ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO INDUSTRIAL

POSICIONAMIENTO EN INTERIORES BASADO EN LUZ VISIBLE (VLC) MEDIANTE LA MANIPULACIÓN DE LA FRECUENCIA

Autor: Miguel Martinez de Salinas Bassols
Director: David Contreras Bárcena

Madrid

Julio 2016

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Miguel Martínez de Salinas Bassols

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Posicionamiento en interiores basado en luz visible (VLC) mediante la manipulación de la frecuencia que esta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ...19.... de .....Julio..... de ..2016

ACEPTA

Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Posicionamiento en interiores basado en luz visible (VLC) mediante la manipulación de la frecuencia

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2015-2016 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Miguel Martinez de Salinas Bassols Fecha: 19 / 07 / 2016

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: David Contreras Bárcena Fecha: 18 / jul / 2016

Vº Bº del Coordinador de Proyectos

Fdo.: Álvaro Sánchez Miralles Fecha: 18 / jul / 2016

POSICIONAMIENTO EN INTERIORES BASADO EN LUZ VISIBLE (VLC) MEDIANTE LA MANIPULACIÓN DE LA FRECUENCIA

Autor: Miguel Martínez de Salinas Bassols
Director: David Contreras Bárcena

Resumen del Proyecto

Introducción

Hasta ahora se ha conseguido posicionar un dispositivo móvil usando satélites en el exterior haciendo que la población se pueda mover con mayor facilidad de un punto a otro y aportando información que posteriormente se puede analizar y de la que se puede sacar provecho. Sin embargo, hasta ahora no se ha conseguido localizar un dispositivo móvil en un espacio cerrado. Ésto ha demostrado ser un problema que alcanza cierta complejidad pues los sistemas no parecen ser fiables en todos los entornos, en este sentido se ha podido observar como el mobiliario, la temperatura o los usuarios dentro de un espacio cerrado generan grandes interferencias que reducen la fiabilidad. De este modo las mejoras en los sistemas de posicionamiento en interiores proveerían grandes posiblidades para los negocios además de otros usos en distintos sectores.

Objetivos

En este proyecto se pretende desarrollar el posicionamiento de un dispositivo móvil en superficies cerradas mediante la iluminación, tambien conocido como VLC o *Visible Light Communication*. Como todavía no hay una tecnología que esté considerada como la más idónea para lograr este objetivo, puede resultar interesante desarrollar la tecnología VLC porque no es intrusiva y todos los locales cerrados la necesitan.

Los documentos que se han estudiado durante este proyecto utilizan sensores externos al teléfono móvil [Luo et al., 2014], con más precisión y frecuencia de muestreo que el sensor de luz de los dispositivos, o los dispositivos con las mejores cámaras fotográficas del mercado[Kuo et al., 2014]. Este proyecto se va a centrar en el desarrollo de una solución al posicionamiento en interiores utilizando unicamente los sensores y herramientas disponibles en un dispositivo de gama media.

Los objetivos marcados para el proyecto son los siguientes:

1. Crear un algoritmo de posicionamiento respecto a una fotografía de referencias conocidas.

2. Crear un LED con un ciclo de encendido y apagado (PWM) susceptible de modificación.

3. Crear un algoritmo que identifique la frecuencia de la luz con un dispositivo

móvil.

Metodología

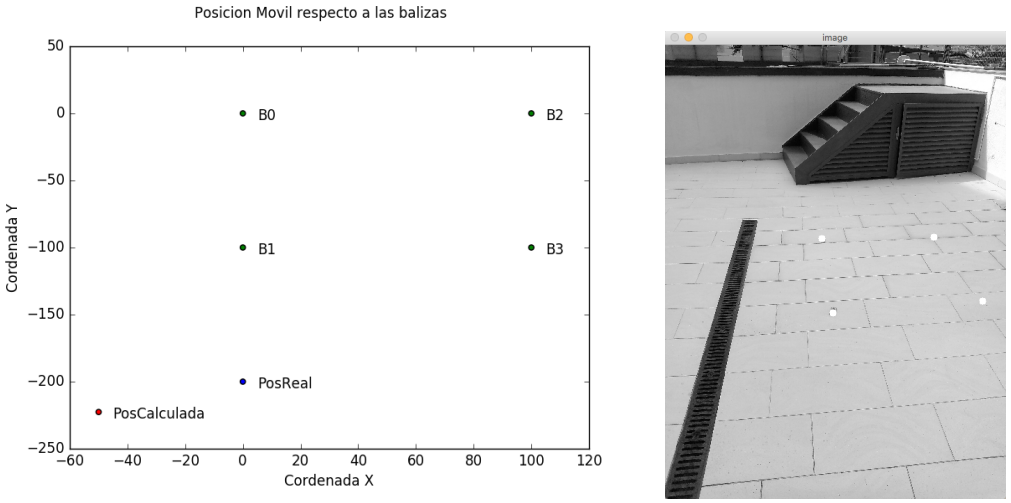
La metodología que se ha llevado a cabo para realizar el proyecto no ha sido el mismo en todas las secciones desarrolladas. Para los primeros apartados, donde se han estudiado tanto las diversas posibilidades que ofrece el posicionamiento en espacios cerrados como las diferentes tecnologías para ello, se ha realizado un estudio intensivo de documentos de investigación, noticias e información de compañias relevantes en cada sector.

En las secciones principales del proyecto, donde se han desarrollado herramientas experimentales para la localización respecto a referencias y para identificar la frecuencia de un LED, se ha utilizado una metodología iterativa donde se estudiaba el problema, se creaba una soución y se utilizaban los resultados obtenidos como *feedback* para mejorarla.

Resultados

Posicionamiento respecto a referencias

El primer problema al que se ha intentado buscar una solución ha sido el posicionar un dispositivo móvil utilizando una fotografía tomada por la cámara en la que aparecen 3 o más referencias. Los resultados obtenidos muestran muy poca precisión al posicionar el dispositivo móvil y además no es robusto, pues en el momento que el usuario desplaza el télefono unos centrimetros el resultado varía de forma drástica, en la figura 1 se muestra un ejemplo de la localización.



(a) Ejemplo de resultado (b) Ejemplo de fotografía

Figura 1: Ejemplo de posicionamiento utilizando referencias. Elaboración propia.

Identificación de las luces con el sensor de luz del dispositivo

Los resultados al intentar identificar la frecuencia utilizando el sensor han sido infructuosos pues los datos recopilados por el sensor no se podían interpretar ya que no existía relación alguna con dicha frecuencia, dichos datos se encuentran en la figura 2. Todo esto se debe a que el sensor del móvil tiene una velocidad de muestreo muy inferior a 100 Hz y por lo tanto es incapaz de recoger señales con esa magnitud de frecuencias.

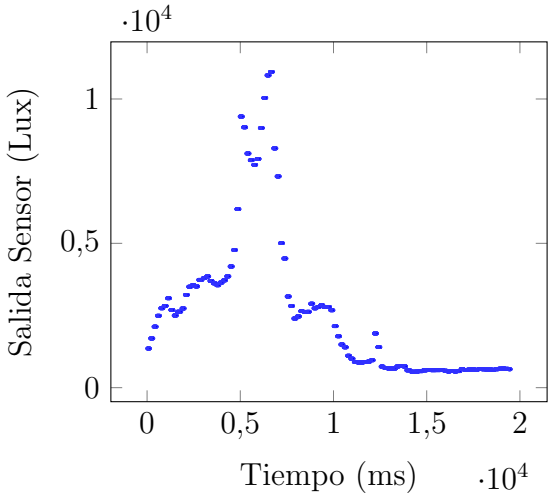
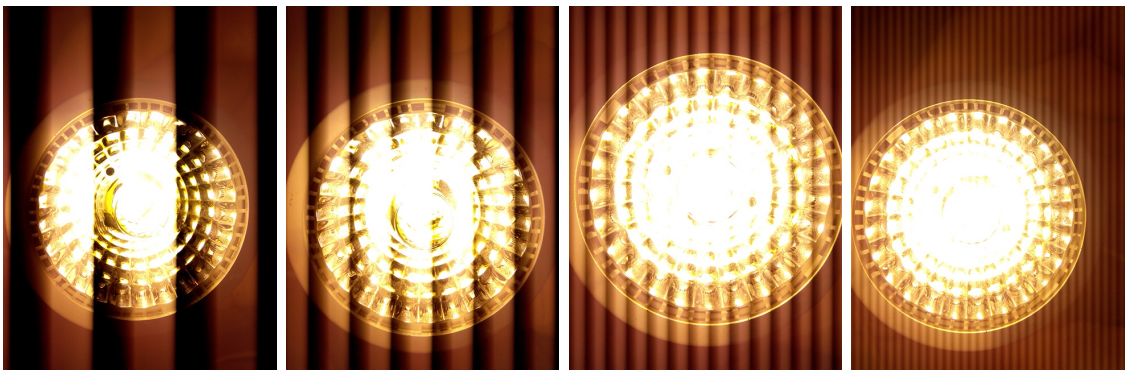


Figura 2: Diagrama de puntos donde se muestra la iluminación recogida por el sensor de luz a 100 Hz. Elaboración propia.

Identificación de las luces con la cámara fotográfica del móvil

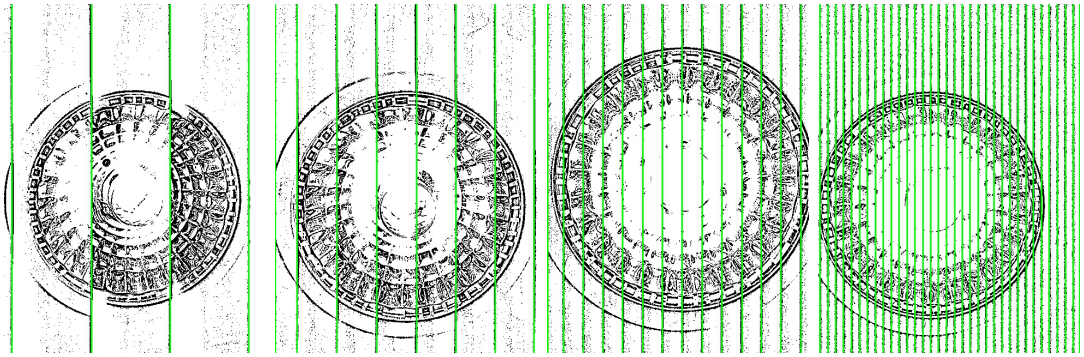
Los resultados de utlizar la cámara fotográfica para identificar el PWM de la iluminación han sido muy positivos. Tan solo analizando las fotografías sin realizar modificación alguna, se observa que hay una relación entre la frecuencia y la imagen (figura 3), donde los periodos (ancho de una banda encendida y una apagada) se van haciendo más estrechos a medida que aumenta la frecuencia.



(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

Figura 3: Fotografías tomadas por el télefono móvil a distintas frecuencias. Elaboración propia.

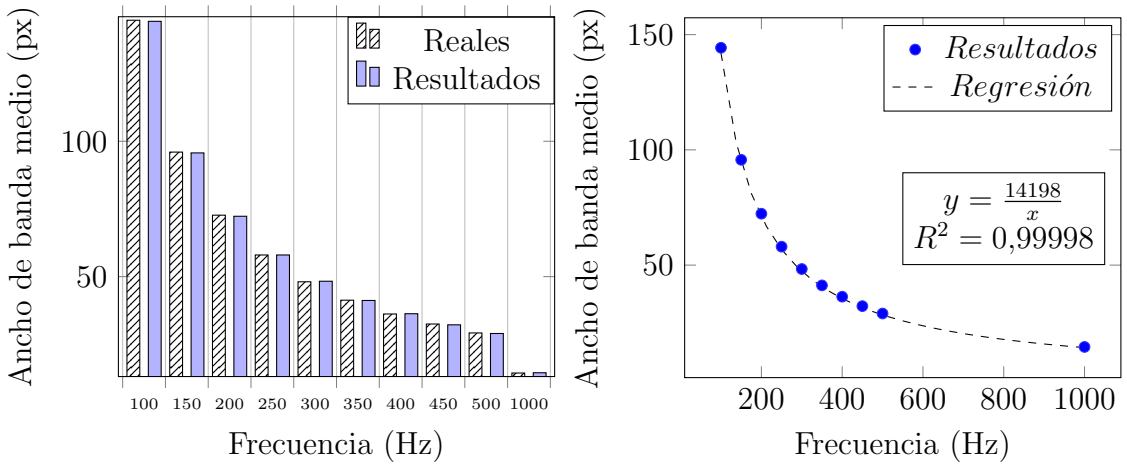
A partir de ahí se ha procedido a realizar diferentes transformaciones a las imágenes para intentar identificar de forma automática la frecuencia de la iluminación LED. Despúes de varios procesos se han conseguido obtener unos resultados significativos utilizando la funcion filtro adaptativo de Gauss y Hough Lines.



(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

Figura 4: Resultado final del proceso. Elaboración propia.

Los resultados de este proceso, visibles en la figura 4 (las líneas verdes representan las rectas detectadas por el algoritmo), son extremadamente buenos. El algoritmo tiene un error casi nulo si se comparan con los resultados reales obtenidos manualmente (figura 5a). Además, existe una alta relación entre el incremento de la frecuencia y el grosor de los periodo. De este modo, la ecuación de regresión potencia mostrada en la figura 5b tiene un coeficiente de determinación del 0.99998 por lo que es correcto afirmar que el ancho de los periodos (banda encendida más banda apagada) y la frecuencia de la iluminación están relacionadas con dicha ecuación para el teléfono móvil utilizado.



(a) Diagrama de barras Frecuencia-Ancho de Banda con datos reales y resultados.

(b) Diagrama de puntos Frecuencia-Ancho de Bandas de los resultados.

Figura 5: Resultados gráficos. Elaboración propia.

Conclusiones

La conclusión extraída después de realizar el proyecto es que es posible localizar un dispositivo móvil mediante la iluminación.

El posicionamiento respecto a las referencias conocidas en el espacio no parece viable debido a los resultados obtenidos. La conclusion que podemos extraer de ello es que la calidad del teléfono utilizado no permite identificar con precisión las referencias.

El montaje de un sistema de iluminación con frecuencia modificable nos ha permitido tanto comprobar el efecto que tiene dicha frecuencia en las fotografías como descubrir que existe una relación entre la frecuencia y el ancho de los periodos.

Los resultados al identificar la frecuencia de los LEDs utilizando la cámara fotográfica del teléfono móvil han sido precisos y robustos. Por ello parece interesante seguir investigando por esta línea para poder posicionar el dispositivo móvil en una situación real.

Para trabajos futuros sería interesante poder estudiar la utilización de iluminación con tecnología Zigbee, ya que la iluminación desarollada en el proyecto no funcionaría en un local interior debido a la corriente que utiliza. Tambien sería interesante investigar el efecto que tiene distintas condiciones como la distancia del móvil a la iluminación o la inclinación en los resultados.

Bibliografía

[Kuo et al., 2014] Kuo, Y.-S., Pannuto, P., Hsiao, K.-J., and Dutta, P. (2014). Luxapose: Indoor positioning with mobile phones and visible light. Technical report, University of Michigan.

[Luo et al., 2014] Luo, P., Ghassemlooy, Z., Minh, H. L., Khalighi, A., Zhang, X., and Yu, C. (2014). Experimental demonstration of an indoor visible light communication positioning system using dual-tone multi-frequency technique. Technical report, Northumbria University, Ecole Central Marseille, Beijing University of Posts & Telecommunication, National University of Singapore.

INDOOR POSITIONING USING VISIBLE LIGHT (VLC) THROUGH THE MODIFICATION OF THE LIGHTNING

Author: Miguel Martínez de Salinas Bassols
Director: David Contreras Bárcena

Project summary

Introduction

Technology nowadays is able to locate a smartphone in the open air using satellites, this has transformed the way that people move from one place to another and gives information that can be analysed and used. Nevertheless, indoor positioning has not been achieved. The complexity of the problem is big because the systems being used nowadays are not reliable in all the scenarios, in this sense the furniture in the room, the temperature or the number of people inside generate big interferences that reduce the reliability. An improvement in the reliability of indoor location would bring great opportunities for businesses and other sectors.

Objectives

The objective of this thesis is to develop the smartphone indoor location using lightning, also known as VLC or Visible Light Communication. There is still no leading technology in this quest, and lightning seem advantageous given that it is none intrusive and all indoor places need to have an illumination system.

The papers that have been studied during the thesis used sensors that were not included in smartphones[Luo et al., 2014], with a higher precision and sampling rate, or smartphones with the best camera in the market[Kuo et al., 2014]. This thesis is going to centre in the development of a solution using only the sensors and tools included in a low budget smartphone.

The objectives stablished for the thesis are the following:

1. Design a location algorithm that uses a photograph with known references.

2. Design a LED with a programmable ON and OFF cycle.

3. Design an algorithm that identifies the frequency of a LED using a smartphone.

Methodology

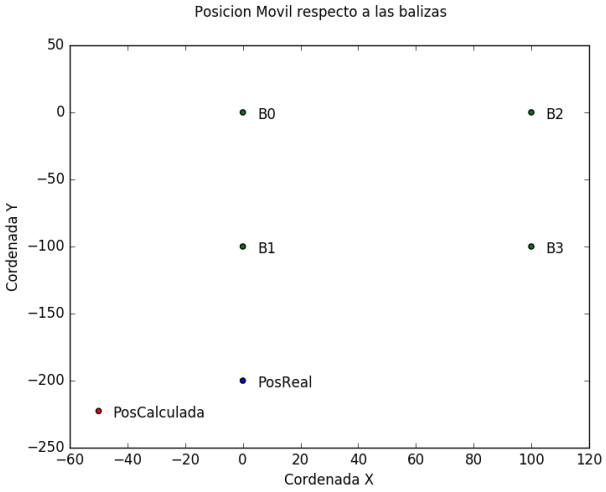
The methodology used has changed along the different sections. For the first sections, where the opportunities of indoor location and the technologies being used have been studied, an intensive research of relevant papers, articles, news and companies developing the technologies has been done.

In the main sections of the thesis, where experimental tools have been developed to locate using known references and to identify a LED's frequency, an interative methodology has been used where the problem was studied, a solution was developed and the results of the solution were used as feedback to improve the solution.

Results

Positioning using references

The first problem faced has been to locate a smartphone using a photograph where 3 or more references appeared. The results obtained show that the algorithm created has a very bad precision and robustness, when the device moves slightly the results change drastically. An example of the location result can be seen in 1.



(a) Result example

(b) Photograph example

Figura 1: Example of positioning using known references. Own elaboration.

Lightning identification using the smartphone's light sensor

The results when trying to identify the frequency with the smartphone's light sensor have not been as expected. Understanding the data was not possible because no pattern or relationship with the LED's frequency was found, the data is represented in figure 2. The problems found are due to the smartphone's sensor having a sampling rate lower than 100 Hz and therefore is incapable of recovering frequency signals of that magnitude.

Identification of a LED using the smartphone's camera

The results when using the smartphone's camera to identify the LED's PWM cycle have been very successful. Just looking at the unmodified pictures, a relationship can be seen between the LED's frequency and the image. In figure 3 it can be seen

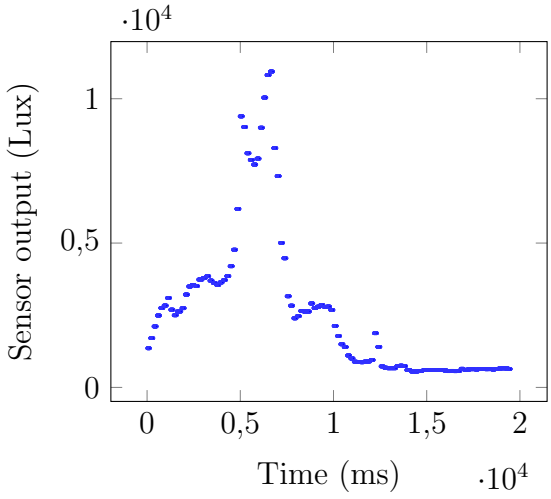
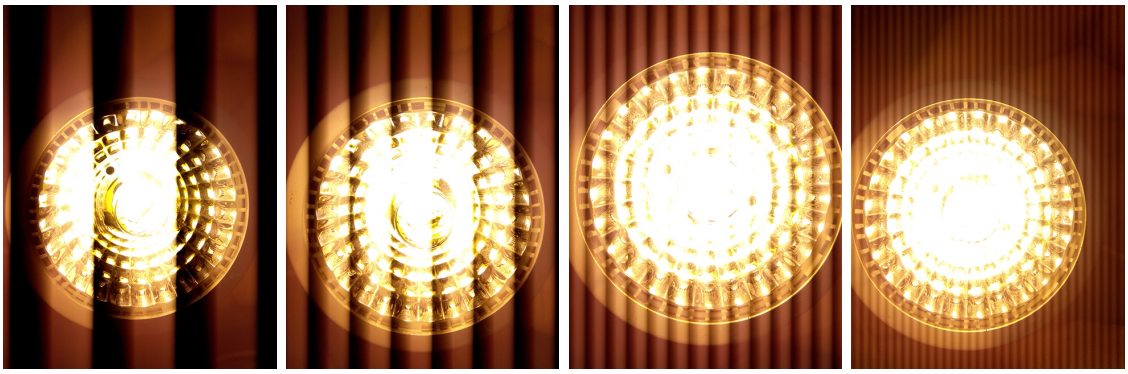


Figura 2: Scatterplot of the light registered by the smartphone's light sensor using a LED with 100 Hz. Own elaboration.

how the periods(width of an Off band and On band) get smaller as the frequency gets bigger.

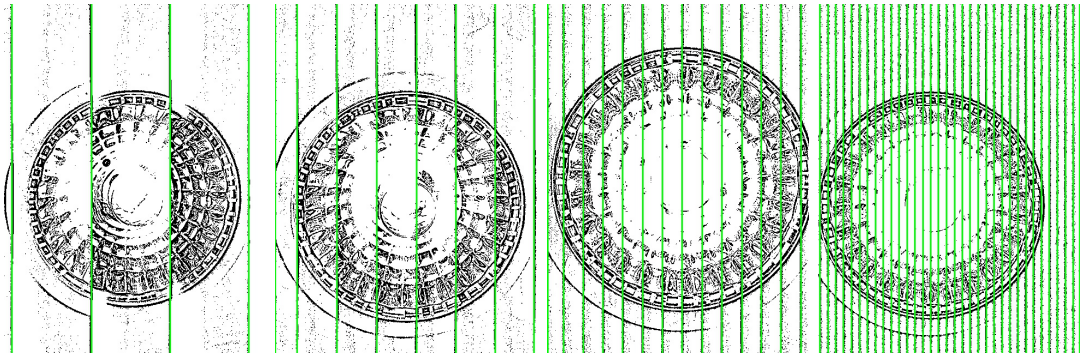


(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

Figura 3: Fotografías tomadas por el télefono móvil a distintas frecuencias. Own elaboration.

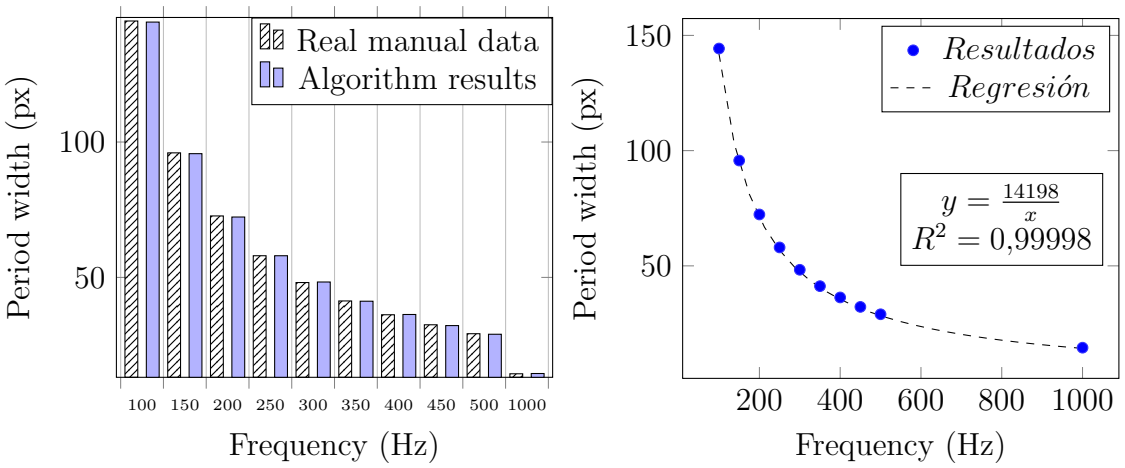
Once the unmodified images seemed to make sense a number of different transformations were applied to the image to try to automatically identify the frequency of the LED. After several processes a successful result was achieved when using Gauss adaptive filter and Hough Lines transform.

As it can be seen in figure 4(where the green lines represent the lines detected by the algorithm) the results are impressive. The algorithm has an error near to 0 when they are compared with the manually measured results (figure 5a).Besides, a very strong correlation can be seen between the frequency and the width of the periods. Looking deeper into the results, the potencial regression equation shown in figure 5b has a determination coefficient of 0.99998 and therefore it is fair to say that the width of the periods and the frequency are related with that equation for the used smartphone.



(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

Figura 4: Final result of the process. Own elaboration.



(a) Frequency Period width bar diagram with real and algorithm results. Own elaboration.

(b) Frequency Period width scatterplot with the algorithm results. Own elaboration.

Figura 5: Graphical results. Own elaboration.

Conclusions

The conclusion after developing the thesis is that indoor location of a smartphone using VLC is possible.

Positioning a smartphone using known references doesn't seem viable given the results obtained. The quality of the smartphone and not knowing some physical parameters of the device doesn't allow a good location algorithm.

The setup of the lightning system with a programmable frequency is valid because it has allowed us to see the effect that lightning with a frequency has in photographs and to see that a relationship exists between the frequency and the period's width.

The results when identifying the LED's frequency using the smartphone's camera are very precise and robust. For that reason, it seems interesting to continue studying down that line to be able to locate a device in real-time.

For future studies it would be useful to study the use of lightning using Zigbee technology, because the lightning system used in the project would not work in a real

location due to the network's alternating current. The effect of different conditions should be studied, such as the distance between the smartphone and the LED or the tilt of the device.

Bibliography

[Kuo et al., 2014] Kuo, Y.-S., Pannuto, P., Hsiao, K.-J., and Dutta, P. (2014). Luxapose: Indoor positioning with mobile phones and visible light. Technical report, University of Michigan.

[Luo et al., 2014] Luo, P., Ghassemlooy, Z., Minh, H. L., Khalighi, A., Zhang, X., and Yu, C. (2014). Experimental demonstration of an indoor visible light communication positioning system using dual-tone multi-frequency technique. Technical report, Northumbria University, Ecole Central Marseille, Beijing University of Posts & Telecommunication, National University of Singapore.

