido que los puntos del otro grupo. En esta categoría también entran los puntos de los objetos que efectivamente se están moviendo en la escena (ej. peatones, autos, etc.).

Para determinar que puntos están en que grupo, lo que se hace primero es determinar una transformación, llamada homografía, que dado un punto del cuadro anterior, nos devuelve su imagen en el cuadro actual. Para determinar esta transformación, se utilizan todos los POIs encontrados en el cuadro anterior y su posición en el actual. Usando una funcionalidad de OpenCv podemos encontrar la mejor homografía en función de los POIs.

Una vez obtenida la homografía H, se aplica H a todos los puntos del cuadro anterior, dándonos así algo que llamaremos la posición estimada de esos puntos en el cuadro actual p^{est}. Luego de obtener este punto podemos determinar el error $E_r(p^i)$ entre el punto estimado p^{est} y el la posición del punto en el cuadro actual p^i, el cual se calcula:

$$E_r(p^i) = ||p^{est} - p^i||$$

Donde $||.||$ hacer referencia a la Norma dos vectorial. Si $E_r(p^i)$ es mayor a un determinado umbral, el punto p^i pertenece a un objeto que se está moviendo o está en primer plano. Estos puntos son los que serán denominados como puntos en movimiento y son el insumo de las siguientes etapas. En esta etapa también se genera como salida información del vector de movimiento (ángulo y magnitud).

Mocanu et al. (2016) [8] desarrolla una mejora a este procesamiento dividiendo la imagen en cuadrantes y usar cuatro homografías, una por cada cuadrante y quedarse con los puntos que cumplen que el $E_r(p^i)$ esta por sobre el umbral en todas. En este desarrollo no genero buenos resultados, por lo que se mantuvo la idea de una homografía global.

Figure 5: Salida de la etapa de detección de POIs en movimiento, comparándola con los puntos de la etapa anterior

8.5 Clusterización

El proceso de clusterizar consiste en agrupar los puntos de la etapa anterior de tal manera de que pertenezcan al mismo objeto, en este caso no solo es necesario agruparlos por proximidad espacial (sean cercanos) sino que también es necesario tener en cuenta la componente de dirección y magnitud del movimiento. Para contemplar ambas componentes se utiliza lo propuesto en [8], donde se propone una transformación no lineal de los puntos, pasando los mismos de un sistema cartesiano a uno polar de la siguiente manera:

$$v_{ix} = d \cdot cos(\alpha); v_{iy} = d \cdot sen(\alpha)$$

Donde d es calculado de la forma:

$$d = 1 + \frac{d_i}{d_{max}}$$

con d_i es la magnitud del movimiento, y d_{max} es el máximo de las magnitudes de todos los puntos en movimiento y α el ángulo del movimiento. Luego de hacer el cambio de coordenadas se usa como algoritmo de clusterización *K-means* utilizando los puntos transformados, donde se busca como máximo 10 clusters. En la figura 6 es posible ver como quedan los puntos de la imagen en el plano polar, y se agrupan los puntos que no solo están juntos sino que también se mueven de la misma forma.

Figure 6: Representación de los puntos en polares / Posición de los puntos en la imagen

Dado que clusters muy cercanos pueden pertenecer en realidad a un mismo cluster, se agrupan los mismos en función de que el centro y el vector de dirección promedio sean similares.

De esta etapa se tiene información de que puntos pertenecen a que cluster y también la información del centro del mismo y su vector de movimiento promedio.

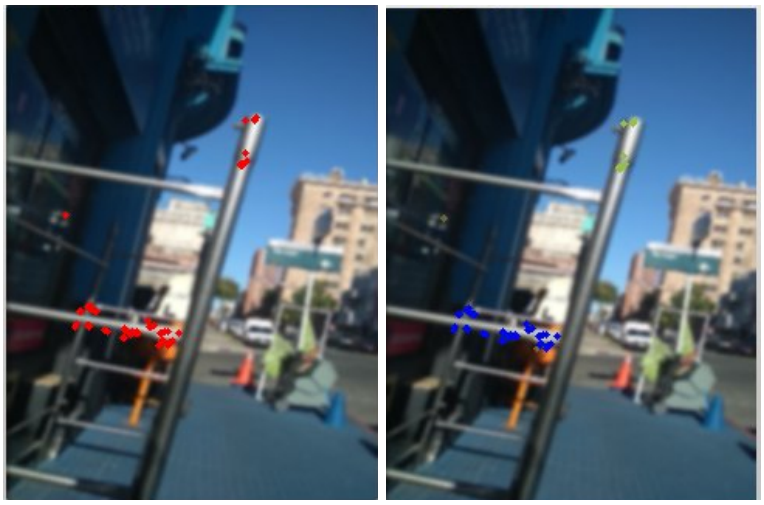

Figure 7: Se pueden ver los puntos en moviendo de la etapa anterior con distintos colores, cada uno representando un cluster.