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

Las nuevas tecnologías han supuesto una revolución en todos los sectores de la economía real, no sólo en la forma en que nos relacionamos entre nosotros sino también el modo de viajar y movernos por las distintas ciudades del mundo, todo ello con la intención de facilitar la vida a la sociedad y poder ahorrar tiempo en el día a día. Una de las tecnologías que más ha podido cambiar nuestras vidas ha sido el GPS o *Global Positioning System*, que sirve para posicionar un dispositivo móvil mediante el uso de los satélites. El GPS ha revolucionado de forma significativa la forma en la que nos movemos teniendo como finalidad guiar a sus usuarios de un punto concreto a otro de destino, ya sea en coche, andando o transporte público y monitoriza la cantidad de coches y personas que se encuentran en la carretera. El problema surge cuando se quiere utilizar el GPS en un espacio cerrado como puede ser una tienda o un supermercado. Para que el GPS funcione tiene que haber visión entre cuatro satélites y el dispositivo, sin embargo esto no ocurre en lugares cerrados debido a las paredes y techos. En este sentido, posicionar dispositivos móviles en zonas interiores ha demostrado ser un problema con difícil solución, pues gran parte de los problemas surgen porque los sistemas no parecen ser fiables en todos los entornos; Así, el mobiliario, la temperatura o los usuarios dentro de un local generan grandes interferencias que provocan la reducción de fiabilidad. En cambio, ciertas mejoras en los sistemas de posicionamiento en espacios cerrados producirían además de buenas oportunidades para negocios, otros usos a nivel particular.

Se pueden encontrar numerosas diferencias entre el posicionamiento en espacio interior y exterior que, a su vez, hacen más complicada la localización:

- Varias señales debido a la refracción y reflexión de muebles y paredes.
- Muchas veces puede no haber línea de visión entre el objeto a localizar y los localizadores.
- Las señales se debilitan en mayor medida ya que tienen que atravesar muchos obstáculos.
- cambios en el ambiente (entra gente nueva, cambian las estanterías, se abren y cierran puertas).
- Se requiere más precisión que en el exterior, ya que las superficies son m?as pequeñas.

Sin embargo, hay ciertos factores que hacen que la localizaci?on en espacios interiores sea más sencillo:

- La climatología no afecta a la localización.
- Los movimientos son más lentos que en sitios abiertos y por lo tanto la localización es más fácil.

- Las áreas que deben ser cubiertas presentan un menor tamaño.

1.2. Aplicaciones de la tecnología

Poder localizar un teléfono movil en espacios interiores tiene muchos usos. Es cierto que algunos de ellos no tienen sentido porque aun se encuentran en fase de desarrollo. Sin embargo ciertos usos pueden ayudar a la sociedad a hacer su vida más fácil, como avisar a los usuarios de que hay gente en los hogares o ser capaz de guiarles en un museo sin necesidad de contar con una persona física. A continuación se añade una lista de usos que se podría dar a esta tecnología:

- **hogar:** Se podría por ejemplo localizar todos los objetos en una casa, detectar en que habitación esta un usuario y actuar en función de ello (apagar otras luces que no se estén usando, encender la música de forma inteligente en cada habitación...etc.). También ayudaría en la seguridad, por ejemplo avisando si queda alguien dentro de un hogar durante un incendio.

- **Ofertas según posicionamiento:** Ésto habilitaría a los negocios para ofrecer información y publicidad en función de la localización del individuo. Por ejemplo, serviría para informar a los usuarios que están en un cine de aquellas ofertas y promociones relaciondas, tambien serviría para guiar dentro de un supermercado hacia las ofertas o descuentos que en ese momento se esten promocionando, o para guiar a un usuario hacia una tienda específica dentro de un centro comercial. Además, también sería muy útil para indicarle a una persona donde esta su embarque en una estación de tren o aeropuerto.

Figura 1: Ejemplo de como funcionaría el sistema de ofertas.Fuente: [ByteLight, 2015].

- **Servicios de emergencia:** Poder saber donde están los efectivos de policía, bomberos o otros cuerpos de seguridad es vital para mejorar su funcionamiento. A través de esta tecnología se podría saber donde ha caído un herido, o si todos han abandonado una zona de riesgo.

- **Guiar a gente con discapacidad visual:** Guiar a ciegos en espacios interiores para que puedan moverse con normalidad dentro del mismo.

- **Industría y Servicios:** Monitorizar donde se encuentra toda la mercancía en un almacén, guiar robots para que se muevan entre distintas estaciones, mejorar la seguridad del personal sabiendo donde se encuentran y así evitar colisiones.

- **Optimización y logística:** Si se tiene el control sobre todas las personas y objetos dentro de un espacio cerrado, se puede monitorizar y más adelante optimizar la logística de todas las operaciones,por ejemplo sería útil para minimizar un recorrido entre distintas estaciones, hacer más eficiente el transporte de mercancías o el cargamento en un barco.

- **Subsuelo:** Trabajar donde no hay luz o donde ésta es escasa puede ser una de las grandes aportaciones de la localización en espacios cerrados. Se podría utilizar en minería, creación de túneles o trabajos en alcantarillado.

- **Captura de movimiento y realidad virtual:** Si se puede localizar con gran exactitud la posición de una parte del cuerpo humano, se capturaría el movimiento de las extremidades y se realizaría simulaciones. Además la localización en interiores mejoraría la realidad virtual ya que podríamos interactuar con paredes y objetos.

2. Tecnologías para la localización

2.1. Introducción

A día de hoy existen diferentes alternativas que se están desarrollando e implementando para localizar dispositivos móviles en interiores. Ninguna de las opciones está reconocida como la mejor solución teniendo cada una de ellas sus ventajas e inconvenientes [Mautz, 2012]. A continuación se va a explicar como funciona el sistema GPS y que tecnologías se están utilizando y desarrollando para posicionar en espacios interiores.

2.2. GPS

El sistema GPS o *Global Positioning System* utiliza satélites en órbita para posicionar los teléfonos móviles. Para que funcione correctamente se necesita que el dispositivo tenga visión directa con cuatro o más satélites (el cuarto sirve para descartar una de las dos posiciones que se encuentran utilizando trilateración). Hoy en día, el sistema no sólo se emplea para la conducción sino tambien para fines militares. Además, con la aparición de la aplicaicíon *Google Maps* ha conseguido sustituir al mapa en las ciudades.

El sistema GPS tiene dos problemas fundamentales para posicicionar un teléfono móvil en espacio cerrado. Por un lado, este sistema tiene una precisión de varios metros, adecuada para lugares y distancias grandes pero no para centros comerciales, donde 2 metros pueden marcar la diferencia entre un pasillo y otro. El segundo problema que tiene es que necesita que los satélites tengan visión directa con el dispositivo, y por lo tanto no funcionan en lugares cerrados con paredes y techo.

2.3. Bluetooth

Esta tecnología inalámbrica permite intercambiar información a corta distancia utilizando ondas de corto alcance entre dispositivos fijos y móviles. Uno de los problemas que ha padecido hasta ahora es el alto consumo de la batería del dispositivo móvil, sin embargo con la incorporación del Bluetooth 4.0 (bajo consumo) se ha eliminado este inconveniente. El hecho de intentar utilizar el sistema Bluetooth para localizar dispositivos móviles no es muy fiable, ya que las señales Bluetooth (RSSI) no son muy estables y son muy susceptibles a interferencias con otros objetos.

2.4. Beacons

Un beacon es un dispositivo que envía información vía bluetooth cuando un smartphone u otro dispositivo compatible se encuentra cerca. Los beacons utilizan el denominado protocolo ibeacon desarrollado y lanzado en 2013 por Apple Inc. También se puede emplear para localizar un teléfono móvil con una precisión de 1 a 4 metros. La transmisión de información es unidireccional, esto significa que el beacon envía información al dispositivo, pero no al contrario. Aunque no sirviera para posicio-

Figura 2: Dibujo de un iBeacon. Fuente: [DuckMa, 2015].

nar, los Beacons sí que podrían saber cuando un dispositivo se ha acercado a una estantería, y enviarle publicidad relevante de ese *stand*.

2.5. Cámaras

Las cámaras se están convirtiendo en un método de posicionamiento utilizado sobre todo con el aumento en la velocidad transmisión de un dispositivo móvil. Así,La capacidad de cálculo de los ordenadores y la mejora de los sensores de las cámaras permiten una mayor definición en la imagen y precisión en el posicionamiento.

Actualmente, nos podemos encontrar con diferentes métodos de posicionamiento. El primero, utilizando un modelo 3D como referencia,se basa en la detección de objetos en las imágenes obtenidas de la cámara. Una vez se identifica el objeto se obtiene información de la posición mediante una base de datos. Este proceso consiste principalmente en dos pasos: el primero trata de obtener información de la sala en la que se encuentra utilizarizando para ello objetos fijos (comparando las propiedades geométricas con la base de datos) y en el segundo se llevaría a cabo el proceso trilateración para obtener una mayor precisión

El segundo método consiste en comparar la imagen presente en el dispositivo móvil con una secuencia de fotos tomada con anterioridad [Ravi et al., 2006]. La mayor dificultad que presenta este método es el poder posicionar un dispositivo móvil en tiempo real, pues depende en gran medida de la capacidad de calculo debido a que no se utilizan balizas o objetivos que faciliten la detección por parte de los ordenadores. En este sentido, para identificar la posición del dispositivo móvil se utiliza un algoritmo que calcula la correlación entre la imagen del dispositivo y todas las de la base de datos.

También se puede obtener dicha posición respecto a unas balizas o referencias pues los métodos que sólo utilizan objetos están colocados de forma natural en los espacios presentando ciertos problemas de robustez (sobre todo cuando las condiciones pueden ir cambiando a lo largo del día). Para hacer el cálculo más simple se co-

locan referencias que automatizan la detección de una referencia y automatizan la identificación.

2.6. Campo Mágnetico

Esta tecnología cuya base se justifica en el campo magnético, nunca es igual. Para ello, se puede crear en un edificio un mapa del campo magnético donde se utilizaría el sensor de los smartphones para así poder localizar el dispositivo móvil. Existe una compañía de reconocimiento cuya actividad principal consiste en explotar dicha tecnología denominada Indoor Atlas[IndooAtlas.com, 2016]. Esta sociedad, de origen finlandés, emplea la distorsión que los edificios y estructuras crean en el sensor magnético del dispositivo móvil para crear ese mapa.

Las ventajas que ofrece este sistema es que puede localizar el dispositivo móvil en un rango de 1-2 metros, además de no necesitar ningún tipo de instalación, pues el campo magnético está instalado de forma natural en el ambiente. Además, al no necesitar hardware se podría escalar a un mayor tamaño cuando fuese necesario sin necesidad de pensar en la instalación o en el coste de la misma.

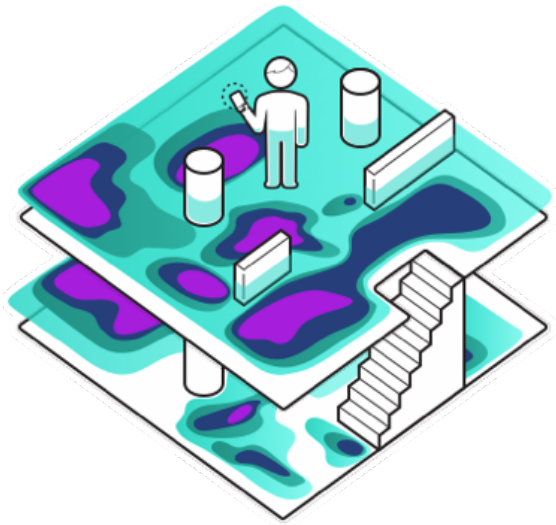


Figura 3: Mapa del Campo Magnético. Fuente: [Stenius and Yoo, 2016].

2.7. RFID

La tecnología RFID (Radio Frequency Identification) utiliza campos electromagnéticos para identificar etiquetas pegadas a objetos. Son muy utilizados en almacenes para saber qué está saliendo y entrando. En un futuro se estudiará la opción de sustituir los códigos de barras por identificadores RFID ya que permitiría, por ejemplo, pasar el carro de la compra por debajo de un arco lector de RFID para realizar el pago.

A día de hoy este sistema no se ha implementado debido al coste que el mismo conlleva pues sería muy superior respecto a los códigos de barras (en el mercado se encuentran identificadores RFID de entre 7 céntimos de euro y 100 euros, de-

pendiendo de la información que almacenen, de la batería y otras características). En cambio, si que se utilizan en almacenes para saber las existencias que quedan en stock o como sistema antirrobo de productos. Existen 2 tipos de RFID: pasivos y activos. Los pasivos son aquellos que se activan cuando entran en contacto con ondas de radio de los lectores RFID (induce una corriente que los activa) mientras que los activos son aquellos que tienen una fuente de energía que le permite emitir una onda a largas distancias. A día de hoy esta tecnología no está disponible en los teléfonos móviles. Sin embargo a través de ella se podría triangular con tres lectores RFID para localizar un objeto, o utilizar arcos que delimiten zonas y saber cuando un objeto entra en la misma.

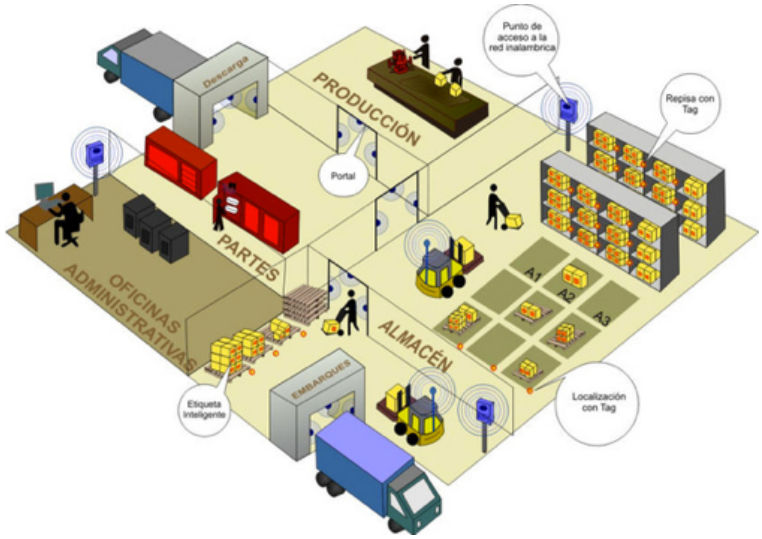


Figura 4: Funcionamiento de la tecnología RFID. Fuente: [RevistaCompetitividad, 2016].

2.8. WiFi

La tecnología local sin cables es aquella que permite a dispositivos electrónicos conectarse a la red. Los puntos de acceso WiFi emiten una señal que los dispositivos reciben, provocando que la intensidad de la señal (RSSI) baje con la distancia y así permite que se pueda utilizar varios puntos WiFi para calcular la posición del dispositivo.

La tecnología WiFi padece defectos similares al de la tecnología Bluetooth. Varía mucho según el entorno, y sufren efectos de refracción y reflexión que alteran la señal y actúan de una forma u otra según los materiales que tenga cercanos.

2.9. Google Tango

Tango es una plataforma de Google que aún sigue en proceso de desarrollo. Es una herramienta que emplea visión asistida por ordenador con el fin de permitir a smartphones o tabletas posicionarse respecto al exterior sin la necesidad de usar GPS u otras señales externas. Esto hará que los desarolladores de aplicaciones puedan ayudar a los usuarios con navegación en interiores y puedan medir distancias usando el móvil o mapas en dimensión 3D entre otras muchas cosas. El hecho de que no

Figura 5: Posicionamiento usando la señal RSSI de varios puntos de acceso. Fuente: [Tistory, 2010].

necesite ningún tipo de señal exterior hará que su implantación sea muy simple y barata. De momento Lenovo (que es la que trabaja con Google en este proyecto) pretende lanzar la primera tablet con esta tecnología en verano de 2016.

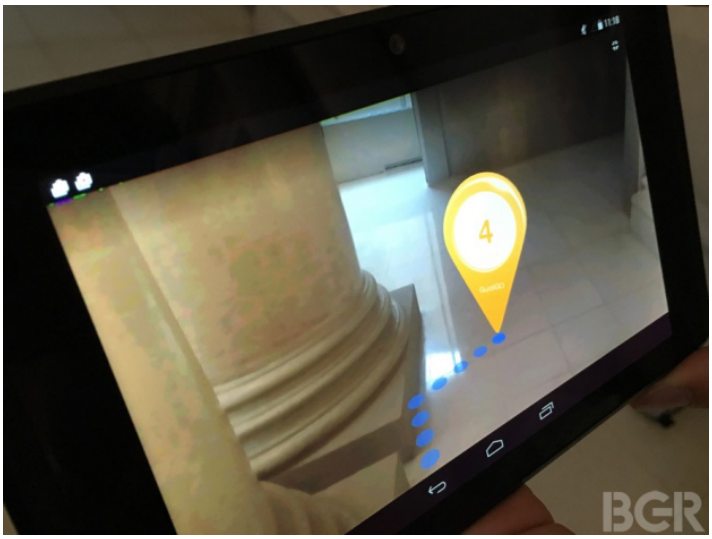


Figura 6: Navegación en interiores usando Tango. Fuente: [Smith, 2016].

2.10. Iluminación

Todos los locales interiores poseen iluminación. sta suele estar distribuida por todas partes, entre un foco de luz y el contiguo no suele haber más de 2 metros. Esto convierte a cada foco de iluminación en unas referencia muy idónea para localizar el dispositivo móvil. Además, la iluminación es una cuestión necesaria para todos los locales, suponiendo ésto una inversión inicial forzosa para todos ellos. Para poder distinguir las diferentes fuentes de luz, se debe conseguir que cada una de ella se active y desactive con una cierta frecuencia pues si el dispositivo es capaz de distinguir las distintas frecuencias se puede saber que fuentes hay cercanas. Al mismo tiempo, conviene distinguir las diferentes formas de identificar esta frecuencia y de

posicionar a partir de la identificación.

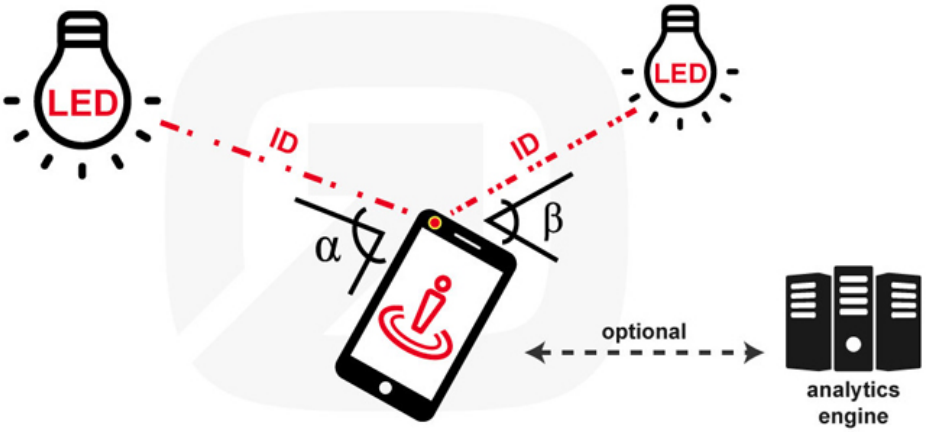


Figura 7: Posicionamiento utilizando la iluminación. Fuente: [IndoorNavigation, 2015].

3. Desarrollo del proyecto

3.1. Objetivos

En este proyecto se pretende desarrollar el posicionamiento de un dispositivo móvil en superficies cerradas mediante la iluminación, tambien conocido como VLC o *Visible Light Communication*. Como todavía no hay una tecnología que esté considerada como la más idónea para lograr este objetivo, puede resultar interesante desarrollar la tecnología VLC porque no es intrusiva y todos los locales cerrados la necesitan.

Los documentos que se han estudiado durante este proyecto utilizan sensores externos al teléfono móvil, con más precisión y frecuencia de muestreo que el sensor de luz de los dispositivos, o los dispositivos con las mejores cámaras fotográficas del mercado. Este proyecto se va a centrar en el desarrollo de una solución al posicionamiento en interiores utilizando unicamente los sensores y herramientas disponibles en un dispositivo de gama media. Sí los resultados obtenidos son positivos, el proyecto podría servir para crear una solución de posicionamiento para todos los tipos de teléfonos móviles.

Los objetivos marcados para el proyecto son los siguientes:

1. Crear un algoritmo de posicionamiento respecto a una fotografía de referencias conocidas (Sección 4).

2. Crear un LED con un ciclo de encendido y apagado (PWM) susceptible de modificación (Sección 5).

3. Crear un algoritmo que identifique la frecuencia de la luz con un dispositivo móvil (Sección 6 y 7).

3.2. Herramientas y recursos utilizados

Para este proyecto se han utilizado una serie de herramientas y libreras de programación, a continuación se explica en cada sección que recursos se han utilizado:

- Sección 4:
  - Lenguaje de programación Python para la creacion algoritmos.
  - Librería open cv: Transformación de fotografías y para seleccionar en las fotografías las referencias con el ratón.
  - Librería numpy: Operaciones vectoriales con coordenadas de las referencias.
  - Libreriá itertools: Funciones de combinatoria con las coordenadas de las referencias.
  - Librería scipy: Optimización de funciones de posicionamiento cuando hay más ecuaciones que incógnitas.
  - Librería matplotlib: Representación gráfica de funciones.
- Sección 5:
  - Lenguaje de programación para arduino para poder programar la placa "Arduino Uno".
  - Librería Timer One: Utilización de timers para el apagado y encendido de la iluminación LED.
- Sección 6:
  - BQ Aquaris E4.5: teléfono móvil utlizado para obtener los datos.
  - Aplicacion AndroSensor: Obtención de datos del sensor de luz del teléfono móvil.
- Sección 7:
  - BQ Aquaris E4.5: teléfono móvil utlizado para realizar las fotografías.
  - Lenguaje de programación Python para la creación de algoritmos.
  - Librería numpy: Utilización de funciones matématicas como seno, coseno y estádistica.
  - Librería open cv: Transformación de las imágenes obtenidas de la cámara,*threshold* simple, *threshold* adaptativo, *Canny edge detection* y *Hough lines*.

Para la redacción del proyecto se ha decidido utilizar latex por la facilidad que tiene

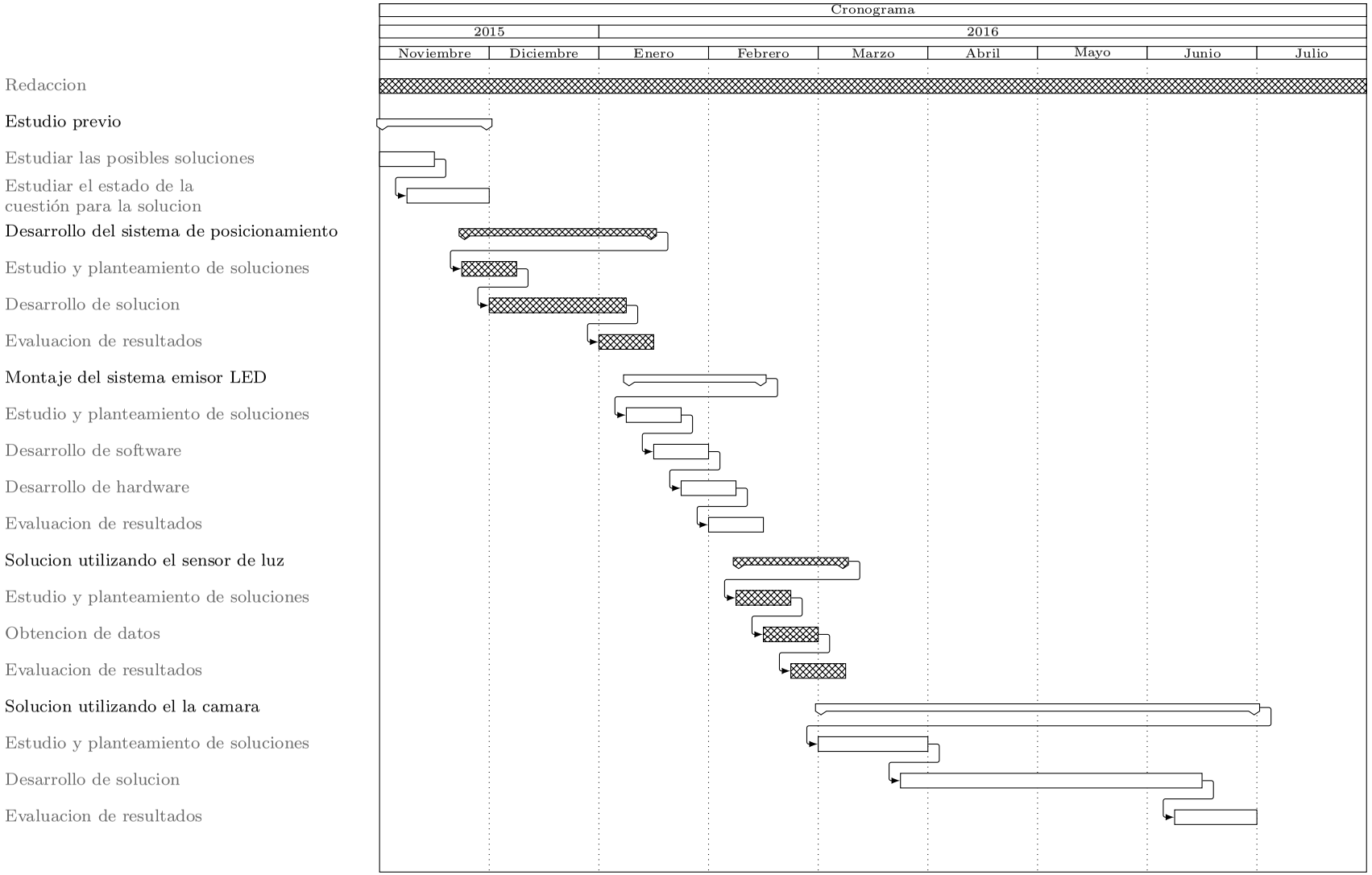
para tratar con un gran numero de tablas, gráficas, imagenes y código. Ha permitido redactar el documento de forma uniforme y sin problemas de tamaño por el número de imágenes que contiene.

3.3. Cronograma y Metodologia

La metodología que se ha llevado acabo para realizar el proyecto no ha sido el mismo en todas las secciones desarrolladas. Para los primeros apartados, donde se han estudiado tanto las diversas posibilidades que ofrece el posicionamiento en espacios cerrados como las diferentes tecnologías para ello, se ha llevado a cabo un estudio intensivo de documentos de investigación con temas relevantes, noticias y compañias punteras en cada sector.

En las secciones principales del proyecto, donde se han desarrollado herramientas experientales para la localización respecto a referencias y para identificar la frecuencia de un LED, se ha utilizado una metodología iterativa donde se estudiaba el problema, se creaba una soución y se utilizaban los resultados de la solución como *feedback* para mejorarla.

En la siguiente página se muestra un cronograma con el tiempo que se espera dedicar a cada una de las partes del proyecto.


Cronograma
2015
2016
Noviembre
Diciembre
Enero
Febrero
Marzo
Abril
Mayo
Junio
Julio
Redaccion
Estudio previo
Estudiar las posibles soluciones
Estudiar el estado de la cuestión para la solucion
Desarrollo del sistema de posicionamiento
Estudio y planteamiento de soluciones
Desarrollo de solucion
Evaluacion de resultados
Montaje del sistema emisor LED
Estudio y planteamiento de soluciones
Desarrollo de software
Desarrollo de hardware
Evaluacion de resultados
Solucion utilizando el sensor de luz
Estudio y planteamiento de soluciones
Obtencion de datos
Evaluacion de resultados
Solucion utilizando el la camara
Estudio y planteamiento de soluciones
Desarrollo de solucion
Evaluacion de resultados

4. Posición respecto a una fotografía de referencias conocidas

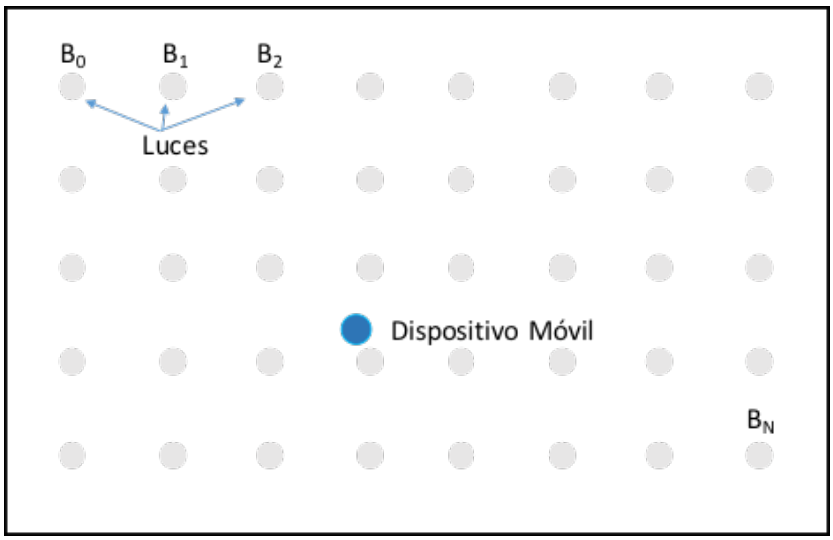


Figura 8: Esquema del mapa de luces y la posición del dispositivo. Elaboración propia.

4.1. Introducción

Una vez se ha conseguido que el dispositivo identifique las luces que ve en la pantalla tiene que ser capaz de posicionarse en el local. Esta tarea sería bastante simple si el teléfono estuviera siempre en posición horizontal con la cámara frontal paralela al techo.

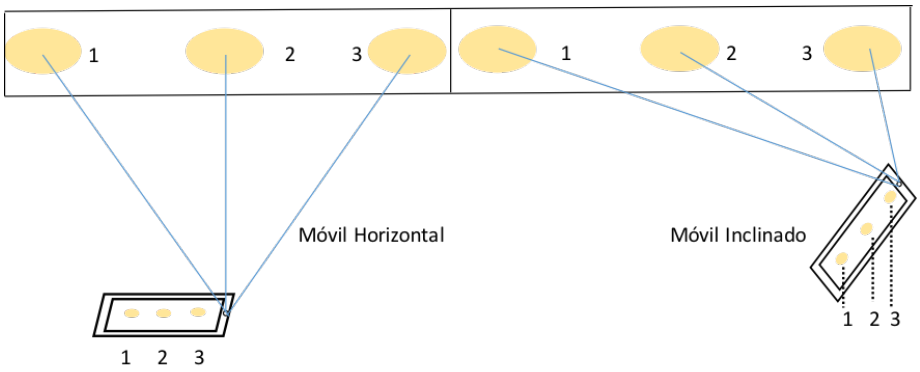


Figura 9: Proyección de las referencias en el móvil. Elaboración propia.

Si el dispositivo móvil esta horizontal, la luz que que se ve en el centro de la pantalla seria la luz que esta encima de la cámara(observar la figura 9, y por lo tanto sabríamos que el dispositivo móvil esta justo debajo de esa luz. Sin embargo si el móvil esta inclinado, la luz que se vería en el centro de la pantalla seria la n2 sin embargo el dispositivo estaría debajo de la luz n3 (como se muestra en el diagrama). Debido a que los portadores de los dispositivos no los llevan siempre en posición horizontal(lo suelen llevar inclinado para poder mirar la pantalla a la vez que se camina) la solución a este problema debe poder hacer frente a ese aspecto. Para esta

solución se asume que conocemos la posición de tres o mas luces (que llamaremos balizas) en tres dimisiones. Para este paso se tienen que hacer ciertas suposiciones:

- El teléfono móvil es capas de distinguir las diferentes balizas entre ellas (esté objetivo se supone conseguido en los apartados anteriores).
- Las fotos realizadas por la cámara están enfocadas y por lo tanto las balizas quedan en lugares (píxeles) distintos en la imagen.
- Se conocen las especificaciones del teléfono móvil, los píxeles que tiene, la distancia focal y se puede saber en que píxel esta cada baliza.

4.2. Lentes Biconvexas

Tal y como menciona [Kuo et al., 2014] las lentes biconvexas funcionan de forma que cuando un rayo de luz incide sobre ella, éste incide en la lente en paralelo al eje horizontal de esta, una vez penetra en la lente biconvexa el haz de luz se refracta y converge, atraviesa en línea recta la lente y cuando sale vuelve a refractarse y converger, para continuar recto, donde se juntan los distintos haces de luz se forma la imagen. Resulta que el haz de luz que pasa por el centro de la lente (donde se cruzan su eje horizontal y vertical) no se refracta. Esto hace que por la baliza real, el centro de la lente y la imagen de la baliza en el plano de la imagen pase una línea recta.

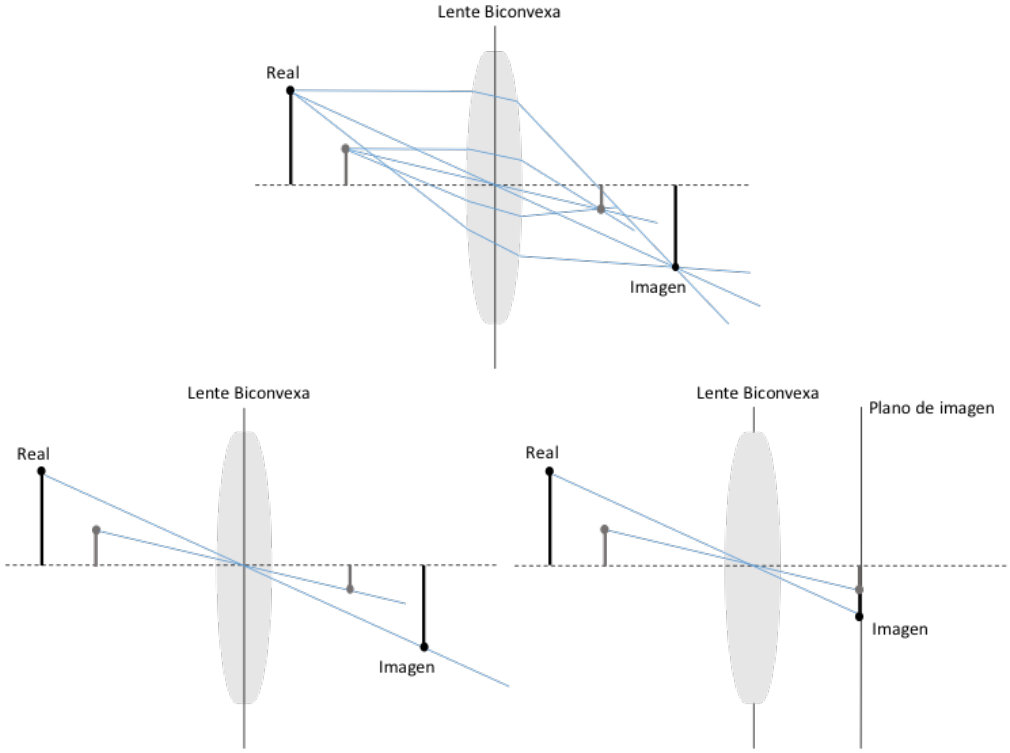


Figura 10: Funcionamiento de una lente Biconvexa. Elaboración propia.

Por ejemplo hay una recta que pasa por la Baliza 0 (B_0) con coordenadas (x_0,y_0,z_0)G en la referencia Global, por el centro de la lente biconvexa $(0,0,0)_R$ en la referencia del receptor y por la proyección de la baliza (P_0) con coordenadas (a_0,b_0,Z_f)$_R$ en la referencia del dispositivo receptor (Z_f es la distancia del centro de la lente al plano de imagen).

Por el teorema de triángulos semejantes (los dos triángulos que se forman comparten ángulos homólogos y por lo tanto sus lados tienen que ser proporcionales, con una constante de proporción K).Por lo tanto se puede establecer una relación entre las coordenadas de la proyección (a_0,b_0,Z_f)$_R$ en la referencia del receptor (centro de la lente) y las coordenadas de la baliza real en la referencia del receptor(r_0,s_0,t_0)$_R$ tal

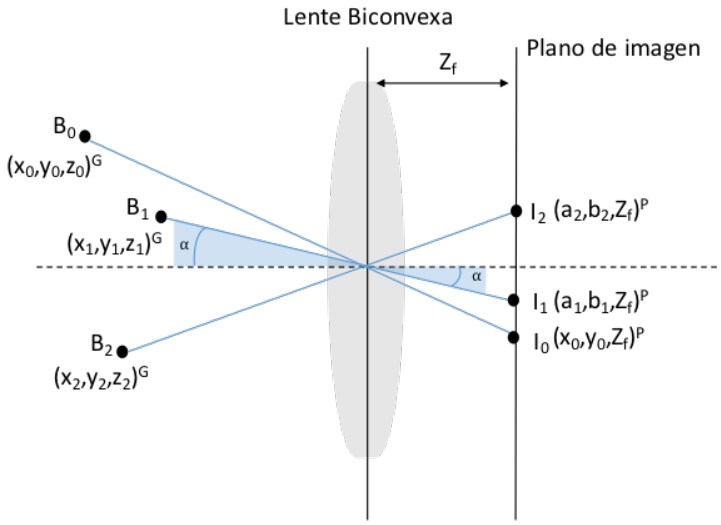


Figura 11: Semejanza entre Referencias e Imagen. Elaboración propia.

que:

$$r_0 = K_0 \cdot a_0$$
$$s_0 = K_0 \cdot b_0$$
$$t_0 = K_0 \cdot Z_f$$

Como se supone que se conocen las localizaciones de las diferentes balizas podemos saber la distancia entre ellas ($\overline{B_0B_1}$), esta distancia es igual en las distintas referencias al ser las unidades las mismas. Además se sabe la distancia que hay entre las proyecciones de dos balizas ($\overline{P_0P_1}$) por lo tanto se pueden obtener las siguientes ecuaciones:

$$\begin{aligned}
&d_{01}^2 = (r_1 - r_0)^2 + (s_1 - s_0)^2 + (t_1 - t_0)^2 \\
&= (K_1 \cdot a_1 - K_0 \cdot a_0)^2 + (K_1 \cdot b_1 - K_0 \cdot b_0)^2 + (K_1 \cdot Z_f - K_0 \cdot Z_f)^2 \\
&= (K_1 a_1)^2 + (K_0 a_0)^2 - 2K_1 a_1 K_0 a_0 + (K_1 b_1)^2 + (K_0 b_0)^2 - 2K_1 b_1 K_0 b_0 + (K_1 Z_f)^2 \\
&+ (K_0 Z_f)^2 - 2K_1 K_0 Z_f^2 \\
&= K_1(a_1^2 + b_1^2 + Z_f^2) + K_0(a_0^2 + b_0^2 + Z_f^2) - 2K_1 K_0(a_1 a_0 + b_1 b_0 Z_f^2) \\
&= (x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2
\end{aligned}$$

Dado que conocemos todas las incógnitas menos K_0 y K_1 y también se pueden obtener las ecuaciones de distancia d_{02} y d_{01}, por lo que aparece un sistema de 3 ecuaciones y 3 incógnitas (K_0,K_1,K_2) que puede ser resuelto (sistema determinado).

Debido a que las fotografías no son perfectas, las balizas no se pueden situar con facilidad en un punto exacto, para mejorar esto se puede aprovechar que el dispositivo móvil es capaz de situar en la pantalla más de tres balizas. Cuando hay mas balizas se añaden una serie de ecuaciones que nos permiten hacer una estimación mejor con el método de mínimos cuadrados. Eso se haría minimizando la suma de los cuadrados de las diferencias entre la distancia entre balizas y la distancia que se

estima con las variables K_0,K_1,...,K_n:

$$\sum_{m=1}^{N-1}\sum_{n=m+1}^{N}[K_m^2|\vec{t_m}|^2 + K_n^2|\vec{t_n}|^2 - 2K_mK_n(\vec{t_m}\cdot\vec{t_n}) - d_{mn}^2]^2$$

$N:$ *número de ecuaciones.*

$t_n:$ *vector de las balizas n en la proyección,* $\vec{t_0} = (a_0, b_0, Z_f)$.

Una vez se han conseguido estimar las constantes de proporción podemos obtener la posición P (P_x,P_y,P_z), ya que la distancia del dispositivo a cada una de las balizas tiene que ser igual en los dos sistemas de referencia (tanto en la referencia de las balizas, como en la referencia del dispositivo):

$$(P_x - x_m)^2 + (P_y - y_m)^2 + (P_z - z_m)^2 = K_m^2 \mid \vec{t_m} \mid^2$$

Si se tienen más de 3 balizas esto se convierte como en el caso de las constantes (K_0,K_1,K_2) en unos error que se quiere minimizar, ya que tenemos más ecuaciones que incógnitas:

$$\sum_{m=1}^{N}[(P_x - x_m)^2 + (P_y - y_m)^2 + (P_z - z_m)^2 - K_m^2 \mid \vec{t_0} \mid^2]^2$$

Una vez resuelto ya se tiene la localización del dispositivo.

4.3. Diseño de software de posicionamiento

Para el diseño del software se ha utilizado el lenguaje de Programación Python. Python cuenta con una librería de módulos muy amplia que servirá para diseñar el software.

4.3.1. Obtención de las referencias

Se ha considerado que el origen de la referencia global coincide con la Baliza 0, y el eje X y Y tienen los sentidos que se puede ver en la figura 12, las cuatro balizas forman un cuadrado de longitud 100 cm. Con lo descrito antes se obtienen las siguientes coordenadas para las referencias (en la referencia global):

$B0$: (X,Y,Z=0,0,0)
$B1$: (X,Y,Z=0,-100,0)
$B2$: (X,Y,Z=100,0,0)
$B3$: (X,Y,Z=100,-100,0)

La obtención de las referencias se hace a partir de una fotografía tomada por el dispositivo móvil (en la fotografía deben aparecer 3 referencias o más), el archivo de Python "SeleccionarBalizas.py.ªbre la fotografía de 3264 por 2448 píxeles y la reduce a un tamaño de 653 por 490 píxeles, un tamaño con el que se puede trabajar visualmente mejor en el ordenador.

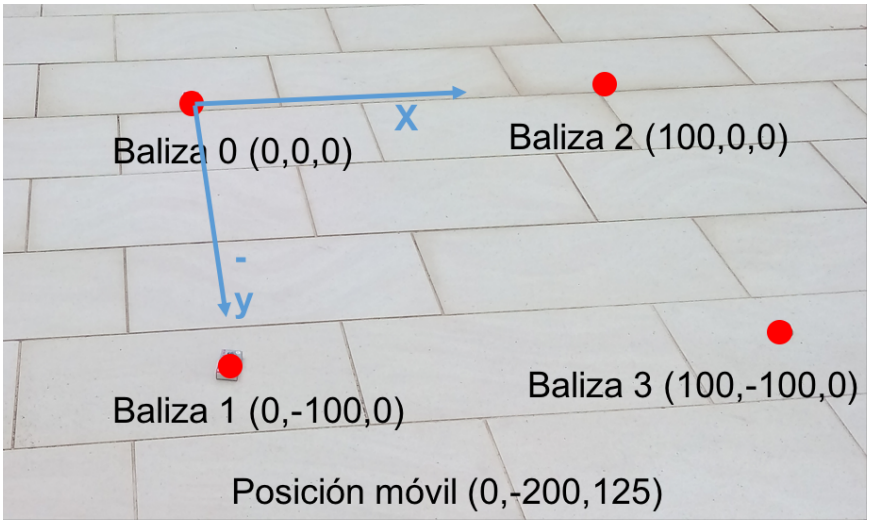


Figura 12: Fotografía tomada con la cámara del móvil donde se han marcado 4 referencias. Elaboración propia.

Una vez se ha reducido el tamaño, el programa permite seleccionar donde se encuentran las referencias, cada vez que se selecciona una referencia esta se marca con un circulo y la coordenada de ésta (X,Y,Z_f) se incorpora a una lista de puntos que contiene la posición de cada referencia en la pantalla del dispositivo. La distancia

Figura 13: Ventana del programa donde se tienen que seleccionar las referencias. Elaboración propia.

Z_f corresponde a la distancia entre el foco de la cámara y el plano de la imagen, conocido también como distancia focal, en este dispositivo es de 30 píxeles. Una vez se han introducido todas las referencias se sale del programa y lo que queda es una lista con las coordenadas de todas las referencias. En el ejemplo expuesto el la lista con las coordenadas en la referencia de la pantalla del móvil seria el siguiente:

$$
\begin{aligned}
&lista\ de\ referencias: \ [[227, 280, 30], [240, 383, 30], [387, 275, 30], [454, 367, 30]] \\
&referencia\ 0: \ (X, Y, Z = 227, 280, 30) \\
&referencia\ 1: \ (X, Y, Z = 240, 383, 30) \\
&referencia\ 2: \ (X, Y, Z = 387, 275, 30) \\
&referencia\ 3: \ (X, Y, Z = 454, 367, 30)
\end{aligned}
$$

El calculo de la posición se puede realizar tan solo con 3 de las 4 referencias, como se explica en el apartado 3.2.1 la posición del objeto consta de 3 coordenadas, y de 3 balizas salen 3 ecuaciones, por lo tanto solo con esas 3 ya se podría obtener la posición. Sin embargo, cuantas mas referencias se utilice mejor, porque se puede aprovechar el exceso de información para buscar una solución más óptima, cuantas mas referencias se tenga mas robusta es la respuesta, y pequeñas desviaciones debidas a ruido de distintos tipos afectaran menos a la respuesta

Una vez obtenida la lista de referencias, se procede a obtener la posición del móvil. Para este paso se utilizara el método explicado en el apartado 3.2.1. Para el apartado de optimización se ha utilizado el modulo de optimización óptimize'de la librería

'Scipy'. Se ha utilizado en las dos partes del algoritmo donde hace falta minimizar las constantes K, y también minimizar la distancia a cada una de las balizas.

Los resultados del posicionamiento para el ejemplo expuesto es:

$$Posicion\ Calculada:\ (X,Y,Z = -50,-222,9{,}6)$$
$$Posicion\ Real:\ (X,Y,Z = 0,-200,125)$$

Se puede ver que el error más grande se encuentra la coordenada Z del dispositivo (altura del móvil). A pesar de ellos, esta coordenada es la que menos interés tiene para situar en un plano 2D un dispositivo móvil, ya que la altura a la que se encuentra no varia mucho mientras una persona camina, y no da mucha información. La información de las variables X e Y dan mucha mas información y son en las que hay que intentar posicionar con exactitud. El error de las dos variables es de:

$$Error:\ \sqrt{(0-50)^2+(200-222)^2} = 54{,}87\ cm$$

Este error es aceptable para el posicionamiento en interiores ya que con 50cm de error se puede saber en que pasillo se encuentra el dispositivo, y aproximadamente que productos tiene delante.

4.3.2. Representación gráfica

Una vez se ha obtenido los resultados de posicionamiento, se hace un mapa gráfico del resultado, para poder entender mejor lo que esta pasando y la calidad de los resultados. En la figura 14se puede ver como están situadas las balizas (o referencias) en la fotografía tomada por el dispositivo móvil. Este tipo de herramienta sirve para comprobar pasos intermedios del proceso, como es por ejemplo ver que se hayan seleccionado todas las balizas, y que coincida su posición con la fotografía tomada.

En la figura 15 esta representado el resultado del posicionamiento, se puede ver las referencias que se hayan usado, así como la posición real de la cámara y la que ha calculado el algoritmo.

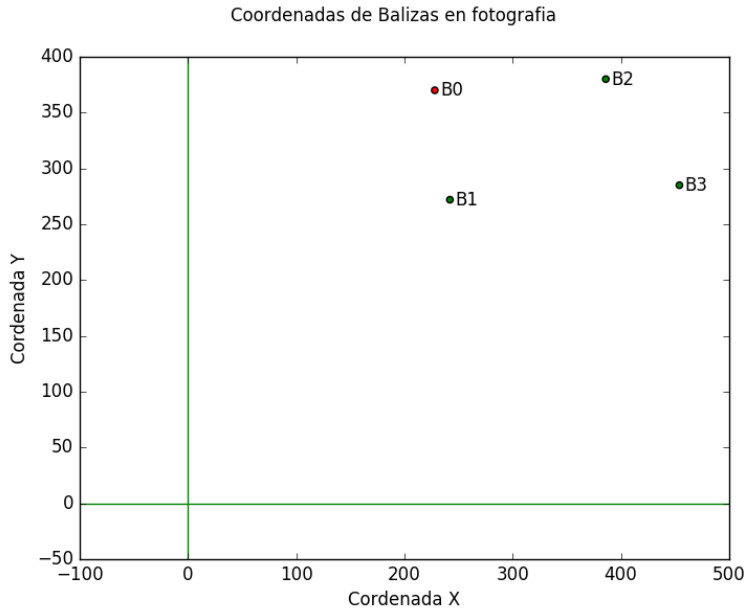


Figura 14: Coordenadas de las referencias en la pantalla del móvil. Elaboración propia.

4.4. Resultados

Los resultados del algoritmo utilizado no es robusto, las pruebas con diferentes fotografías tomadas muestran una dispersión muy alta entre los mejores y peores posicionamiento.

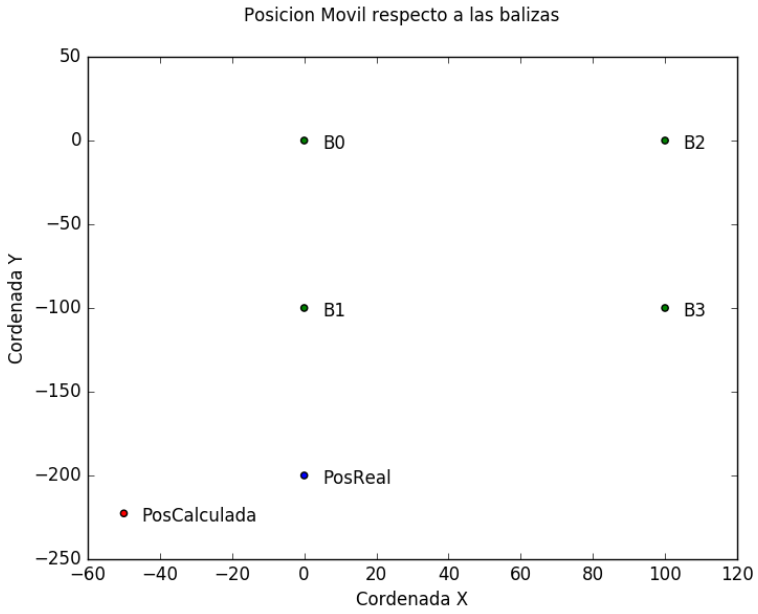


Figura 15: Resultado gráfico del posicionamiento. Elaboración propia.

5. Montaje de la iluminación

5.1. Introducción

El objetivo de esta parte del proyecto es alterar la conexión de un LED para que la frecuencia de la iluminación se pueda modificar. Las luces convencionales conectadas a la red no pueden ser modificadas para que se apaguen y enciendan de forma automática de manera simple. Para intentar simular la iluminación de una luz en un local se va a utilizar un LED de 7 W y un transistor que permite cortar el paso de corriente cuando está activado. Con Arduino se puede programar el transistor para que se encienda y apague a frecuencias muy altas de tal forma que la iluminación también funcione a esa frecuencia.

5.2. hardware

Para el montaje de cada uno de los transmisores se ha utilizado el siguiente material:

- **Bombilla:** La bombilla utilizada es una bombilla LED de 7W alimentada a 12V. Se ha decidido utilizar bombillas LED para el ensayo debido a que hoy en día estas luces se estan imponiendo en el mercado, y aun más si cabe en locales comerciales, debido al ahorro energético que generan. Además, tienen un tiempo de respuesta muy inferior al de las bombillas convencionales, debido a la naturaleza del ensayo este factor es importante, ya que si el tiempo de respuesta de las bombillas es lento, a frecuencias altas este retardo puede afectar a los resultados. La alimentación utilizada ha sido de 12 Voltios, pese a que en los lugares comerciales se alimenta a 220V se debe a que resultaba mas conveniente para el montaje y el uso en los laboratorios de la universidad.

- **Resistencias:** Se ha utilizado 2 resistencias para el circuito, la primera para alimentar la bombilla LED de 1 Ohmios y una resistencia para la salida del pin de la placa de 300 Ohmios.

- **Transistor BD137:** Se ha utilizado un transistor para poder alimentar la bombilla a los 12 Voltios necesarios, la placa de Arduino Uno tiene una salida máxima de 5V y por lo tanto no permite conectar correctamente la bombilla. El transistor permite utilizar la señal de salida de un pin de Arduino con salida de 5V para activar o desactivar la bombilla que esta alimentada a 12V. Este transistor puede soportar corrientes de hasta medio amperio. Por lo tanto el circuito se mantendrá alimentado a 0,3A como máximo por seguridad.

- **Fuente de Alimentación:** La fuente de alimentación permite alimentar el circuito a 12V y ademas poner un tope a la intensidad para que no se estropee el transistor ni otros componentes.

- **Arduino Uno:** Placa utilizada para programar la frecuencia de la bombilla. Se ha utilizado este dispositivo porque cuenta con una programación muy simple y mucho código abierto de donde coger información.

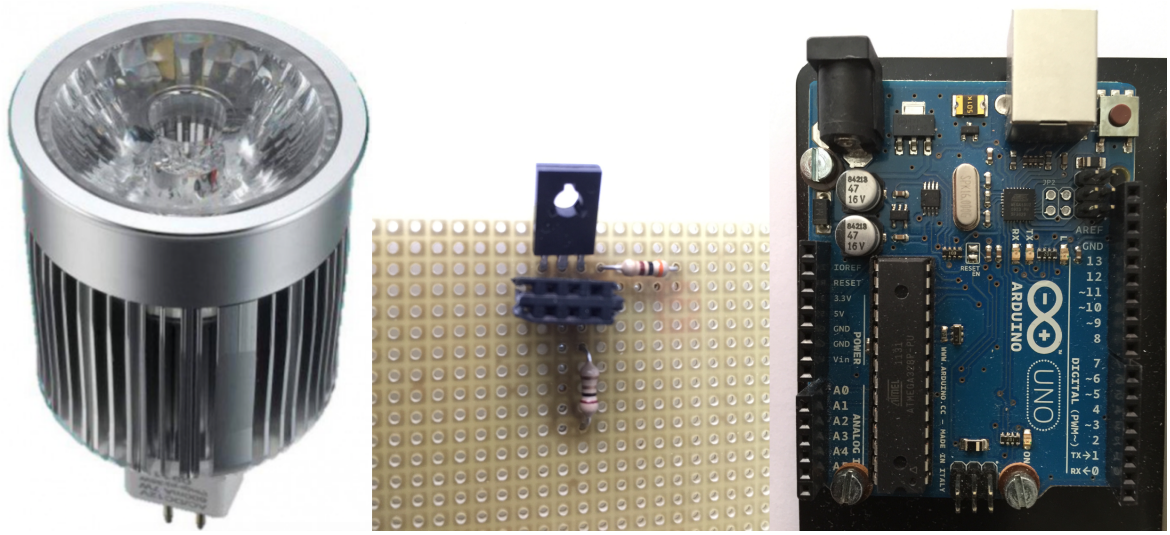


Figura 16: Arduino Uno, Circuito y Bombilla LED. Elaboración propia.