8.6 Identificación de clusters

Dado que de la etapa anterior es posible obtener información de los clusters que se encontraron, en esta etapa se trata de identificar a que cluster ya detectado en los cuadros anteriores corresponden o si son nuevos. La información que se mantiene es bastante simple y es el centro y vector de movimiento promedio, por lo tanto para identifican si ya existe el cluster se compara que la ubicación de centros y el vector de movimiento sean similares a alguno de los ya encontrados en cuadros anteriores. Si se encuentra una similitud se actualiza

posición y vector de movimiento del cluster, sino se crea como un cluster nuevo en la estructura. En la figura 8, se muestran los centros de los clusters que se encontraron.

(a) Puntos clusterizados (b) Centro de los clusters

Figure 8: Se pueden ver en la imagen b los centros de los clusters encontrados en la etapa anterior (imagen a)

8.7 Aviso de objetos en zonas criticas

En esta etapa se trata de identifican cuales de los clusters están dentro de ciertas zonas que se denominaron criticas. Éstas son una proyección estimada en la imagen, de las zonas físicas donde podrían haber obstáculos, que de seguir en esa dirección, el usuario podría chocar con ellos. Si se encuentran centros de clusters dentro de esas zonas por mas de dos cuadros, entonces se emitirá una alarma sonora y el usuario podrá decidir que acción tomar.

9 Problemas encontrados

- Mecanismo para determinar POI's: Primeramente al tener solo la publicación [9], se tardo mucho tiempo en entender a que se refería con grilla regular dado que no ahondaba en el concepto en la publicación y no se encontraba material al respecto. Luego al poder dar con la segunda publicación [8] fue posible entender el mecanismo de construcción y se implemento el mismo, pero se llego a que no daba buenos resultados para etapas siguientes.
- Estabilidad entre cuadro y cuadro: Al utilizar la grilla se perdían puntos a veces de un cuadro a otro, incluso si se demoraba en disparar el mecanismo de detección de POIs. Se encontró que sacando la grilla parte del problema se soluciono. Igualmente, como se vio antes, el sistema de grilla puede dar mejores puntos dado que los distribuye uniformemente.

- Determinación de los POIs en movimiento: Se encontró que en [8] se hacia un ajuste para esta etapa generando 4 homografía, pero se encontró que daba mejores resultados usar una. Igualmente el mecanismo lanza falsos positivos o descarta puntos útiles.

10 Trabajo a futuro

- Mejorar el mecanismo para detectar puntos en movimiento: El mecanismo funciona pero descarta puntos útiles así como tampoco es estable, puede detectar un punto como en movimiento en un cuadro pero eliminarlo al siguiente.
- Mejoras al clusterizar: Encontrar un mejor mecanismo para clusterizar y que pueda tener en cuenta tanto la componente espacial como la del movimiento, el usado no es muy robusto, también tener un mejor sistema para determinar clusters en el tiempo, o sea, identificar si el cluster encontrado es efectivamente otro anterior o es uno nuevo.
- Flujo de clusters: Teniendo mas información o mejor identificados los clusters que se están viendo se podría estimar hacia donde se mueven y el sistema puede ser mas robusto al momento de tener una oclusión.
- Determinar si un objeto esta mas cercano a otro: Seria interesante tener la capacidad que objeto efectivamente se esta acercando mas rápido que otro, una posibilidad que se exploro, pero no se logro completar es utilizar la vecindad de Delauny, como se plantea en [4].
- Hacer pruebas en hardware similar a Smartphones.

11 Conclusiones

La principal conclusión a la que se llega con este trabajo, es que sí es posible detectar obstáculos usando visión monocular, dentro de los requerimientos que se plantean en la sección 2. No obstante es necesario hacer mejoras en varias de las etapas del flujo, dado que se observan varias inestabilidades, quizás agregando algún mecanismo de aprendizaje automático de bajo consumo de recursos y que no dependa de internet.

Con respecto al uso en la realidad planteada, queda inconcluso, dado que se eligió experimentar dentro de un ambiente controlado y tampoco se utilizo el lenguaje que puede ser utilizado dentro de los Sistemas Operativos mas comunes en teléfonos celulares de hoy día. Pero en las publicaciones de referencia se habla de celulares de alta gama de hace unos años que hoy en día pueden aparejarse con celular de gama media, por lo que podría decirse que puede ser factible.

References

[1] Opencv library. https://www.opencv.org/.

[2] Robust multi-person tracking from mobile platforms. https://data.vision.ee.ethz.ch/cvl/aess/dataset/.

[3] Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision, Computer Vision (ICCV), 2011 IEEE International Conference on*, page 2564, 2011.

[4] *Collision Detection for Visually Impaired from a Body-Mounted Camera*, Portland, OR, 2013.

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988.

[6] E. Ko and E. Y. Kim. A vision-based wayfinding system for visually impaired people using situation awareness and activity-based instructions. *Sensors*, 17(8), 2017.

[7] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[8] B. Mocanu, R. Tapu, and T. Zaharia. When ultrasonic sensors and computer vision join forces for efficient obstacle detection and recognition. *Sensors (Switzerland)*, 16(11), 2016.

[9] R. Tapu, B. Mocanu, A. Bursuc, and T. Zaharia. A smartphone-based obstacle detection and classification system for assisting visually impaired people. June 2013.