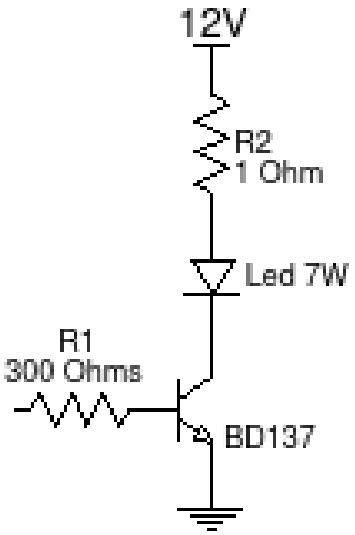


Figura 17: Esquema del circuito electrónico. Elaboración propia.

5.3. Software

Para el software se ha utilizado el lenguaje de programación de Arduino, en el se han programado salidas conectadas a cada uno de los LED's para que se pueda modificar la frecuencia con la que hace ciclos de encendido y apagado.

```
PROGRAMA FRECUENCIALED1 ARDUINO
#include <TimerOne.h>
int pinLED1=12;
float frecuencia=400;
float periodo=1/frecuencia;
float medioperiodo=periodo/2;
float mediopermicro=medioperiodo*1000000;
void setup(){ pinMode(pinLED1,OUTPUT);
pinMode(pinLED2,OUTPUT);
digitalWrite(pinLED1,LOW);
Timer1.initialize(mediopermicro);
Timer1.attachInterrupt(ISR_Tiempo);
}
void ISR_Tiempo(){
 contador++;
}
void loop() {
if (contador>0){
if (digitalRead(pinLED1)==LOW){
   digitalWrite(pinLED1,hIGh);
}
else{
digitalWrite(pinLED1,LOW);
    }
contador=0;
 }
}
```

Aquí se puede ver un ejemplo del código, donde se ha programado el primer LED (pin 12 de la placa Arduino) para que siga un ciclo de 400Hz. el programa utiliza la librería 'TimerOne.h'que se utiliza para programar un Timer que salte cuando ha pasado un medio periodo, cuando pasa este tiempo se activa el LED si estaba apagado, y se apaga si estaba encendido.

6. Identificación de la luminaria con el sensor de luz del dispositivo

6.1. Introducción

Existen diferentes maneras de identificar la frecuencia de la iluminación, una de ellas utiliza sensor de luz del teléfono móvil. La idea para justificar esta hipótesis se basa en que el sensor de luz del teléfono es capaz de saber cuando está recibiendo más o menos luz devolviendo una señal mayor o menor de luxes.Por lo tanto si la iluminación esta programada a una frecuencia el sensor debería ser capaz de captar la variación en la iluminación. Existen ensayos [Luo et al., 2014] que se han hecho con sensores ajenos al dispositivo móvil, aunque estos ensayos han dado resultados positivos no servirían para un dispositivo móvil, ya que éstos carecen de la misma precisión y tiempo de muestreo. Lo que se pretende es conseguir identificar las diferentes frecuencias de las luces con el sensor de luminosidad del móvil. En la figura 18 se puede observar los resultados que se deberían obtener en teoria (suponiendo que con la luz apagada el sensor recibe 0 lux y encendida recibe 1000 lux) a una frecuencia de 100 Hz.

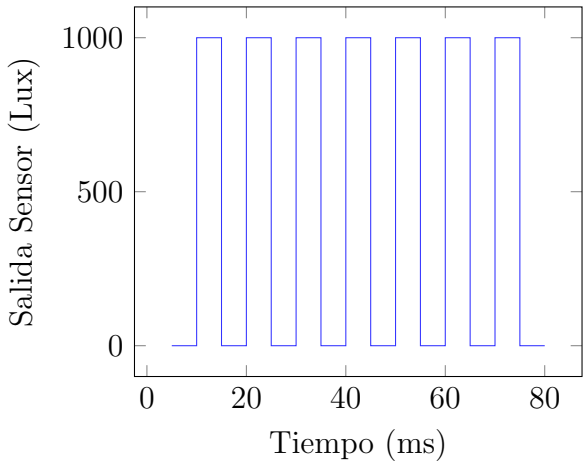


Figura 18: Respuesta teórica del sensor de luz del dispositivo. Elaboración propia.

6.2. Preparación del dispositivo

El dispositivo móvil cuenta con un sensor de iluminación, para grabar los valores de la iluminación se ha utilizado la aplicación AndroSensor (ver la Figura 19) que esta disponible en el Playstore de Android de forma gratuita. Esta aplicación permite

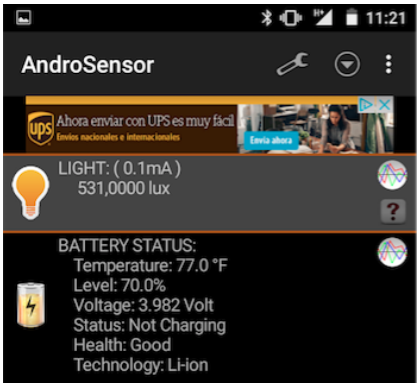


Figura 19: Pantalla de inicio de AndroSensor.Fuente: [Asim, 2015].

recopilar datos del sensor con una velocidad de 200 muestras por segundo. Por el teorema de Nyquist la frecuencia mas alta que puede identificar el móvil es de 100Hz.

6.3. Resultados

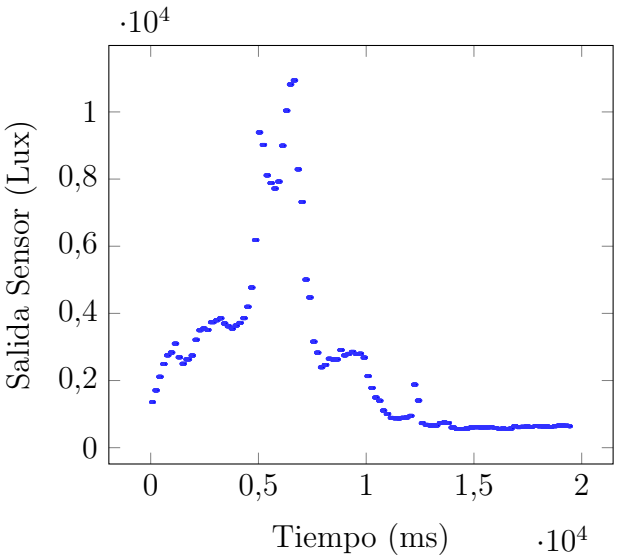


Figura 20: Diagrama de puntos donde se muestra la iluminación recogida por el sensor de luz a 100 Hz. Elaboración propia.

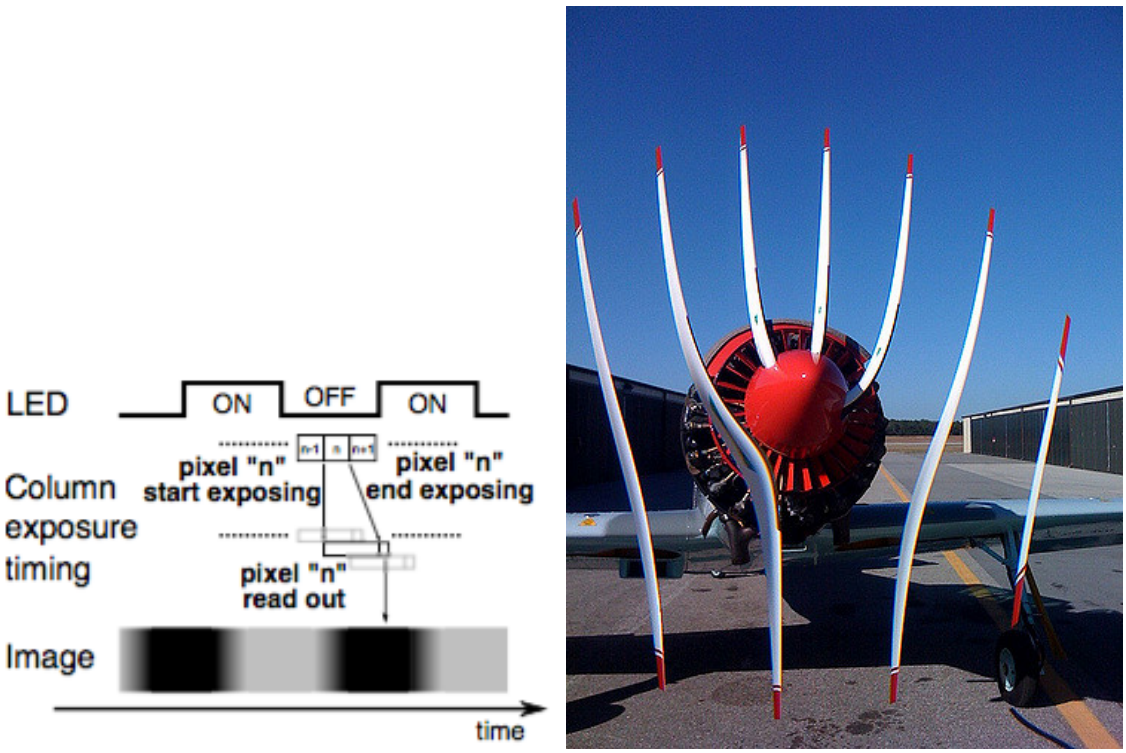
Pese a que en teoría el dispositivo puede ser capaz de recopilar correctamente informacíon de hasta 100 Hz en la práctica no ha sido así. Aunque el programa recopila datos a 200 muestras por segundo, se puede ver en la figura 20 que el sensor de luz no es capaz de recopilar correctamente esa información y recopila datos de forma errónea y que no se pueden interpretar. La conclusión es que el dispositivo móvil tiene un sensor con una velocidad de muestreo inferior a la que se necesita para la frecuencia de la iluminación.

7. Identificación de la luminaria con la cámara fotográfica del dispositivo

7.1. Introducción

Esta forma de identificar la los emisores de luz esta basada en que la iluminación modificada permite el encendido y apagado de ellas a una frecuencia alta que el ojo humano es incapaz de captar, pero si la cámara de un dispositivo móvil. Mediante una instantánea el dispositivo es capaz de detectar la presencia de las luces (o balizas) así como identificar cada una de ellas, si se van tomando fotos continuamente se puede posicionar en tiempo real. Los conceptos principales a tener en cuenta son dos:

- Se puede modificar los circuitos de la iluminación LED para que transmitan su identificación usando una frecuencia de encendido y apagado, también conocido como *pulse-width modulation* o PWM.



(a) Exposición por columnas (b) Ejemplo de efecto rolling shutter

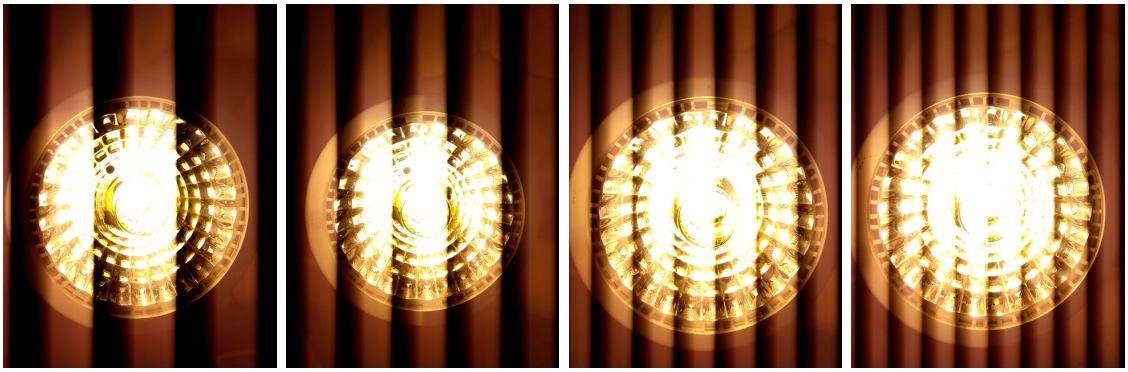
Figura 21: Efecto Rolling Shutter.Fuente: [Kuo et al., 2014] y [Cole, 2014].

- El efecto Rolling Shutter permite ver en una instantánea la frecuencia de la iluminación.Los sensores de fotografía CMOS que llevan los dispositivos móviles utilizan un método de captura conocida como Rolling Shutter effect, esto hace que la cámara no realice una sola captura en un instante, si no que realiza un barrido por filas o columnas en un tiempo muy reducido. Estos sensores contrastan con los obturadores globales, que toman instantánea global. La 21 muestra como este efecto posibilita captar la frecuencia de la iluminación

con la cámara del móvil, como las columnas de pixeles se van exponiendo progresivamente, algunas columnas se expondrán cuando la iluminación esta encendida y por lo tanto aparecerán iluminados, mientras que otras columnas se expondrán cuando el LED este apagado y por lo tanto en la imagen dichas columnas se verán oscuras.

7.2. Fotografías tomadas por la cámara

En las figuras 22 y 23 se encuentran fotografías tomadas con el dispositivo móvil donde se puede ver las bandas apagadas y encendidas que aparecen debido al efecto Rolling Shutter, Las fotografías se han tomado con la cámara normal del móvil sin realizar ningún ajuste.



(a) LED a 100 Hz (b) LED a 150 Hz (c) LED a 200 Hz (d) LED a 250 Hz

Figura 22: Luces encendidas con frecuencia. Elaboración propia.

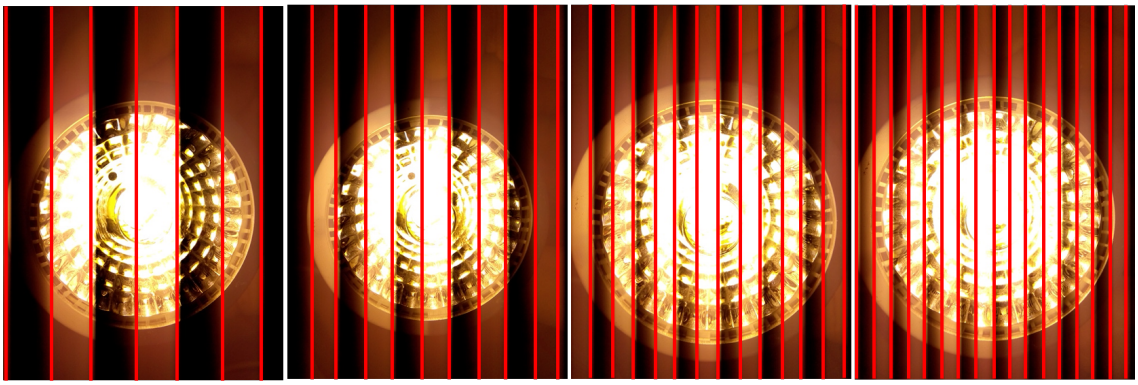


Figura 23: Identificación manual de las bandas oscuras debido al efecto Rolling Shutter. Elaboración propia.

De las fotografías se ha extraído información que sirve para poder diseñar un algoritmo que diferencia las distintas frecuencias:

- Las bandas se van haciendo mas estrechas a medida que aumenta la frecuencia.
- Estas bandas se ven con mas claridad en las partes de la fotografía mas alejadas de el foco de luz, ya que éste impide que se puedan ver con claridad las franjas.
- A medida que se aumenta la frecuencia estas franjas pierden calidad (se ven con menos densidad y cuesta mas delimitarlas).
- Al alejarse del dispositivo el grosor de las franjas se mantiene, y por lo tanto para compararlas no hace falta una referencia y permite cierta variabilidad en la distancia a la que se toma la fotografía.
- La calidad de las franjas depende de la calidad del dispositivo. Cuantos mas píxeles tenga el sensor de la cámara, mayor nitidez tendrá la fotografía, con

las cámaras de hoy en día, muy superiores a la utilizada para el montaje, las imágenes tendrían mayor nitidez.

- Al aumentar la intensidad de la luz aumenta la nitidez de las bandas . En locales las luces estan alimentadas a 230V y tienen una potencia e intensidad lumínica superior a las del montaje actual, por lo tanto el funcionamiento en locales debería ser más efectivo.

7.3. Obtención de datos para comprobar los algoritmos

Para conseguir datos para un futuro benchmark se han medido manualmente los anchos de las franjas (o bandas), para ello se ha utilizado un programa en los que hay que seleccionar el inicio y fin de cada periodo. El resultado es el presente en la tabla 1.

Frecuencia (Hz)	Medida Manual (px)
100	144.7
150	96.0
200	72.7
250	58.0
300	48.1
350	41.3
400	36.2
450	32.5
500	29.2
1000	14.4

Tabla 1: Resultados Obtenidos Manualmente. Elaboración propia.

Como se ha mencionado en el apartado anterior se puede ver como desciende el grosor a medida que aumenta la frecuencia, Si se dobla la frecuencia el grosor de las franjas se divide entre dos. En la figura 24 se puede ver un scatterplot del grosor en función de la frecuencia. Efectivamente se ve una regresión claramente marcada y si se aplica una regresión potencia se puede ver que se ajusta al 0,9995, por lo que se considera validad para expresar los valores del ancho en función de la frecuencia.

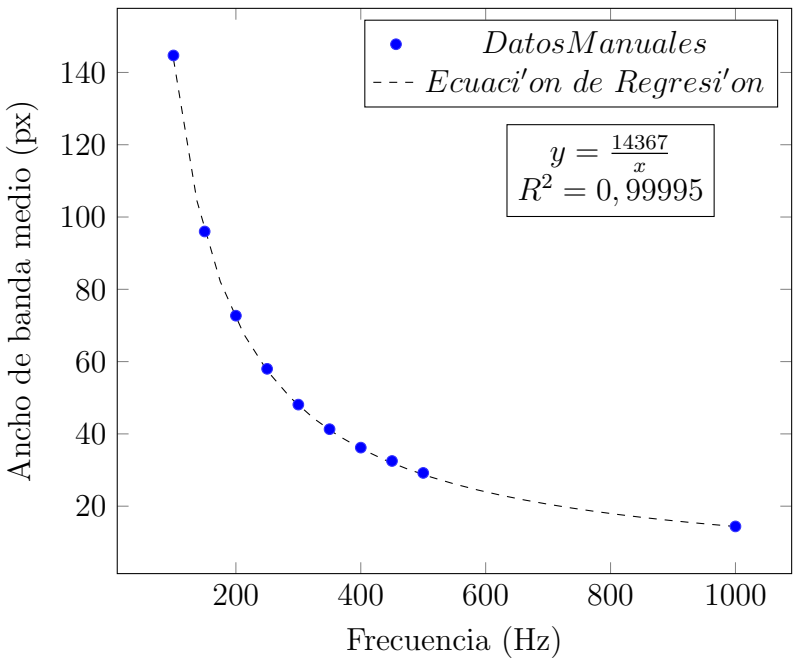


Figura 24: Scatter Frecuencia-Ancho de Bandas con linea de regresión. Elaboración propia.

7.4. Obtención de las bandas con *threshold* simple

7.4.1. Introducción

Para conseguir identificar las bandas de la fotografía original hay que llevar acabo *thresholding* (teoría del valor mínimo). Consiste en coger cada pixel de una imagen en escala de grises, comprobar si es superior o inferior a un valor mínimo (en escala de grises un pixel puede ir de negro a blanco, de 0 a 255) y otorgarle un valor u otro en función de eso. Lo explicado antes se conoce como *threshold* simple, la función que sigue este método es[Docs.opencv.org, 2016b]:

$$dst(x,y) = \begin{cases} maxVal & if\ src(x,y) > thresh \\ 0 & otherwise \end{cases}$$

La función de Python que aplica este método es la siguiente:

cv2.threshold(src, thresh, maxval, type, [dst]) → retval,dst

Parametros de entrada:

- src: matriz de entrada donde estan los píxeles de una imagen en escala de grises.
- dst: matriz de salida opcional del mismo tamaño y tipo que src.
- thresh: valor *threshold* utilizado para delimitar.
- maxval: valor que se le asignara al pixel se esta por encima de thresh.
- type: tipo de *threshold* que se va a utilizar, en este caso $ThRESh\_BINARY$.

Parámetros de salida:

- retval: valor optimo de *thresholding* utilizando Otsu's Binarization. Binariz
- dst: matriz de salida del mismo tamaño y tipo que src.

7.4.2. Aplicación de Threshold Simple

En la figura 25 se puede comprobar como al aumentar el *threshold* (thresh) el número de píxeles que siguen siendo negros va creciendo. También se observa como al disminuir el *threshold* para obtener las bandas separadas, se pierde parte de su información. Esto se debe a que en las zonas de las bandas donde cambia de apagado a encendido, el color es más claro y difuminado y por lo tanto el *threshold* simple no es capaz de captarlo.

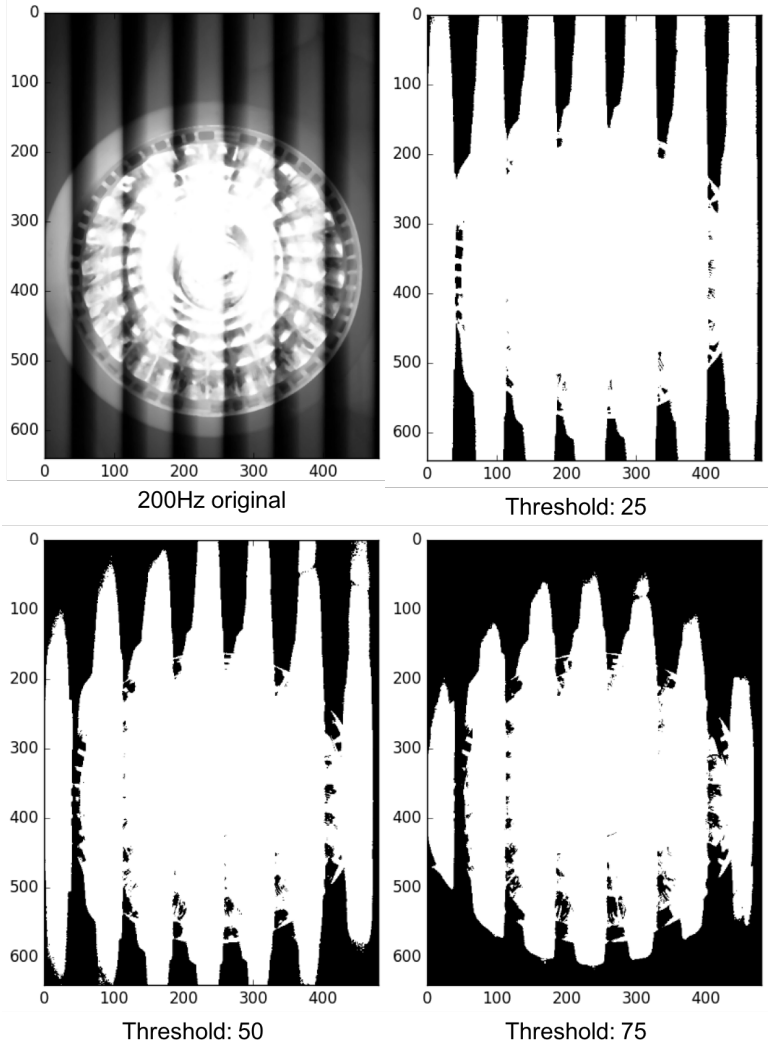
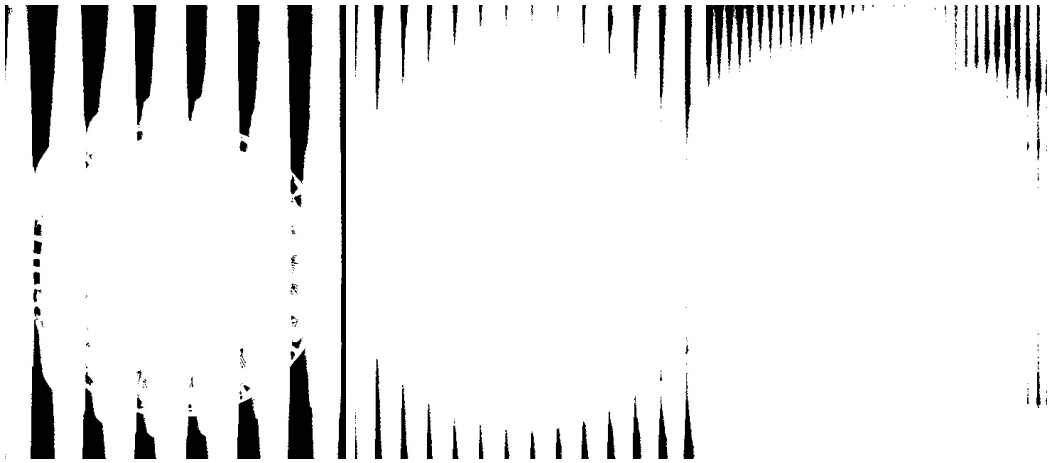


Figura 25: Iluminación a 200 Hz con diferentes valores de *threshold*. Elaboración propia.

Además cuando se cambia de una frecuencia a otra un mismo valor de *threshold* puede dejar de ser el adecuado. Cuando se aumenta la frecuencia, las zonas de transición se solapan mucho con las barras y las zonas claras, esto hace que se pierda mucha información y a medida que se aumenta la frecuencia se pierde información sobre las barras. En la figura 26 se puede comprobar como a 1000 Hz las barras inferiores de la imagen ya no se han podido detectar, además en la de 400 Hz el ancho de las barras sufre mucha variabilidad en según que zonas, lo que lleva a un calculo equivocado y poco robusto.



(a) 200 Hz (b) 400 Hz (c) 1000 Hz

Figura 26: Distintas frecuencias con filtro *threshold* simple, valor *thresh*:25. Elaboración propia.

El algoritmo inicial utilizado con *threshold* simple aplicaba un thresh de 25, ya que se ha comprobado que a frecuencias bajas es el que mejor recupera las bandas de las imágenes.

Una vez se tiene una imagen en blanco y negro con las bandas se tiene que poder obtener el numero de bandas y el grosor de ellas. El grosor es mas importante ya que el numero de bandas variara si se esta mas cerca o lejos del foco de luz, mientras que el grosor de las bandas siempre mantiene una proporción respecto al tamaño del foco de luz LED. Para ello se ha recopilado los pixeles de la primera y ultima fila, que es donde las bandas estan mas alejadas del foco. Con esto se obtienen dos listas con 0 y 255 representando los pixeles de cada fila, por ejemplo la primera fila como ejemplo podría ser:

```
la primera fila para 200 Hz es:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪  0, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0,
 ↪  0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255,
 ↪  255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 0, 0, 0, 0,
 ↪  0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪  255, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255,
 ↪  255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255, 255,
 ↪  255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0,
 ↪  0, 0, 0, 0, 255, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 255,
 ↪  255, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 255, 0, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
 ↪  255, 255, 255, 255, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
```

```
255, 255, 255, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
↪  255, 255, 0, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
↪  0, 0, 0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 0, 0, 0,
↪  0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 0, 0, 0, 0,
↪  255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 255,
↪  255, 255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255,
↪  255, 255, 255, 255, 255, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255,
↪  255, 255, 255, 255, 255, 0, 0, 0, 255, 255, 255, 255, 255, 255, 255,
↪  255, 255, 255, 255]
```

Un algoritmo pasa por las dos listas encontrando las bandas, para ello busca cuando cambia de 255 a 0 (blanco a negro) y de 0 a 255 (negro a blanco) y por lo tanto encuentra las bandas que estan completas. si una lista empieza en una banda negra no se tiene en cuenta ya que puede estar incompleta. Las dos listas se juntan con lo que se tiene el numero de bandas duplicado (esto se hace así porque al medir dos veces el sistema se vuelve mas robusto ya que tiene mas medidas).

lista de bandas: [2, 43, 38, 33, 29, 31, 35, 43, 33, 31, 31, 32, 37]

Después se desprecian las bandas que tienen una longitud muy desviada del resto (outliers), el *thresholding* no es exacto y la calidad varia según la frecuencia y la fotografía, por lo tanto es necesario filtrar bandas provocadas por un *thresholding* incorrecto (se supone que una mayoría de las bandas estan bien, ya que se usaran para despreciar el resto). para despreciar las bandas se utiliza la mediana de los datos, la media no se utiliza porque se ve muy afectada por los outliers, y por lo tanto no es una buena medida para descartar valores (esto se comprueba retirando los outliers en muestras y viendo como la media cambia mucho mas que la mediana).

Mediana: 33

También se obtiene el primer y tercer cuartil:
Q1:31

Con esto se consigue el rango intercuartil:
IQR=Q3-Q1=37-31=6

Se retiran los valores que estan por debajo de:
Q1-1,5· IQR=25

o por encima de:
Q3+1,5·IQR=43

Por lo tanto los bandas que quedan son:
lista de bandas sin outliers: [43, 38, 33, 29, 31, 35, 43, 33, 31, 31, 32, 37]

En este caso solo la banda con un grosor de 2 píxeles era un outlier.

7.4.3. Resultados

Los resultados obtenidos con el *threshold* simple no son positivos. En la tabla 2 aparece la media de los valores obtenidos para cada banda y la desviación estándar de los resultados. La desviación en los valores es muy alta, el valor máximo de la desviación se encuentra en 150 Hz, donde llega a 10.45. Esta desviación no es adecuada para el objetivo final del proyecto ya que existe mucha variabilidad al identificar cada una de las bandas, esto puede provocar identificar las bandas de forma errónea si se encuentran pocas bandas.

Frecuencia (Hz)	**Media Ancho Periodo (px)**	**Desviación Estandar (px)**
100	155.2	6.72
150	110.9	12.90
200	69.3	9.35
250	51.3	10.40
300	40.1	10.45
350	25.4	8.61
400	15.4	7.38
450	12.8	5.46
500	21.7	7.46
1000	12.7	8.00

Tabla 2: Resultados con algoritmo Threshold Simple. Elaboración propia.

En la tabla 3 se comparan los valores obtenidos en el algoritmo con los valores medidos de forma manual, estos se consideran correctos ya que se puede medir con exactitud el grosor de cada banda. Se puede comprobar que el error es muy grande alcanzando un 61 % en la iluminación de 450 Hz. Además el cálculo indica que la frecuencia de 500 Hz tiene un grosor mayor que la frecuencia de 450 Hz, visualmente y numéricamente se comprueba que no es así

Frecuencia (Hz)	**Real (px)**	**Calculado (px)**	**Error (px)**	**Error (%)**
100	144.7	155.2	10.5	7
150	96.0	110.9	14.9	15
200	72.7	69.3	3.3	5
250	58.0	51.3	6.8	12
300	48.1	40.1	8.0	17
350	41.3	25.4	15.9	39
400	36.2	15.4	20.8	57
450	32.5	12.8	19.7	61
500	29.2	21.7	7.4	25
1000	14.4	12.7	1.7	12

Tabla 3: Comparativa entre Threshold Simple y los datos manuales. Elaboración propia.

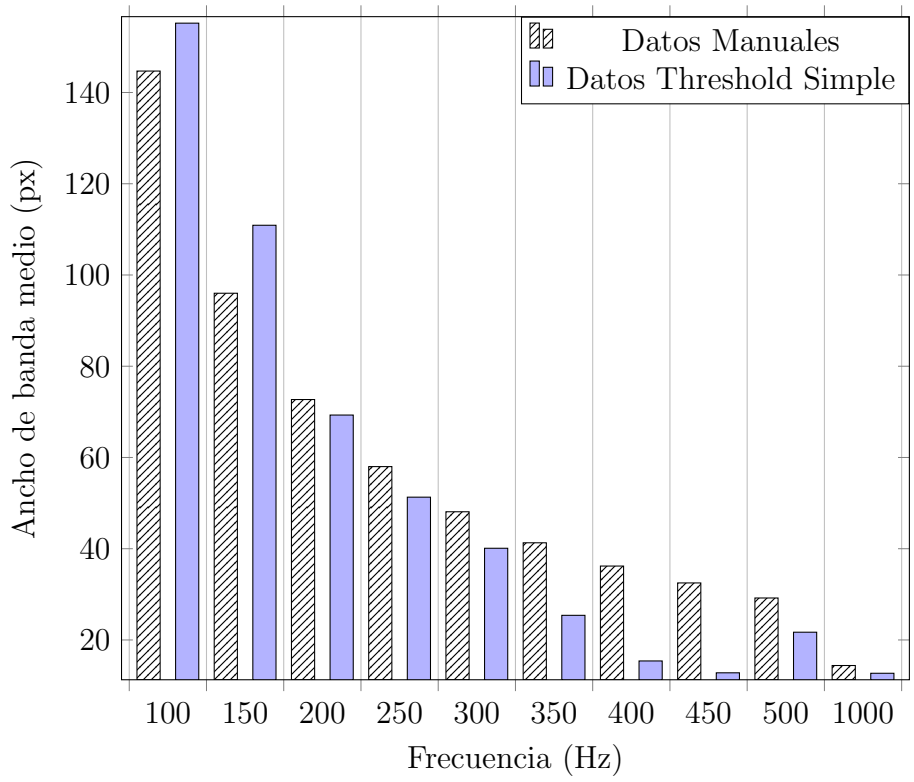


Figura 27: Diagrama de barras Frecuencia-Ancho de Banda con datos manuales y Threshold Simple. Elaboración propia.

7.5. Obtención de las bandas con *threshold* adaptativo

7.5.1. Introducción

Realizando un estudio mas exhaustivo debido a las limitaciones de la solución anterior, se ha encontrado otra forma de identificar las bandas de las imágenes. Este método consiste en utilizar un valor de *threshold* diferente para cada pixel utilizando los pixeles colindantes, de esta forma consigue mas detalle, ya que conseguirá ver diferencias en zonas más pequeñas. La función que sigue este *threshold* es [Docs.opencv.org, 2016a]:

$$dst(x,y) = \begin{cases} maxVal & if\ src(x,y) > T(x,y) \\ 0 & otherwise \end{cases}$$

donde T(x,y) es un *threshold* diferente calculado para cada pixel, este *threshold* se puede calcular de dos formas:

- **método de la media aritmética:** el valor *threshold* T(x,y) de cada pixel se obtiene de la media aritmética de los pixeles vecinos,incluido el pixel en cuestión [Fisher, 2004b]. Con esto lo que se obtiene es un valor *threshold* que depende solo de los pixeles más cercanos, y así se puede diferenciar mas los cantos y aristas en las imágenes. En este caso todos los pixeles tienen el mismo peso, si se utiliza una matriz 3X3 para realizar el *threshold* de un pixel, cada uno de los pixeles tendría un peso de $\frac{1}{9}$

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Figura 28: Matriz de 3X3 con el peso de cada pixel para la media. Elaboración propia.

- **método de Gauss:** En el método de Gauss, los pesos de los pixeles vecinos se obtienen usando una distribución de Gauss en 2D, realiza una media ponderada de los píxeles [Fisher, 2004a]. La ecuación es la siguiente:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

La forma más simple de entender esto es que los puntos mas cercanos al píxel que se está tratando tendrán un peso mayor que los mas alejados. Dado que los píxeles son discretos (no hay una función continua que diga el color de cada píxel) se tiene

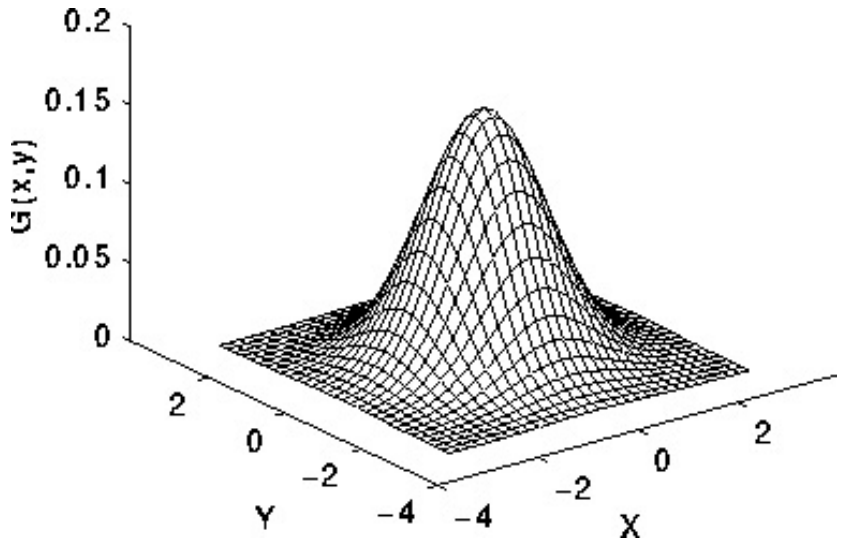


Figura 29: Distribución de Gauss en 2D con media 0,0 y σ=1. Fuente: [Fisher, 2016].

que encontrar una aproximación discreta de la función de Gauss antes de poder realizar la media ponderada. La función de Gauss es continua, nunca alcanza el cero absoluto, sin embargo, al discretizarla se considera que mas allá de 3σ se considera que la función es nula.

El siguiente paso consiste en determinar como encontrar el peso de un píxel en la distribución, el problema esta en que el valor de la distribución cambia alrededor de un mismo píxel, por lo tanto hay que integrar la distribución sobre todo el píxel.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figura 30: Ejemplo de matriz de Gauss con los pesos reescalados por 273. Elaboración propia.

La función de Python que realiza esta transformación es:

cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst]) → dst los parámetros son:

- src: matriz de entrada donde estan los píxeles de una imagen en escala de grises.
- dst: matriz de salida opcional del mismo tamaño y tipo que src.
- maxValue: valor que se le asignara al pixel se esta por encima de thresh.

- adaptiveMethod: Algoritmo a utilizar para aplicar el método adaptativo, ADAPTIVE_THRESH_MEAN_C (media) o ADAPTIVE_THRESH_GAUSSIAN_C (Gauss).
- thresholdType: tipo de *threshold* que se va a utilizar, en este caso $THRESH\_BINARY$.
- blockSize: El tamaño de los pixeles vecinos a utilizar para el cálculo (3,5,7...), tiene que ser impar.
- C: constante que se resta a la media (aritmética o ponderada en cada caso), suele ser positiva aunque puede ser cero o negativa.

Parámetro de salida:

- dst: matriz de salida del mismo tamaño y tipo que src.

7.5.2. Aplicación de Threshold Adaptativo

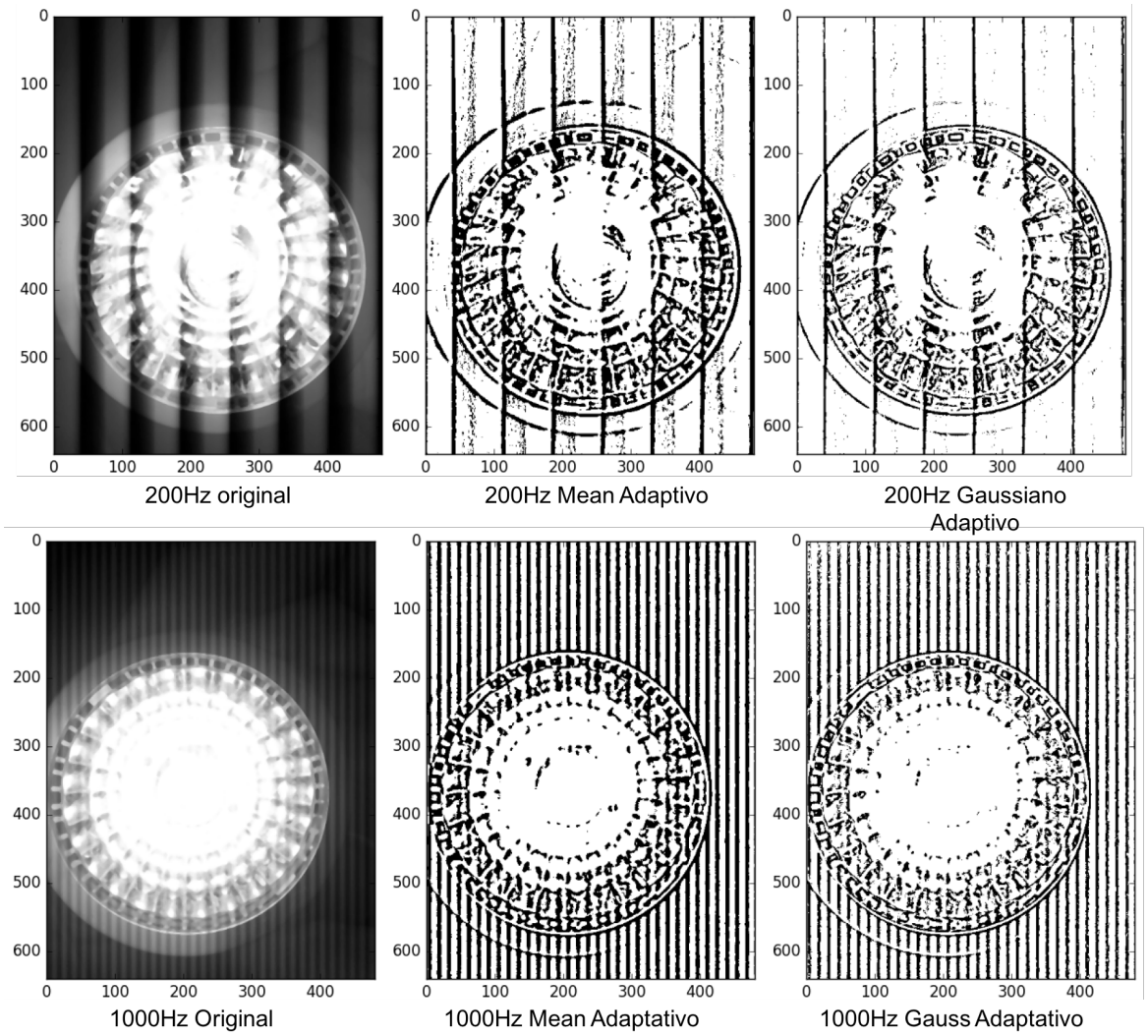


Figura 31: Threshold adaptativo con blockSize:11 y C:2. Elaboración propia.

En la figura 31 se muestran los dos tipos de *threshold* adaptativo que se pueden usar aplicado a las frecuencias de 200Hz y 1000Hz. Se puede comprobar como este tipo de transformación para la imagen es más efectiva, al ser adaptativo es eficaz en todas las frecuencias probadas (100Hz a 1000Hz) y además capta muy bien los cantos de las imágenes (la transiciones entre bandas y claros). También es interesante ver las diferencias entre los dos *threshold* adaptativo, el que utiliza la media obtiene los cantos con menos precisión que el de Gauss, ya que las lineas que aparecen son mas gruesas. La distancia entre las lineas que aparecen en ambos métodos adaptativo corresponden a un periodo entero. ya que es la distancia entre el inicio de una banda y el inicio de la siguiente.

También afecta el tamaño del bloque escogido para realizar el cálculo adaptativo. En las figuras 32 y 33 se puede ver como afecta alterar el tamaño del bloque. Al coger un bloque mas grande aumentar el grosor de las líneas y disminuye el grosor con un bloque más pequeño. El tamaño de bloque mas adecuado para los cálculos parece ser 11, ya que a frecuencias bajas aparece ruido entre las lineas si se escogen

bloques mas grandes y a frecuencias altas un bloque mas pequeño hace que no se aprecien con claridad las lineas.

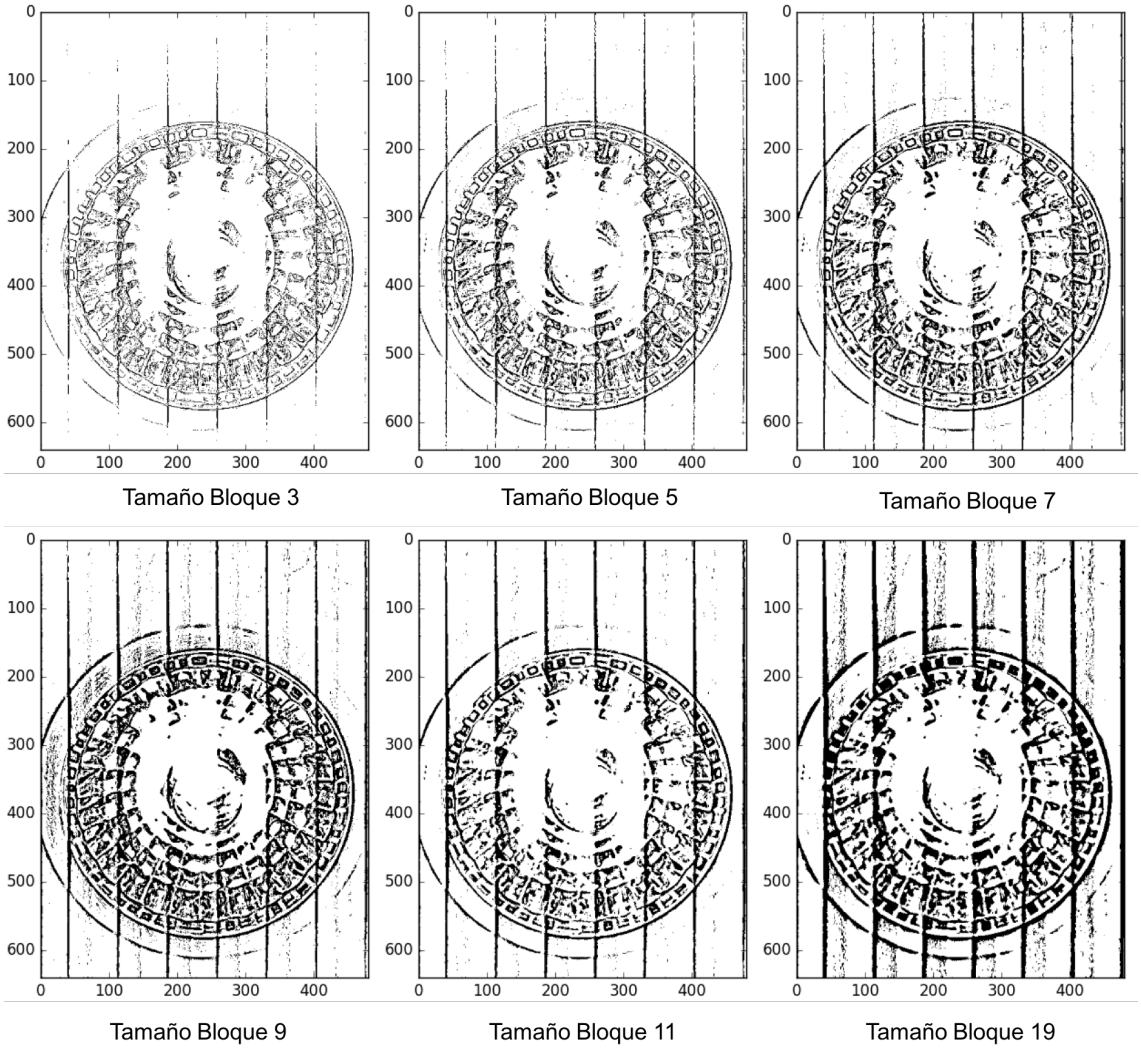


Figura 32: Diferentes blocksize a 100Hz. Elaboración propia.

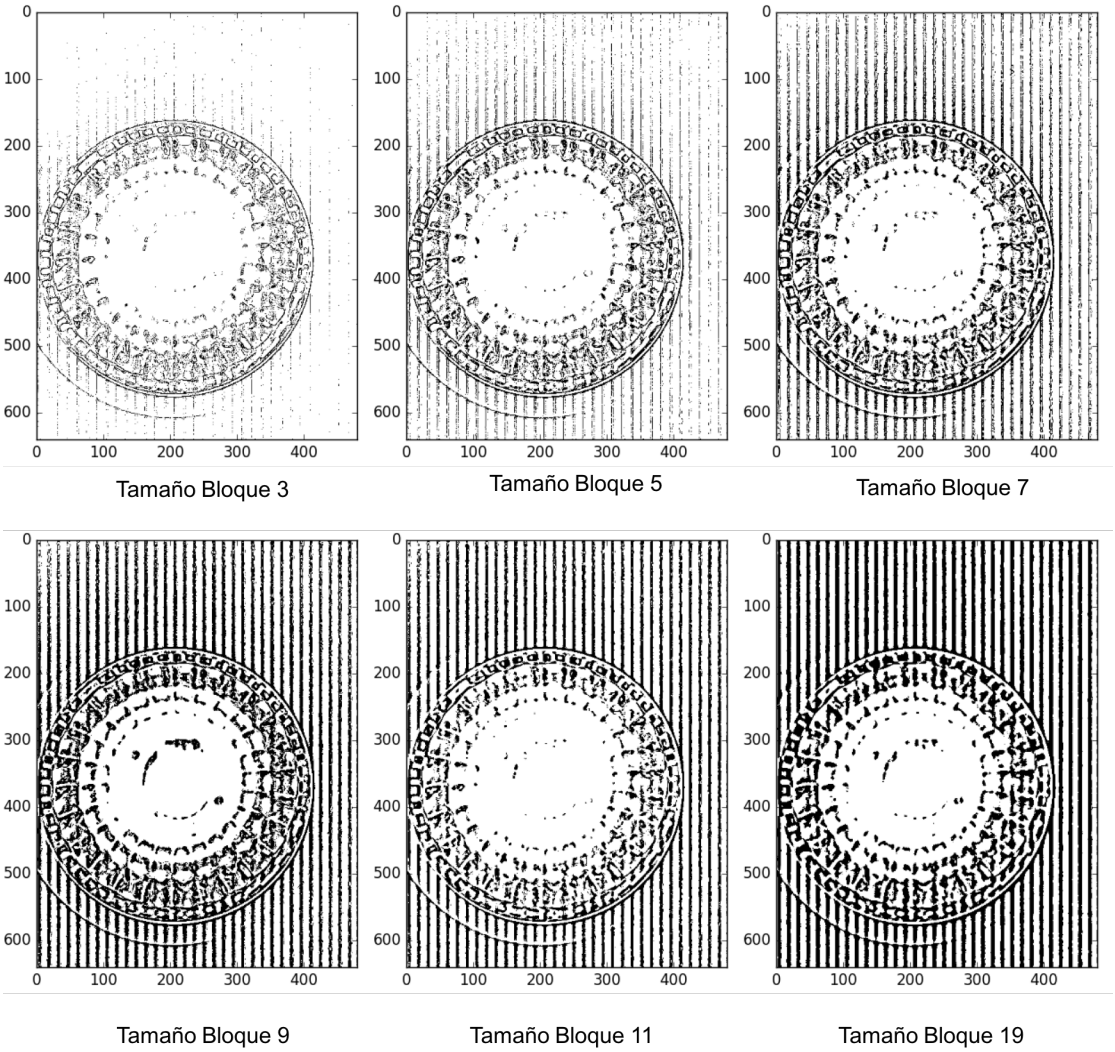


Figura 33: Diferentes blocksize a 1000Hz. Elaboración propia.

7.5.3. Obtener las coordenadas de las bandas

Una vez se ha decidido utilizar el *threshold* adaptativo se tienen las lineas que marcan el principio y final de cada periodo, sin embargo hay que conseguir identificar esas lineas de forma automática para cualquier frecuencia. El método utilizado antes no es valido porque se tiene lineas y no bandas que pueden tener el grosor de pocos pixeles o de tan solo uno, además debido a que con este método se quiere poder usar altas frecuencias como 500Hz o 1000Hz, hay que encontrar una forma mas exacta y donde el ruido no afecte. En el ejemplo de la figura 34 se puede comprobar como al leer la primera fila de píxeles el ruido llevaría a contar un número de líneas equivocado.

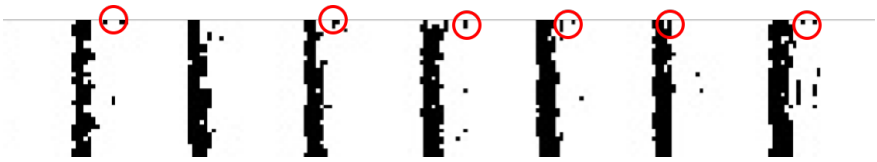


Figura 34: Recorte de 500Hz con Gauss adaptativo con ruido . Elaboración propia.

Para ello se ha utilizado el método de transformación de Hough que contiene la librería OpenCV de python. la transformación de Hough permite detectar figuras en imágenes. En este caso la figura que se quiere detectar son lineas rectas.

7.5.4. Obtención de las bandas con Canny edge detection

7.5.4.1 Introducción

En este apartado se ha utilizado Canny *edge detection* para intentar obtener una imagen donde solo aparezcan los bordes de los anchos de banda. El origen del problema surge porque los grosores de las líneas que se quieren identificar tanto en la imagen original como en la imagen después de aplicar el *threshold* adaptativo de Gauss son de bastantes pxeles. Los algoritmos de detección podrían entonces detectar varias lineas en cada borde y eso se quiere evitar porque entonces hay que identificar lineas repetidas y las descartarlas de forma automática, . Canny parece el método indicado porque en principio se obtiene una sola línea por cada borde. El algoritmo que John F. Canny desarrollo en 1986 tenia el objetivo de optimizar los siguientes aspectos:

- **Detección:** La probabilidad de detectar puntos que realmente estén en un borde se debe maximizar a la vez que se minimiza la probabilidad de detectar puntos falsos que realmente no pertenecen a un borde (Maximizar la proporción señal a ruido).
- **Localización:** El borde detectado deberia estar lo más próximo posible al borde real.
- **Número de respuestas:** Un borde real no debe acabar en más de un borde detectado (objetivo que se busca).

El algoritmo de Canny se lleva acabo en 5 etapas:

1. **Suavizar la imagen:** La imagen se difumina para eliminar el ruido.
2. **Encontrar los gradientes: Los bordes deben estar donde la imagen tiene gradientes grandes.**
3. **Eliminar los no máximos:**Una vez se tienen los valores grandes de la matriz gradiente se deben eliminar los que no son máximos locales.
4. **Realizar un doble** *thresholding***:** bordes potenciales se localizan por *thresholding*.
5. **Encontrar los bordes por histéresis:** Los bordes definitivos se encuentran suprimiendo aquellos bordes que no estan conectados a un borde muy claro.

7.5.4.2 Suavizar la imagen

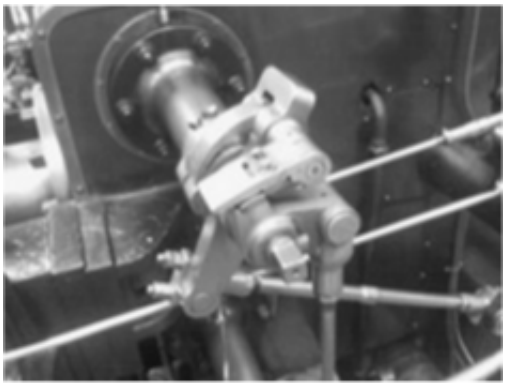


Figura 35: Imagen original. Fuente:[Wikipedia, 2016].

Para eliminar el ruido de las imágenes se utiliza un filtro de Gauss, este filtro, como se ha explicado en el apartado 7.5 suaviza las imágenes otorgándole al píxel en cuestión la media ponderada utilizando la distribución de Gauss.

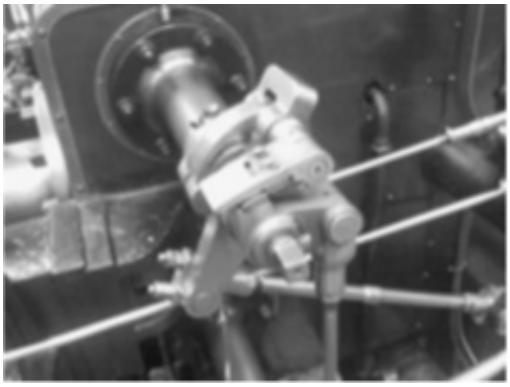


Figura 36: Imagen después de aplicarle el filtro de Gauss. Fuente:[Wikipedia, 2016].

7.5.4.3 Encontrar los gradientes

El algoritmo de Canny encuentra los bordes donde el gradiente de la intensidad en blanco y negro cambia más. Para ello hace falta encontrar los gradientes de la imagen en cada píxel. El operador Sobel utiliza 2 Kernels de dimension 3x3 que sirven para aproximar los gradientes horizontales y verticales:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * MatrizImagen \quad y \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * MatrizImagen$$

Para cada punto de la imagen los gradientes aproximados x e y se combinan utilizando pitágoras para obtener el gradiente absoluto:

$$|G| = \sqrt[2]{G_x^2 + G_y^2}$$

De la imagen se puede comprobar como una imagen de los gradientes ya muestra de por si los bordes de una manera clara, sin embargo estos bordes suelen tener espesor y no muestran con claridad donde se encuentra el borde exacto. Para encontrar los borde con mayor exactitud se debe utilizar la dirección de estos, la dirección de los bordes es:

$$\theta = arctan\left(\frac{|G_x|}{|G_y|}\right)$$

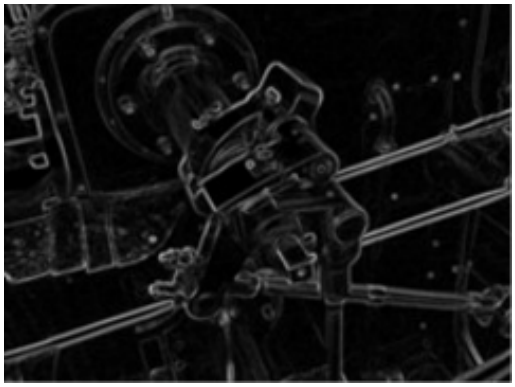


Figura 37: Imagen de gradientes después del filtro Sobel. Fuente:[Wikipedia, 2016].

7.5.4.4 Eliminar los no máximos

En este paso se pretende eliminar los espesores de los bordes y conseguir una imagen con los bordes afilados. De forma simplificada esto se consigue convervando los máximos locales y eliminando el resto de ls gradientes. El proceso a llevar acabo es:

1. Si se utilizan 8 pixeles vecinos (un kernel 3x3 donde el pixel tratado esta en el centro y tiene 8 a su alrededor) se debe aproximar la dirección θ a los 45 más próximos (0,45,90,...,360) ya que en función de ese ángulo se comparara con unos píxeles u otros.

2. Se compara la magnitud $|G|$ de cada píxel con la magnitud de los 2 pixeles situados en la direccion positiva y negativa de la dirección(observar el ejemplo de la figura 41). Si la magnitud del pixel es la mayor se conserva su valor, si no se elimina (cambia por un 0) el valor.

7.5.4.5 Thresholding doble

Los pixeles que forman bordes después del paso anterior aun tienen guardada su intensidad uno a uno. Muchos de estos píxeles son bordes de la imagen original, otros sin embargo pueden ser ruido o marcas en un a superficie sin un color constante. La forma más fácil de eliminar estos pixeles es utilizar un *threshold*, de tal forma que solo los pixeles con una intensidad mayor a un valor se preservaran, y además se les otorgue un mismo valor. El algoritmo Canny utiliza 2 *thresholds*: Si el pixel

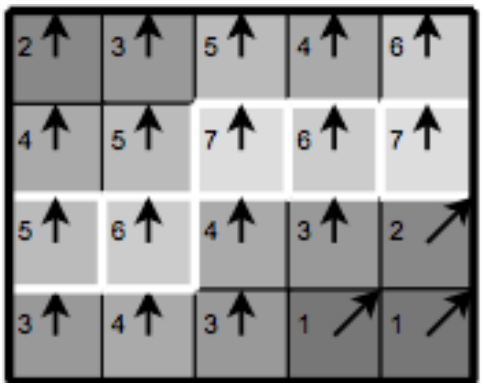


Figura 38: .

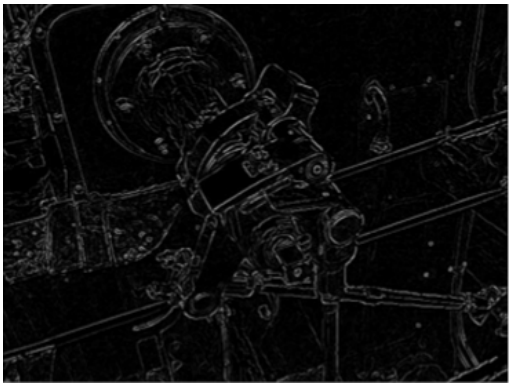


Figura 39: Imagen después de eliminar los no máximos. Fuente:[Wikipedia, 2016].

tiene un valor mayor que el *threshold* alto, se considera que el pixel y el borde son fuertes, si es menor que el *threshold* bajo se elimina y si está en medio se marca como un borde débil. Con este proceso se consigue tener una imagen con solo dos intensidades distintas, en función de cada tipo de borde.

Figura 40: Imagen después de aplicar el doble Threshold. Fuente:[Wikipedia, 2016].

7.5.4.6 Encontrar los bordes por histéresis

Los bordes fuertes se consideran como bordes certeros (no se duda de que sean bordes de la imagen original), sin embargo los bordes débiles se considera que ademas de bordes auténticos pueden ser ruido o un cambio de color en una superficie. los no

bordes probablemente estarán distribuidos de forma independiente a los bordes y los auténticos bordes tienen mas posibilidades de estar conectados. Por ello se realiza un último filtrado, donde los bordes débiles pasan a considerarse fuertes solo is estan conectados a uno de estos.

Para realizar esto se utiliza un análisis BLOB (Binary Large OBject). la imagen salida del doble Threshold (contiene bordes y potenciales bordes) es analizada pixel a pixel en kernels de 3x3, si el pixel en tratamiento contiene almenos un pixel fuerte vecino, entonces pasa a ser un pixel fuerte.

Figura 41: Imagen despues de aplicar el filtro de Histéresis. Fuente:[Wikipedia, 2016].

7.5.4.7 Aplicación del algoritmo Canny.

La función de Python que realiza esta transformación es:

cv2.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]]) → edges

Parametros de entrada:

- image: imagen a la que aplicar la transformación.
- threshold1: primer *threshold*.
- threshold2: segundo *threshold*(el algoritmo automáticamente usa el más pequeño como inferior y el mas grande como superior para asignar el tipo de borde como se ha explicado antes).
- apertureSize: tamaño del operador (matriz o kernel) Sobel que sirve para determinar los gradientes de los píxeles).
- L2gradient: determina si se usa pitágoras para realizar el modulo del gradiente (G_x, G_y) o la distancia euclídea.

Parámetros de salida:

- edges: imagen o matriz donde aparecen marcados los bordes.

7.5.4.8 Resultados del algoritmo Canny.

En la fígura 44 y 42 se puede ver el filtro aplicado a diferentes frecuencias y con diferentes parámetros. Las conclusiones obtenidas de las diferentes pruebas realizadas son las siguientes:

- es posible conseguir un filtro adecuado para frecuencias bajas como se puede ver en la figura 42 sin embargo, los parámetros no sirven por igual para otras frecuencias ,en el caso del threshold los parámetros funcionaban correctamente para toda la gama de frecuencias).

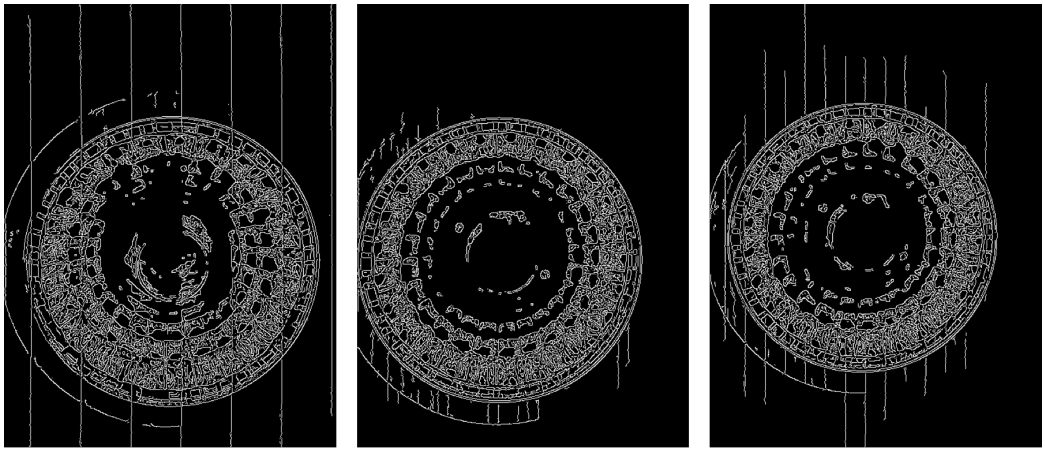


Figura 42: Filtro Canny aplicado a diferentes frecuencias con Threshold1:50, Threshold2:100. Elaboración propia.

- A frecuencias altas el filtro canny no es capaz de detectar los bordes debido a que las bandas tienen un grosor muy pequeño, la única forma de detectarlos es con un limite superior muy bajo y por lo tanto aceptar mucho ruido. en la figura 43 se observa como con el *threshold* a 40 se pueden ver los bordes con mucho ruido. Sin embargo, cuando se aumenta el *threshold* a 60, poco si se tiene en cuenta que el valor puede ir de 0 a 255, se pierde mucha información de los bordes.

- es complicado conseguir un filtro canny que se adapte a cada situación,se ha intentado utilizar el primer y tercer cuartil de la intensidad de los pixeles para obtener el *threshold* superior e inferior. El problema de esta solución es que gran parte de la imagen lo ocupa el foco de luz, y no las bandas, por lo tanto la frecuencia de las bandas afecta poco al valor de la mediana.

7.5.4.9 Resultados del algoritmo Canny con *threshold* adaptativo previo.

El algoritmo de Canny se vuelve más efectivo cuando se trabaja con imágenes que han sido tratadas con un filtro adaptativo de Gauss. En la figura 45 se ve el resultado de aplicar el algoritmo canny con unos parámetros que se ajustan para cada situación

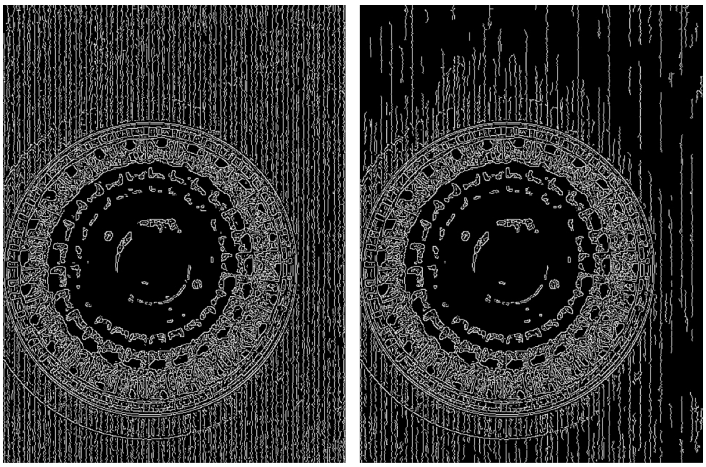


Figura 43: Filtro Canny aplicado a diferentes frecuencias con Threshold1:50, Threshold2:100. Elaboración propia.

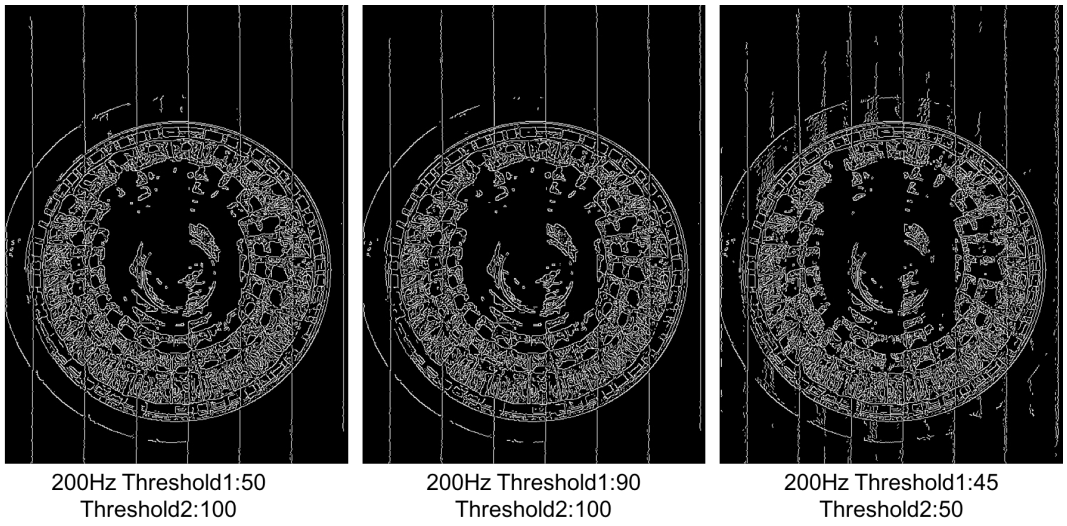


Figura 44: Filtro Canny aplicado a 200Hz con diferentes parámetros. Elaboración propia.

en función de la mediana de los colores.

```
def auto_canny(imagen, sigma=0.33):
  v = np.median(imagen)
  bajo = int(max(0, (1.0 - sigma) * v))
  alto = int(min(255, (1.0 + sigma) * v))
  bordes = cv2.Canny(imagen, bajo, alto)
  return bordes
```

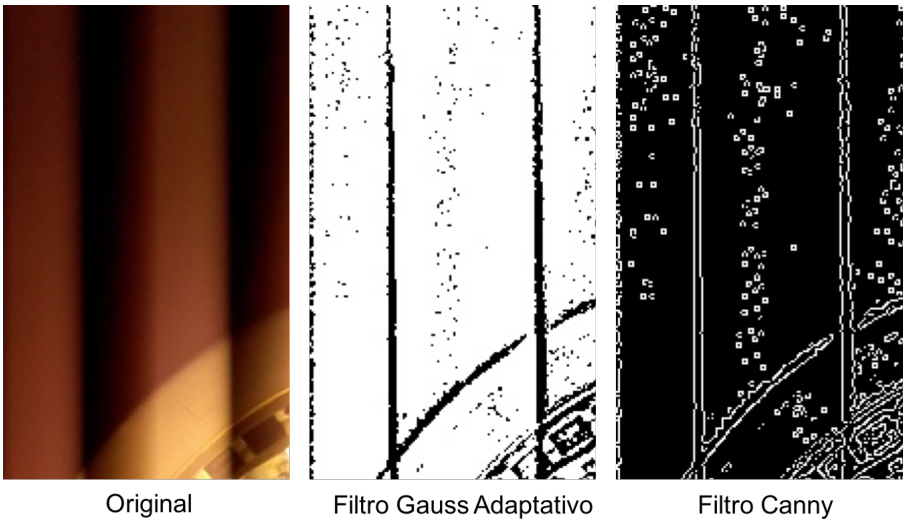


Figura 45: Proceso de Filtro Canny con parámetro automático aplicado a 200 Hz. Elaboración propia.

Como se puede comprobar los parámetros para el *threshold* bajo y alto es $(1 - \sigma) \cdot$ *mediana* y $(1 + \sigma) \cdot$ *mediana* . Esto hace que los parámetros se ajusten a cada situación mejor que si se utilizara el mismo filtro para todas las situaciones, en la figura 46 se ve el resultado en distintas frecuencias. En los resultados se puede ver como en todas las situaciones el algoritmo detecta un borde para el inicio de la linea y otro borde al final de ella.

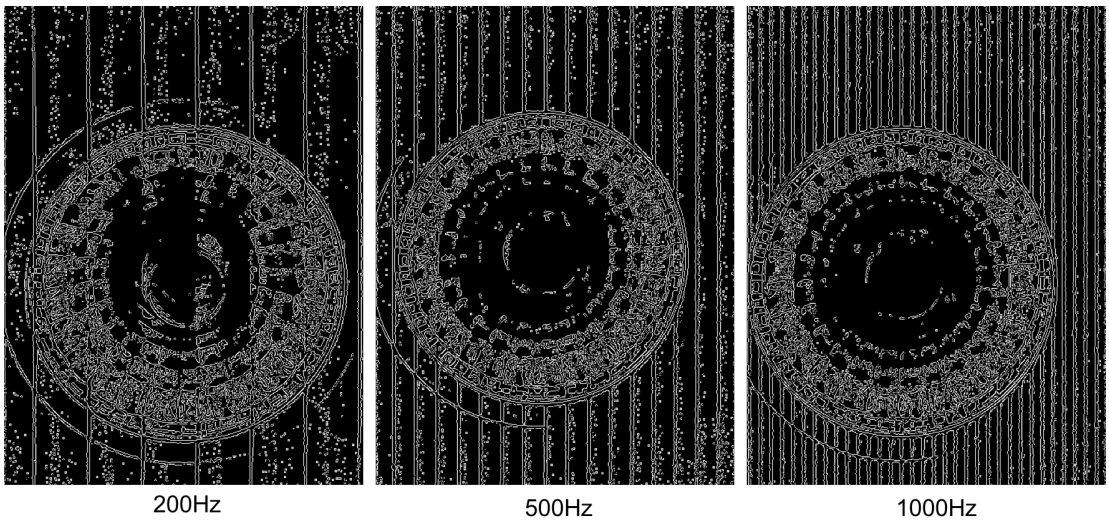


Figura 46: Filtro Canny con parámetro automático aplicado a 200,500 y 1000 Hz. Elaboración propia.

7.5.5. Transformación de Hough en rectas para obtener las bandas

7.5.5.1 Introducción

7.5.5.2 Conceptos

En general una recta cualquiera se puede escribir como $y = mx + n$ con lo que se puede parametrizar cada recta en función de los parámetros m y n (m,n). Sin embargo las rectas verticales al tener pendiente infinita no se pueden representar de esta forma. Por ello para este método se utiliza el método polar:

$$y = \left(-\frac{cos\theta}{sin\theta}\right) \cdot x + \left(\frac{\rho}{sin\theta}\right) \quad o \quad \rho = x \cdot cos\theta + y \cdot sin\theta$$

Por lo tanto cada recta tiene dos variables (ρ, θ) único cuando $\theta \in [0, \pi]$ y $\theta \in \mathbb{R}$, ρ es la distancia perpendicular al origen y θ es el ángulo de la perpendicular al origen (ejemplificado en la figura 47.

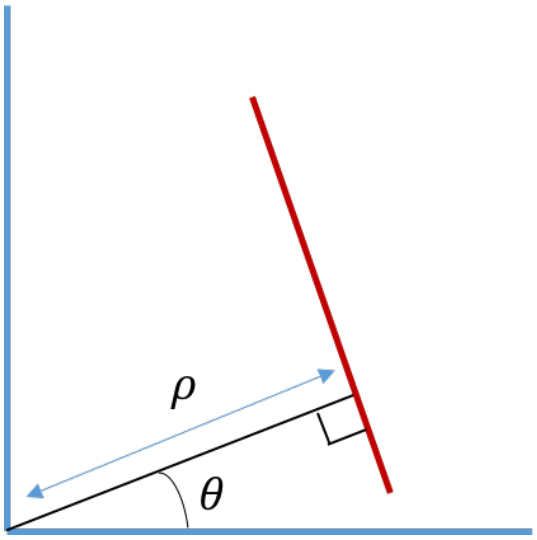


Figura 47: Notación polar de una recta . Elaboración propia.

Para un punto o píxel en una imagen con cordenadas (x_0, y_0), todas las rectas (ρ, θ) forman una sinusoidal única para ese punto donde cada valor representa cada recta única (figura 48). Por lo tanto, cuando las curvas de dos puntos se cortan, al ser esa recta (ρ, θ) única, se ha encontrado la recta que pasa por esos dos puntos. Si se generaliza el concepto anterior a un conjunto de puntos en una recta, estos producirán sinusoides en el espacio de Hough que se cortarán exactamente en los parámetros de esa recta.

Cuando se aplica este concepto para identificar líneas se utiliza un acumulador, esto es una matriz de dos dimensiones, donde se representan ρ y θ). El acumulador esta a 0 al principio, el tamaño de este depende de la precision que se necesite, suponiendo que se quiera una precision de un grado para θ entonces el acumulador tendrá 180 columnas (de 0 a π). El numero de filas máximo que puede haber con una precisión

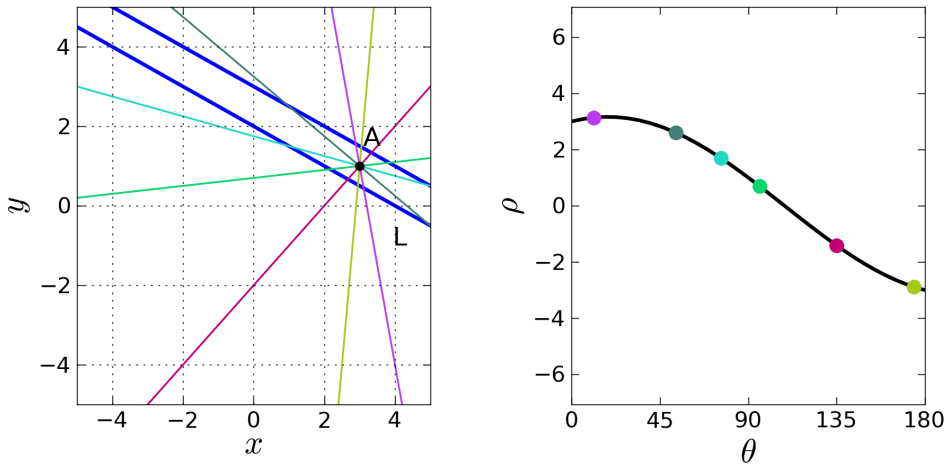


Figura 48: Sinusoide de todos los posibles ρ, θ que pasan por un punto . Fuente: [Pinard, 2011].

de un pixel para ρ es el número de píxeles en la diagonal de una imagen. Para cada punto en una imagen (x_i, y_i) se realiza un barrido con el rango de θ (en el ejemplo expuesto 0,1,2,...,180) y se encuentra el valor correspondiente ρ, en el acumulador se incrementa a cada (ρ, θ) que aparece. Si se imagina una imagen con una línea de píxeles en el centro y se cogen todas las lineas posibles para cada punto (figura 49) se puede ver visualmente como las rectas horizontales de color amarillo (que son la misma) tendrían un valor acumulado de 3, mientras que el resto de rectas al solo aparecer una vez, tendrían un acumulado de 1. Por lo tanto al coger el máximo del acumulador se identifica la linea en la imagen. En una imagen con muchas líneas los máximos en el acumulador son locales y no máximos absolutos.

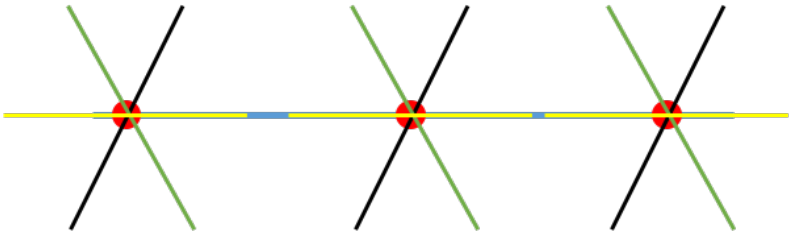


Figura 49: Ejemplo de identificación de un linea en una imagen. Elaboración propia.

7.5.5.3 Aplicación de Hough Lines

La función de Python que aplica este método es la siguiente:

cv2.HoughLines(image, rho, theta, threshold[, lines[, srn[, stn]]]) → lines

los parámetros son:

- image: Imagen binaria en la que identificar lineas.
- rho: Resolución de los píxeles en el acumulador.
- theta: Resolución del ángulo en el acumulador.
- threshold: Valor mínimo de una recta en el acumulador para ser considerado una línea.

Parámetro de salida:

- lines: lista con los valores ρ y θ de cada línea con un acumulador mayor que el *threshold*.

Aplicado al problema que se trata, dado que las lineas son todas verticales, no hace falta que θ pruebe todos los valores porque sólo se buscan líneas verticales. Además da igual como se hayan tomado las fotografías, la orientación de las bandas en la fotografía no depende de como se oriente el móvil sino de como se obtiene la imagen.

7.5.5.4 Aplicación de Hough Lines sobre *threshold* adaptativo de Gauss y detector de bordes Canny.

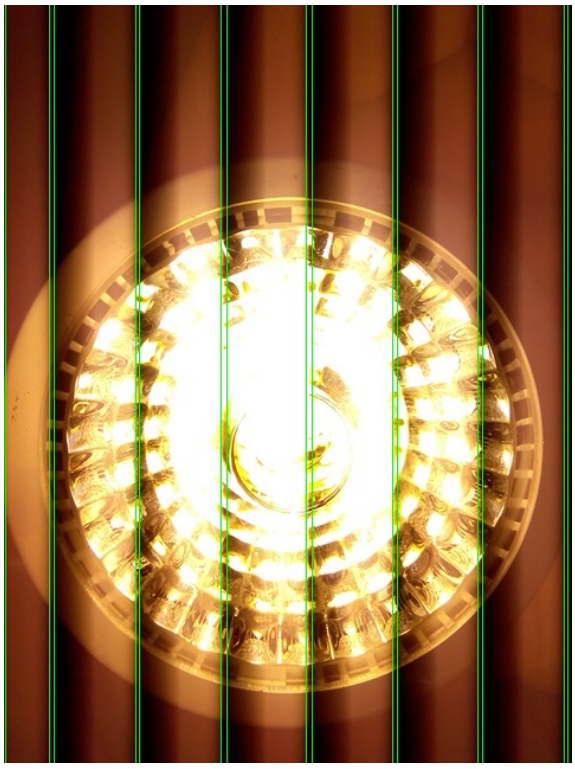


Figura 50: Resultado de Hough lines con filtro Canny a una iluminación de 200 Hz y con *threshold* 160. Elaboración propia.

A priori se ha decidido identificar las líneas de los bordes utilizando Canny, ya que de manera intuitiva al ser los bordes más finos se evita detectar varias líneas en un mismo borde. En la figura 51 se comprueba como afecta el aumento del parámetro *threshold* en la función Hough lines a la identificación de las lineas con 200 Hz de frecuencia. Los resultados muestran como un *threshold* de 100 identifica muchas líneas de forma equivocada, para solucionarlo se tiene que aumentar el *threshold* hasta que identifique solamente las líneas que realmente existen, en el caso de 200 Hz el *threshold* debe ser de 160 ya que con 150 se sigue identificando una linea que no existe, en la figura 51 se ve el proceso hasta conseguir el resultado óptimo.

También se obtiene una lista que contiene la posición o píxel de cada una de las líneas:

lista_líneas_verticales:[1.0, 38.0, 42.0, 111.0, 114.0, 183.0, 188.0, 255.0, 260.0, 328.0, 332.0, 400.0, 404.0, 473.0, 476.0]

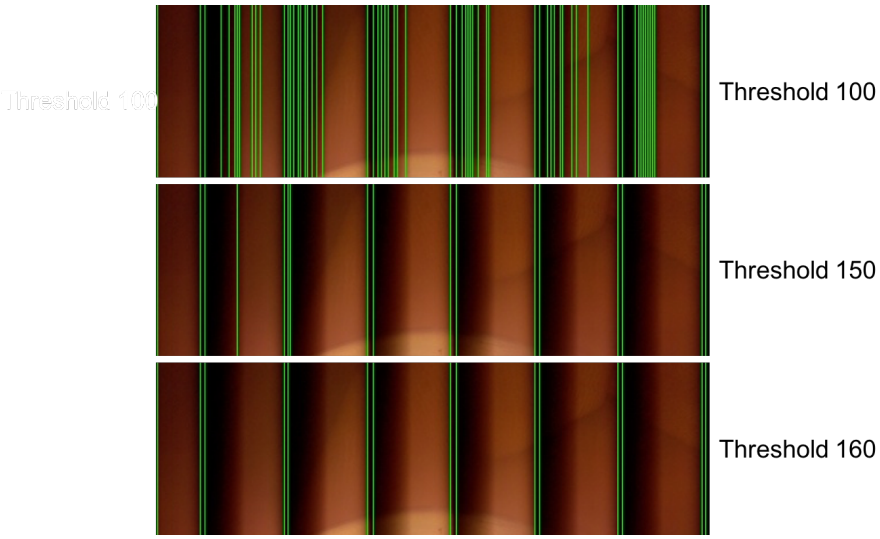


Figura 51: 200 Hz con diferentes *thresholds*, 160 es el que ofrece la mejor solución. Elaboración propia.

El problema surge al realizar *Hough Lines*con *threshold* 160 sobre otras frecuencias, como se puede ver en la figura 52 los resultados no son igual de favorables. En el recorte de 500 Hz se identifica una línea de más en la columna 352, y dos de menos en el recorte de 1000 Hz en las columnas 190 y 219. Esto se debe a que un *threshold* de 160 para 500 es demasiado bajo si se aumentara el *threshold* la linea falsa desaparecería, y un *threshold* de 160 para 1000 Hz es muy alto ya que las líneas no detectadas tienen un acumulador menor a 160, si se bajara el *threshold* las líneas serian detectadas.

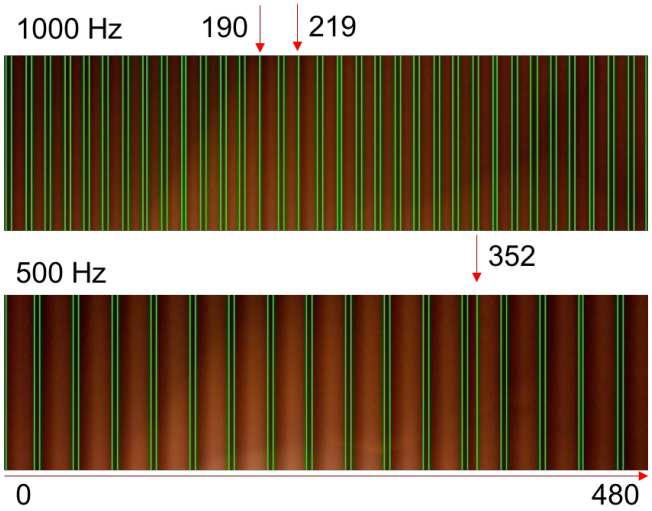


Figura 52: Errores de Hough lines con filtro Canny en iluminación de 500 y 1000 Hz y con *threshold* 160. Elaboración propia.

A pesar de que los errores son de una sola linea en un conjunto de muchas lineas, el error que puede significar es muy alto. Se tienen que agrupar de dos en dos las lineas para identificar el centro de cada borde, si en una de las parejas falta una linea, la media de cada pareja se desplaza. En frecuencias altas el efecto es pequeño, pero en frecuencias bajas como 100 Hz el error es importante:

- las lineas identificadas con *threshold* 160 son: [12.0, 14.0, 156.0, 160.0, 301.0, 304.0, 368.0, 446.0, 450.0]. donde la linea 368 sobra.
- las medias de las parejas queda así: [13.0, 158.0, 302.5, 407.0]. donde la linea 450 ha sido despreciada porque hay un numero impar.
- el grosor medio de las bandas después de eliminar outliers es de 131,33.
- las lineas identificadas con *threshold* 161 son:[12.0, 14.0, 156.0, 160.0, 301.0, 304.0, 446.0, 450.0] donde se han identificado todas las lineas correctamente.
- las medias de las parejas queda así:[13.0, 158.0, 302.5, 448.0].
- el grosor medio de las bandas después de eliminar outliers es de 145.
- el error es del 6,8 %.

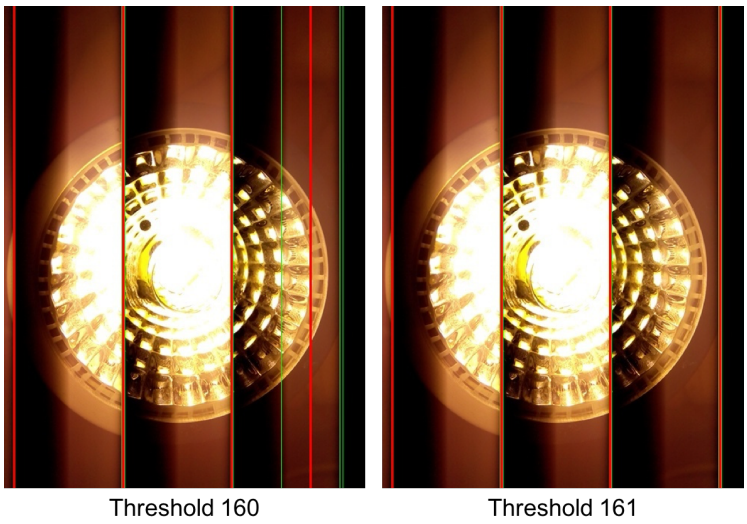


Figura 53: 200 Hz con diferentes *thresholds*, 160 es el que ofrece la mejor solución. Elaboración propia.

El error del 6,8 % se debe solo a una linea identificada de más. A medida que aumenta el número de errores, que es fácil debido a que no se puede adaptar el filtro a cada imagen, aumenta el porcentaje de error. Por esta razón se ha decidido utilizar el algoritmo de detección sin el filtro de canny aunque intuitivamente parezca que vaya a dar problemas por la identificación errónea de multiples líneas.

7.5.5.5 Aplicación de Hough Lines sobre *threshold* adaptativo de Gauss.

El proceso que se usado para obtener las lineas es el siguiente:

1. Abrir la imagen de la iluminación con la frecuencia.

```
def abrir_imagen_Hz(imagen):
  img = cv2.imread(str(imagen)+'Hz.jpeg')
  return img
```

2. Convertir la imagen a escala de grises, es necesario para poder aplicar filtros y transformarla.

```
def convertir_gris(imagen):
  gray = cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
  return gray
```

3. Aplicar el filtro adaptativo de Gauss.

```
def threshold_adaptativo_gauss(imagen):
  thres= cv2.adaptiveThreshold(imagen,255,\
  cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
  return thres
```

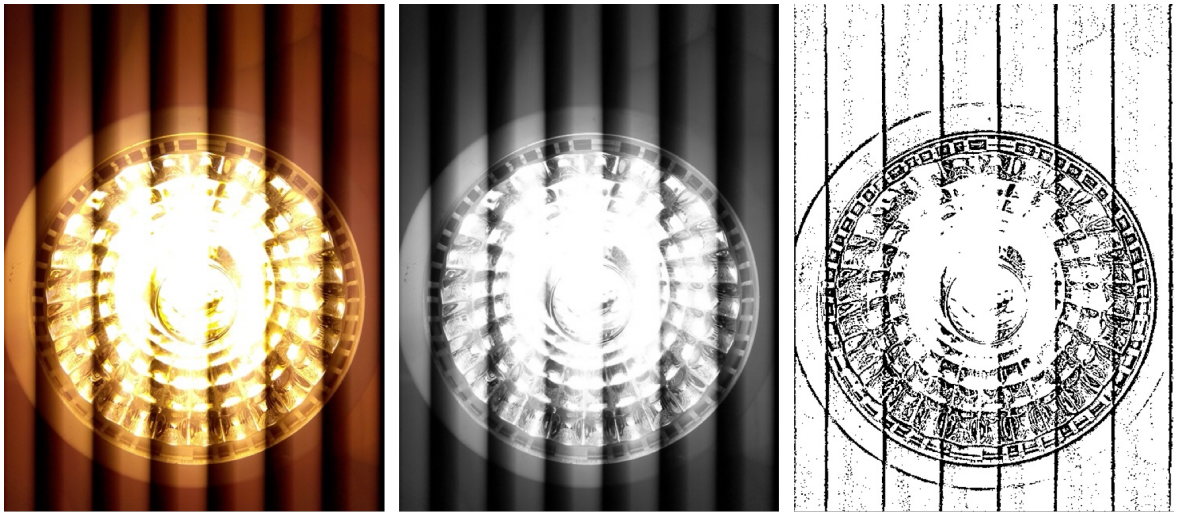


Figura 54: Primeros tres pasos del proceso con 200 Hz. Elaboración propia.

4. Invertir la imagen, el algoritmo de Hough localiza pixeles blancos y no negros, por lo tanto hay que invertir la imagen.

```
def invertir_negro_blanco(imagen):
  invertida=(255-imagen)
  return invertida
```

5. Aplicar el algoritmo de Hough.

```
def recuperar_lineas_imagen(imagen):
  lista=[]
  lines = cv2.HoughLines(imagen,1,np.pi,250)
  for linea in lines[0]:
    lista.append(linea[0])
  return lista,lines
```

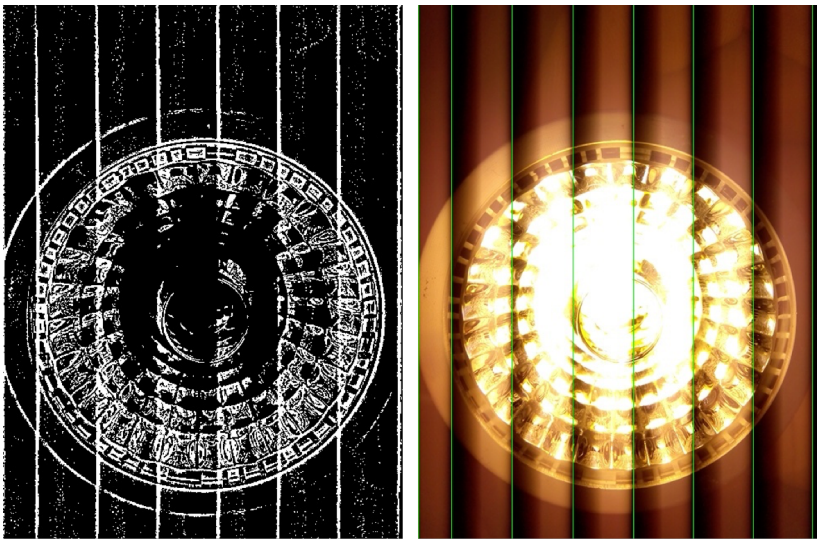


Figura 55: Pasos 4 y 5 del proceso con 200 Hz. Elaboración propia.

6. Obtener los grosores de las bandas restando las lineas.

```
def ancho_bandas(lineas):
  lineas=sorted(lineas)
  anchos=[]
  linea_anterior=lineas[0]
  for linea_actual in lineas[1:]:
    anchos.append(linea_actual-linea_anterior)
    linea_anterior=linea_actual
  return anchos
```

7. Eliminar los grosores outliers.

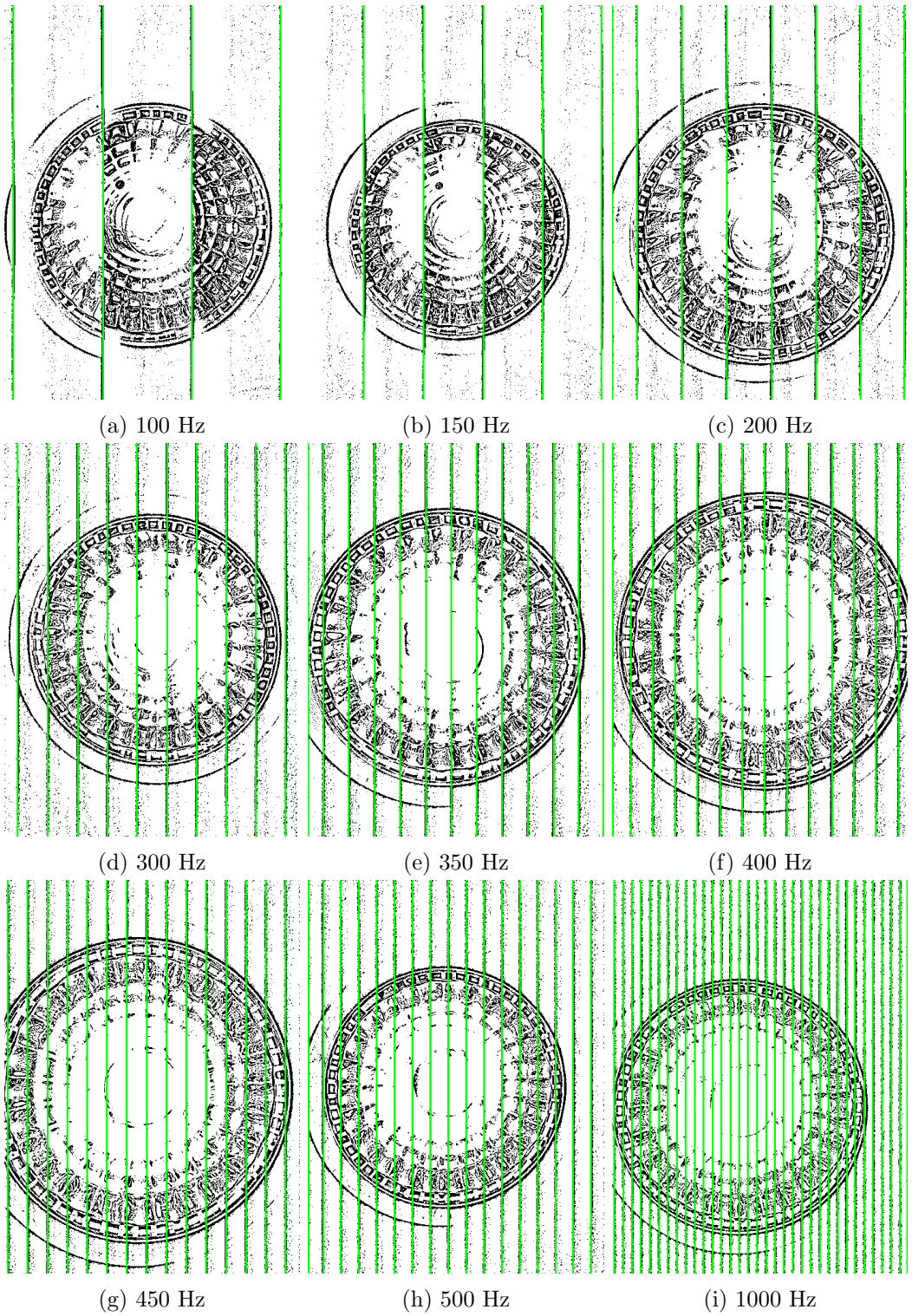
```
def retirar_outliers(anchos):
  percentiles=np.percentile(anchos,[25,50,75])
  Q1=percentiles[0]
  Q3=percentiles[2]
```

```
    IQR=Q3-Q1
    limiteinf=Q1-IQR*1.5
    limitesup=Q3+IQR*1.5
    lista_sin_outliers=[]
    for x in anchos:
      if x>=limiteinf and x<=limitesup:
        lista_sin_outliers.append(x)
    media=np.mean(lista_sin_outliers)
    return lista_sin_outliers,media
```

los resultados obtenidos para la frecuencia de 200Hz en cada apartado intermedio son los siguientes:

```
para la frecuencia 200 Hz:
las lineas encontradas son:
[474.0, 40.0, 403.0, 113.0, 331.0, 0.0, 186.0, 259.0]
los anchos totales son:
[40.0, 73.0, 73.0, 73.0, 72.0, 72.0, 71.0]
los anchos sin outliers son:
[73.0, 73.0, 73.0, 72.0, 72.0, 71.0]
la media es:
72.3333
```

En las figura 56 se puede comprobar el resultado de aplicar Hough Lines a varias de las frecuencias una vez se ha utilizado el *threshold* adaptativo de Gauss con Kernel igual a 11.



(a) 100 Hz (b) 150 Hz (c) 200 Hz

(d) 300 Hz (e) 350 Hz (f) 400 Hz

(g) 450 Hz (h) 500 Hz (i) 1000 Hz

Figura 56: Resultado de Hough Lines aplicado directamente después de un *threshold* adaptativo de Gauss. Elaboración propia.

7.5.5.6 Resultados

Los resultados obtenidos con este método son muy precisos y parecidos a los resultados manuales utilizados para comparar. En la tabla 4 se muestra el grosor de los periodos y la desviación estándar para cada frecuencia, con este método se puede comprobar como la desviación estándar se ha reducido mucho y por lo tanto el proceso es muy fiable para diferenciar las distintas frecuencias. La desviación estándar más grande es de 1,53 para una frecuencia de 100Hz, influye el hecho de que sea el mayor grosor y por lo tanto pequeñas variaciones generan diferencias mayores. Llega a ser incluso 0 en dos de las frecuencias, la conclusion de la baja desviación es que el algoritmo es muy robusto, y daría valores similares en distintas situaciones.

Frecuencia (Hz)	**Media Ancho Periodo (px)**	**Desviación Estandar (px)**
100	144.3	1.53
150	95.7	0.58
200	72.3	0.82
250	58.0	0.00
300	48.3	0.50
350	41.2	0.75
400	36.3	0.62
450	32.2	0.58
500	29.0	0.00
1000	14.5	0.51

Tabla 4: Resultados con algoritmo Threshold Adaptativo. Elaboración propia.

También se ha comparado los resultados con los datos obtenidos manualmente. En la tabla 5 se puede ver como el error absoluto máximo es de 0.3 píxeles, y el error porcentual máximo de 0.88 %. El error porcentual va creciendo a medida que aumenta la frecuencia porque los grosores de los periodos disminuyen de forma que el error se vuelve más representativo.

Frecuencia (Hz)	**Real (px)**	**Calculado (px)**	**Error (px)**	**Error (%)**
100	144.7	144.3	0.3	0.23
150	96.0	95.7	0.3	0.35
200	72.7	72.3	0.3	0.46
250	58.0	58.0	0.0	0.00
300	48.1	48.3	0.2	0.40
350	41.3	41.2	0.1	0.29
400	36.2	36.3	0.1	0.19
450	32.5	32.2	0.3	0.88
500	29.2	29.0	0.2	0.57
1000	14.4	14.5	0.1	0.55

Tabla 5: Comparativa entre Threshold Adaptativo y los datos manuales. Elaboración propia.

Como se ha comprobado al obtener los datos manuales se puede ver como desciende el grosor a medida que aumenta la frecuencia, Si se dobla la frecuencia el grosor de

las franjas se divide entre dos. En la figura 58 se puede ver un scatterplot del grosor en función de la frecuencia donde la función de regresión da el siguiente resultado y coeficiente de determinación.

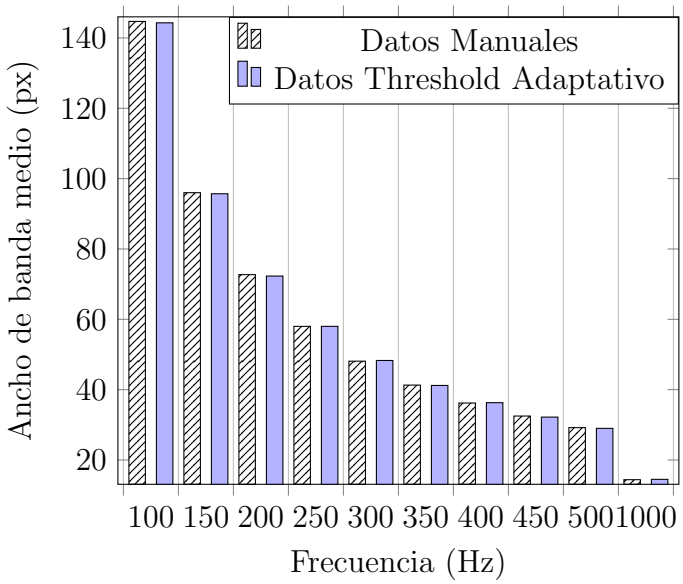


Figura 57: Diagrama de barras Frecuencia-Ancho de Banda con datos manuales y Threshold Adaptativo. Elaboración propia.

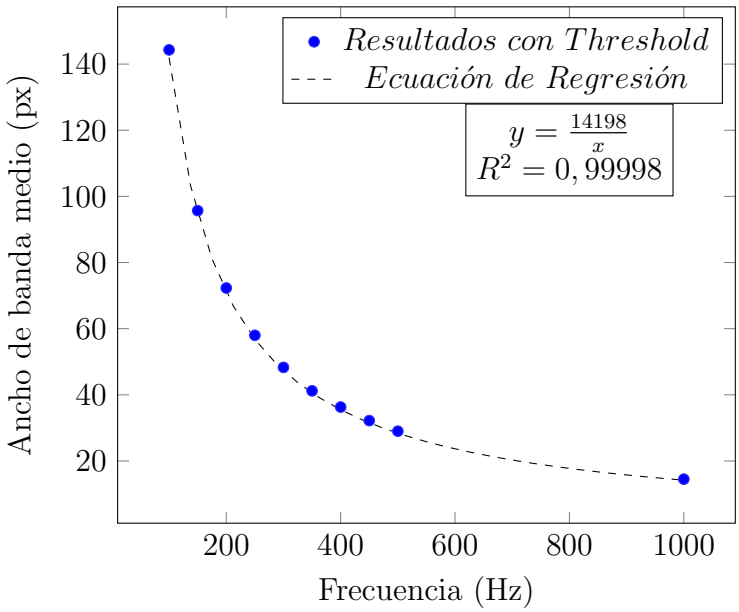


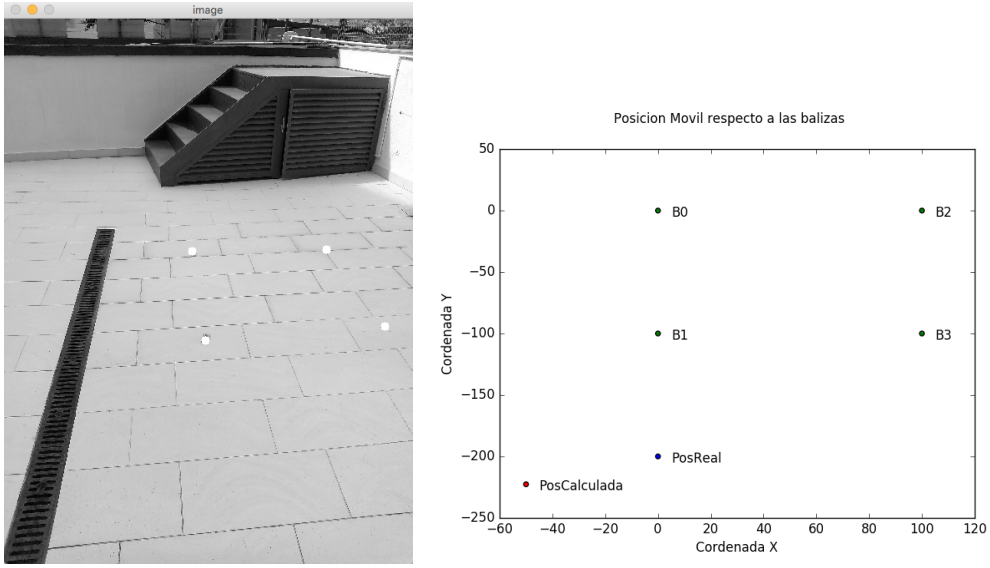
Figura 58: Diagrama de puntos Frecuencia-Ancho de Bandas con Threshold Adaptativo. Elaboración propia.

8. Análisis de resultados

De todo el proyecto realizado se pueden obtener una serie de resultados generales que sirven para resumir el estudio y comprender las líneas generales que este proyecto pretende desarrollar. A continuación se evaluarán y analizarán los resultados recogidos en cada uno de los apartados.

8.1. Posicionamiento respecto a referencias

El primer problema al que se ha intentado buscar una solución ha sido el posicionar un dispositivo móvil utilizando una fotografía tomada por la cámara en la que aparecen 3 o más referencias. Los resultados no han sido los esperados pues en documentos anteriores se han conseguido unos resultados de mayor calidad[Kuo et al., 2014]. Estos resultados muestran muy poca precisión al posicionar un dispositivo móvil y careciendo además de robustez, pues si la foto se hace con una inclinación distinta o desplazado unos milímetros a la derecha la respuesta del programa es completamente distinta y errónea. A pesar de que se ha intentado mejorar el programa, los resultados no han mejorado de forma significativa. El proceso de cálculo se ha realizado manualmente y se han obtenido los mismos resultados que con el algoritmo, por lo tanto parece que el apartado numérico es correcto. En la figura 59 hay un ejemplo de una fotografía tomada por la cámara del teléfono y del resultado del algoritmo de posicionamiento, así como de las cuatro balizas con sus posiciones conocidas.



(a) Fotografía realizada (b) Resultado del algoritmo de localización

Figura 59: Ejemplo del proceso de posicionamiento. Elaboración propia.

Posibles razones por las que no se ha conseguido un resultado igual al de los documentos anteriores se debe a que el dispositivo móvil utilizado en este proyecto es de gama baja mientras que los dispositivos utilizados en documentos antiguos teniían las mejores especificaciones del mercado.

8.2. Identificación de las luces con el sensor de luz del dispositivo

Los resultados al intentar identificar el dispositivo móvil utilizando el sensor han sido infructuosos, pues los datos recopilados por el sensor no se podían interpretar ya que no había ninguna relación con la frecuencia a la que estaba la iluminación LED. Ésto se debe a que el sensor del móvil tiene una frecuencia de muestreo muy inferior a 100 Hz y por lo tanto es incapaz de registrar correctamente frecuencias de esa magnitud. En trabajos anteriores, que se han utilizado como referencia, se han obtenido mejores resultados utilizando dispositivos externos al móvil como sensor de luz con una frecuencia de muestro mucho mayor y siendo capaz de identificar frecuencias de esa escala[Luo et al., 2014]. En la figura 60 se puede observar los resultados obtenidos del sensor. Como se ha explicado anteriormente no se ha podido interpretar una frecuencia de 100 Hz.

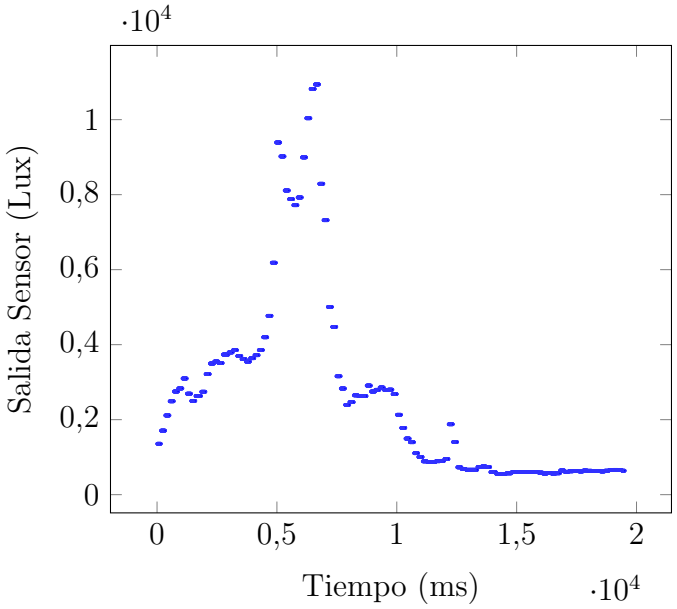


Figura 60: Diagrama de puntos donde se muestra la iluminación recogida por el sensor de luz a 100 Hz. Elaboración propia.

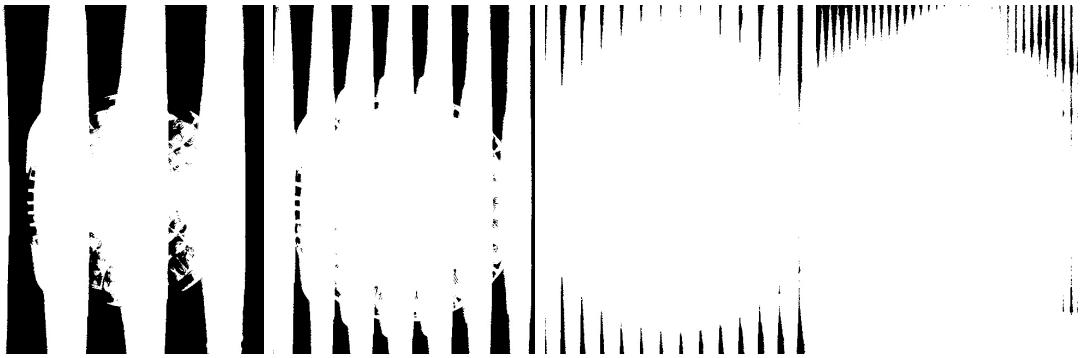
8.3. Identificación de las luces con la cámara fotográfica del móvil

Los resultados al sacar identificar la frecuencia de un LED utlizando la camára fotográfica de un teléfono móvil han sido muy positivos. Tan solo el hecho de analizar las fotografías sin modificar permite observar que hay una relación entre la frecuencia y la imagen, como se puede ver reflejado en la figura 61, donde las bandas se van haciendo más estrechas a medida que aumenta la frecuencia.

(a) LED a 100 Hz (b) LED a 200 Hz (c) LED a 400 Hz (d) LED a 1000 Hz

Figura 61: Fotografías tomadas por el télefono móvil a distintas frecuencias. Elaboración propia.

A partir de ahí se ha procedido a realizar diferentes transformaciones a las imágenes con el fin de identificar de forma automática la frecuencia de la iluminación LED. El primer proceso ha consistido en aplicar un *threshold* simple a la imagen donde la escala de grises se ha ido transformado en una imagen en blanco y negro en función de si el valor de cada píxel era superior o inferior al valor *thresh*. Un píxel en una imagen gris puede ir de 0 a 255 o en otras palabras de negro a blanco.



(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

Figura 62: Distintas frecuencias con filtro *threshold* simple, valor *thresh*:25. Elaboración propia.

El resultado de este proceso no es muy preciso ni robusto, ciertas frecuencias han sido identificadas correctamente pero otras varían mucho respecto al valor real. Ésto se debe a que el *threshold* se aplica exactamente igual a todas las imágenes y no tiene en cuenta las particularidades de cada situación. Como se puede comprobar en la figura 62 un *thresh* de 25 identifica muy bien las bandas para 100 Hz y 200 Hz, pero ha medida que aumenta la frecuencia los resultados dejan de ser fiables. En la

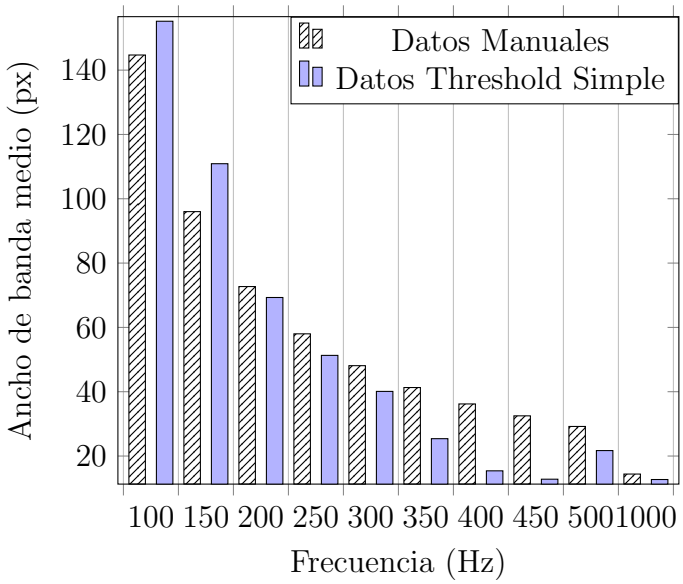
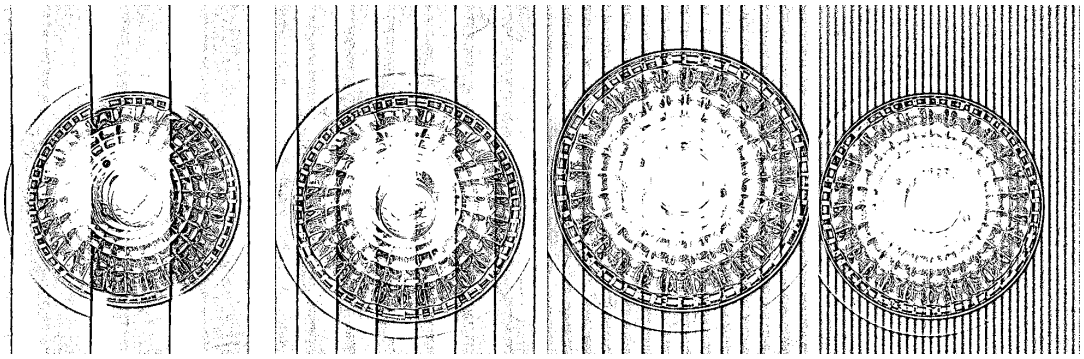


Figura 63: Diagrama de barras Frecuencia-Ancho de Banda con datos manuales y Threshold Simple. Elaboración propia.

figura 63 donde se comparan los resultados con los valores reales se vuelve a llegar a la misma conclusión, es decir, a frecuencias bajas el programa detecta correctamente el ancho de banda pero se vuelve muy impreciso cuando aumenta la frecuencia.



(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

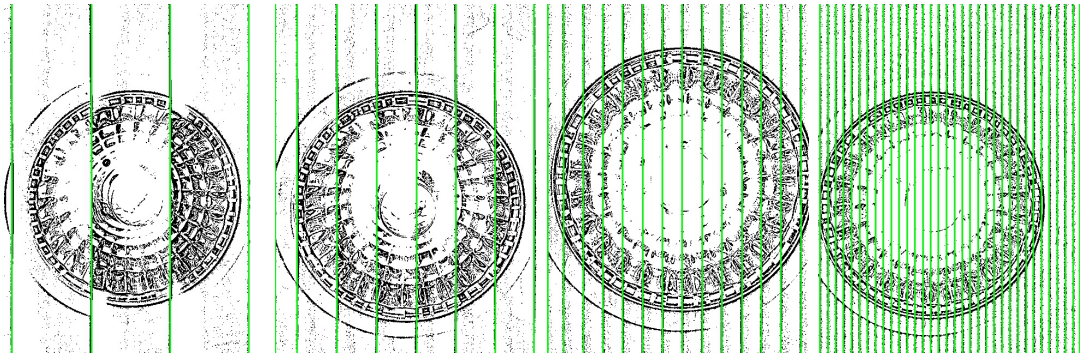
Figura 64: Distintas frecuencias con filtro *threshold* adaptativo Gauss, *blocksize* : 11. Elaboración propia.

Para solucionar este problema se ha buscado un *threshold* que sea capaz de adecuarse a cada situacion. Para ello se ha utilizado un filtro conocido como *threshold* adaptativo de Gauss, el cual usa los píxeles vecinos del píxel que seesta tratando para asignarle un determinado valor.

El resultado de este proceso ha sido extremadamente bueno, en la figura 64 se pueden observar las líneas de inicio y fin de cada periodo, el está formado por una banda encendida y una apagada. Los resultados son robustos y correctos ya que funciona bien con todas las frecuencias probadas y además posiciona correctamente las líneas.

Una vez se ha obtenido una imagen donde se ve claramente las rectas de inicio y fin de periodo ha hecho falta identificarlas numéricamente. Para ello se ha empleado

un fitro de Hough, que identificar rectas en imágenes y devuelve sus respectivas posiciones.



(a) 100 Hz (b) 200 Hz (c) 400 Hz (d) 1000 Hz

Figura 65: Distintas frecuencias con filtro *threshold* adaptativo Gauss, *blocksize* : 11. Elaboración propia.

Los resultados como se pueden ver en la figura 65 son extremadamente buenos pues presentan un error casi nulo si se comparan con los resultados reales obtenidos manualmente. Esta comparación se puede comprobar en la figura 66. Además, existe una alta relación entre el incremento de la frecuencia y el ancho de los periodos. La regresión potencia mostrada en la figura 67 tiene un coeficiente de determinación del 0.99998 por lo que es correcto establecer que el ancho de banda y la frecuencia están relacionadas con esa ecuación para el teléfono móvil utilizado.

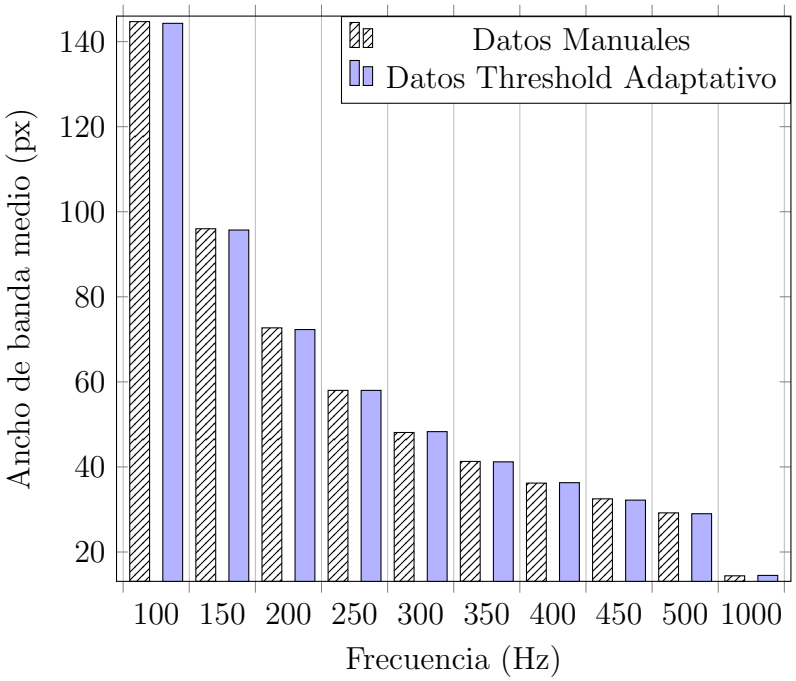


Figura 66: Diagrama de barras Frecuencia-Ancho de Banda con datos manuales y Threshold Adaptativo. Elaboración propia.

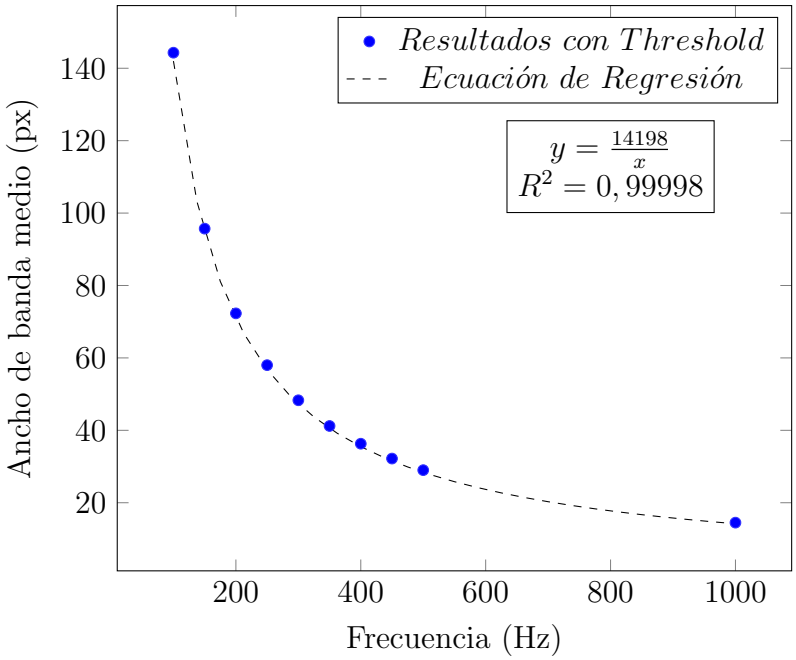
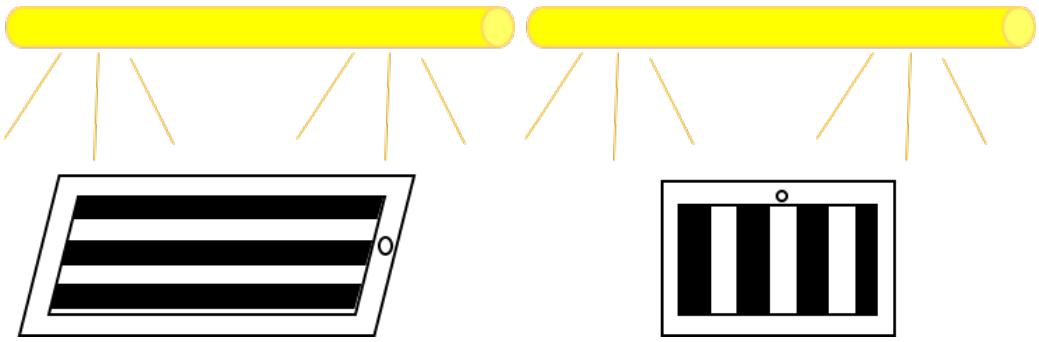


Figura 67: Diagrama de puntos Frecuencia-Ancho de Bandas con Threshold Adaptativo. Elaboración propia.

Este sistema de posicionamiento ofrece además la ventaja de que la orientación de las bandas es siempre la misma como se puede observar en la figura 68. Éstas se obtienen por el sensor CMOS que utilizan los teléfonos moviles y su efecto *rolling shutter* por lo que no dependen de la orientación. Algo similar ocurre con la distancia entre el teléfono y la iluminación pues ésta no afecta al ancho de banda porque solo depende del tiempo de exposición de cada columna de píxeles, y es algo constante en cada dispositivo.



(a) Teléfono paralelo a la luz (b) Teléfono perpendicular a la luz

Figura 68: Ejemplo gráfico de la relación entre orientación y las bandas obtenidas. Elaboración propia.

9. Conclusiones y trabajos futuros

9.1. Conclusiones

La conclusión extraída después de realizar el proyecto es que es posible localizar un dispositivo móvil mediante la iluminación, sin embargo esta afirmación tiene una serie de matices. El proyecto tiene una validez limitada en tanto en cuanto no se desarrollen ciertos aspectos con mayor profundidad. De los objetivos descritos al principio del proyecto, a pesar de que ha perseguido el cumplimiento de todos ellos, sólo dos han conseguido obtener resultados verdaderamente satisfactorios.

El posicionamiento respecto a las referencias conocidas en el espacio no parece viable debido a los resultados obtenidos, pues hay una falta de conocimiento práctico sobre como funcionan las lentes de un dispositivo móvil comparado con su teoría. El proceso llevado a cabo parece correcto desde el punto de vista numérico siendo prueba de ello las diferentes comprobaciones que se han realizado. Sin embargo, Las conclusiones que podemos extraer de las mismas es que posiblemente la calidad del teléfono utilizado no permita identificar con precisión las referencias ya que se desconocen ciertos parámetros físicos del dispositivo necesarios para situar correctamente el teléfono móvil.

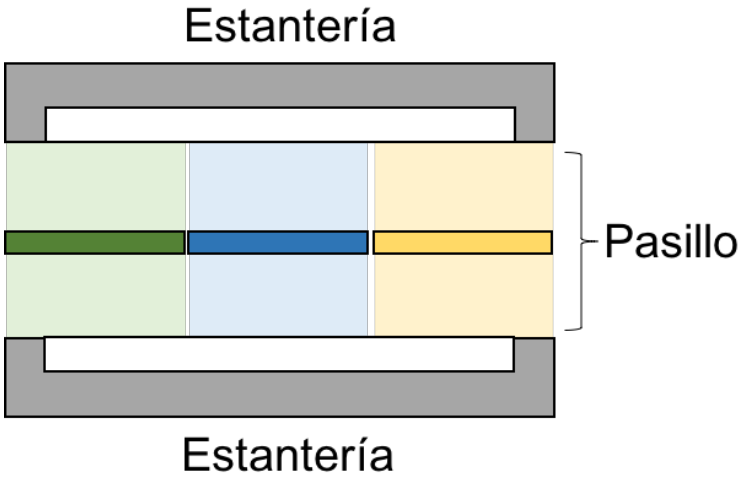


Figura 69: Pasillo de un supermercado con LEDs que delimitan la localización. Elaboración propia.

La creación de un LED con un PWM (ciclos de encendido y apagado) ajustable ha sido una de las partes más complicadas del proyecto pues la falta de conocimiento de electrónica ha hecho que resulte difícil encontrar una solución factible. La iluminación utilizada no tiene las mismas características que los LEDs presentes en espacios cerrados reales, donde son mucho más grandes y potentes. Los LEDs que se han utilizado son de 12 Voltios y 7 Vatios por la facilidad que ofrecían a la hora de trabajar con ellos en el laboratorio. Después de realizar el proyecto, se ha considerado que una mejora sustancial podría haber sido el hecho de utilizar LEDs de 220 Voltios con tecnología Zigbee, pues el montaje hubiera sido muy simple y solo hubiera requerido programar el código para modificar la frecuencia. Además, esto hubiera permitido probar el impacto de varios LEDs con diferentes PWM.

En el proyecto realizado se ha determinado que el sensor de luz de un teléfono móvil no tiene las características necesarias para identificar el PWM de un LED. La utilidad del sensor de luz en un teléfono móvil es adaptar el brillo de la pantalla deduciendo que la velocidad de muestreo puede ser baja y no sería necesario alcanzar una gran exactitud para realizar su función de forma correcta. Utilizar la cámara del móvil para posicionar un dispositivo móvil parece un método con mucha visión de futuro. Los resultados obtenidos, aunque tengan que ser probado en condiciones reales, son alentadores. En este sentido,se ha comprobado que un teléfono móvil realiza fotografías de los LEDs que son identificables por algoritmos. Además, la precisión de los resultados es muy alta y parece que podría ser usado en un futuro para localizar un teléfono móvil en lugares cerrados mediante diferentes PWMs. Debido a que los resultados de posicionar respecto a referencias no han sido muy positivos, se considera que la manera de proceder es utilizando LEDs que delimiten zonas o áreas, como se puede ver reflejado en la figura 69 donde los diferentes colores representan las frecuencias y su zona de localización.

9.2. Trabajos futuros

9.2.1. Desarrollo de una aplicación para android

Para poder posicionar un dispositivo móvil empleando su cámara es necesario programar utilizando Java. La librería esencial utilizada en este proyecto es Open CV, al estar disponible también para Java es posible desarrollar los mismos programas para un dispositivo móvil con sistema operativo Android. Además, también se tiene que desarrollar la localización en tiempo real donde se van obteniendo fotografías a una cierta velocidad y se actualiza la posición del dispositivo en un mapa. Los demás sensores del teléfono móvil tambien deberían ser aprovechados para intentar alcanzar una solución con mayor exactitud.

La carga computacional del proyecto se ha intentado reducir en aquellos puntos en los que ha sido posible: la función hough lines se ha programado para buscar exclusivmente líneas verticales en vez de buscar todas las posibles líneas. Ello hace que el cálculo sea 180 veces menos pesado que si se programa para buscar rectas con todos los posibles ángulos (de 0 a 180). Además, cuando se ha realizado el *threshold* adaptativo se ha intentado reducir al máximo el tiempo de cálculo, tratando de disminuir el tamaño de la matriz de píxeles vecinos al mínimo necesario. Igualmente se tiene que llevar a cabo una optimización de los algoritmos para reducir aún más la carga computacional.

9.2.2. Impacto de varias fuentes de iluminación con PWM diferente

En este proyecto no se ha podido realizar la identificación de frecuencias cuando la iluminación recibida en el dispositivo móvil proviene de más de una fuente. Sin embargo, este estudio es esencial para poder utilizar los conceptos aprendidos en un local cerrado, donde habrá diferentes fuentes de luz con diversas frecuencias una al lado de otra.

9.2.3. Desarollo de iluminación aplicable a locales

La iluminación modificada en este proyecto es un LED de pequeña potencia que se ha utilizado por su fácil montaje y el escaso coste de los componentes. Para poder utilizar este sistema en locales cerrados, se tendría que conseguir modificar la iluminación de muchos LEDs al mismo tiempo que estan conectados a la red eléctrica. Para ello, haría falta varios micros gestionando las diferentes zonas pudiéndose utlizar también un micro con muchas salidas. Sin embargo, ésto supondría el uso de numerosos cables para llegar a cada uno de los LEDs . A pesar de lo mencionado anteriormente, el problema que presentan hoy en día las bombillas LED es que funcionan en continua y la red doméstica está en alterna, por tanto, las bombillas llevan integrado un conversor AC/DC al que no se puede tener acceso. Ante la problemática planteada, lo mas rentable sería instalar una fuente de continua en un local cerrado por el ahorro que produce el consumo energético.

Una solución al problema anterior podría ser el diseño de bombillas con tecnología Zigbee como usa Philips Hue. Esta tecnología, parecida al bluetooth, permite a los LEDs seguir conectados a corriente alterna y recibir el PWM de encendido/apagado de forma inalámbrica.

10. Anexos

10.1. Programa para la selección manual de las balizas

```
# -*- coding: utf-8 -*-
#alg_posicionamiento_seleccionar_balizas.py

"""
Descripcion
-----------------------------------------------
Sirve para Seleccionar manualmente las balizas
en una imagen y recuperar las coordenadas.

Aparece una imagen donde hay que seleccionar 3 o
m'aspuntos de referencia. Un punto blanco aparece
donde se ha seleccionado.
Una vez seleccionados pulsar -Esc.
-----------------------------------------------
"""

"""Modulos Importados"""
import cv2
import numpy as np
"""-----------------"""

coord_referencias=[]
distancia_focal=30
Zf=distancia_focal #simplificar variable

def draw_circle(event,x,y,flags,param):
    global puntos
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(foto_camara_reducida,(x,y),5,(255,0,0),-1)
        coord_referencias.append([x,y,Zf])

print "Selecciona Referencias:"

foto_camara=cv2.imread('bal1.jpg',0)
foto_camara_reducida=cv2.resize(foto_camara,(0,0),fx=0.2,fy=0.2)
print foto_camara_reducida.shape
cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_circle)
while(1):
    cv2.imshow('image',foto_camara_reducida)
    k = cv2.waitKey(20) & 0xFF
    if k == 27:
        break
cv2.destroyAllWindows()

def devolverBalizas(x=1):
    return puntos
def devolverTamano(x=1):
    return (foto_camara_reducida.shape)
```

10.2. Programa para el cálculo de la posición del dispositivo

```
# -*- coding: utf-8 -*-
#alg_posicionamiento_calculos.py


"""
Descripcion
--------------------------------------------------
Importa las posiciones de las referencias del mo-
dulo alg_posicionamiento_seleccionar_balizas.py.

Calcula la posicion del dispositivo respecto a las
referencias.
--------------------------------------------------
"""

"""Modulos Importados"""
import itertools
import numpy
import scipy.optimize as optimize
import alg_posicionamiento_seleccionar_balizas as selBalizas
"""------------------"""


Zf=31.18
Factor=1


"""Constantes no Inputs"""

posmovil=[0,-200,125]
r=[[0,0,0],[0,-100,0],[100,0,0],[100,-100,0]]

"""Recupera el tamao de la imagen (cnst) y la posicion de las balizas en la
 ↪  imagen"""
tamanoImagen=list(selBalizas.devolverTamano())

tdibujo=selBalizas.devolverBalizas()
print tdibujo
t=[]
for x in tdibujo:
    t.append([tamanoImagen[1]-x[0],x[1],Zf])


"""calcula lista con las balizas:0,1,2,3..."""
numbalizas=len(t)
listabalizas=[]

i=0
for x in range(numbalizas):
    listabalizas.append(x)
    i+=1
lista=list(itertools.combinations(listabalizas, 2))

"""Calcula la constantes de proporcionalidad usando semejanza"""
def Constantes(K):
```

```
55      c=0
56      for pareja in lista:
57          vect_balizas=numpy.subtract(r[pareja[1]],r[pareja[0]])
58          dist_real_bal=numpy.dot(vect_balizas,vect_balizas)
59          vect_img_baliza0=t[pareja[0]]
60          vect_img_baliza1=t[pareja[1]]
61          vib0_2=numpy.dot(vect_img_baliza0,vect_img_baliza0)
62          vib1_2=numpy.dot(vect_img_baliza1,vect_img_baliza1)
63          vib01=numpy.dot(vect_img_baliza0,vect_img_baliza1)
64          K0=K[pareja[0]]
65          K1=K[pareja[1]]
66          c+=((K0**2) *vib0_2+(K1**2)*vib1_2-(2*K0*K1*vib01)-dist_real_bal)**2
67      return c
68  resultK = optimize.minimize(Constantes,[1,1,1,1], method='nelder-mead',
     ↪  options={'xtol': 1e-8, 'disp': True})
69
70  constantesK=resultK.x
71  print constantesK
72  for x in range(len(constantesK)):
73      constantesK[x]=constantesK[x]*Factor
74  print constantesK
75  """Calcula la matriz de translacion"""
76
77  def Translacion(T):
78      x=0
79      for baliza in listabalizas:
80          dist_bal_proy=numpy.subtract(T,r[baliza])
81          modulo_dbp=numpy.dot(dist_bal_proy,dist_bal_proy)
82          vect_img_baliza0=t[baliza]
83          vib0_2=numpy.dot(vect_img_baliza0,vect_img_baliza0)
84          x+=(modulo_dbp-(((constantesK[baliza])**2)*vib0_2))**2
85      return x
86
87  resultT = optimize.minimize(Translacion,[1,1,1], method='nelder-mead',
     ↪  options={'xtol': 1e-8, 'disp': True})
88
89  posicion=resultT.x
90  print 'la posicion calculada es: ' + str(posicion)
91  print 'la posicion real es: ' + str(posmovil)
92  def devolverPosCal(x=1):
93      return posicion
94  def devolverPosReal(x=1):
95      return posmovil
96  def devolverPosBal(x=1):
97      return r
98  def devolverPosBalImg(x=1):
99      return t
100 def devolverTamanoImagen(x=1):
101     return tamanoImagen
102 """Calcula la desviacion"""
103 vecdesviacion=numpy.subtract(posmovil[:2],posicion[:2])
104 print vecdesviacion
105 cuadrados=numpy.dot(vecdesviacion,vecdesviacion)
106 desv=numpy.sqrt(cuadrados)
107 print Zf
108 print Factor
109 print ' el error es de: ' + str(desv)
```

10.3. Programa para la representación gráfica del cálculo

```
# -*- coding: utf-8 -*-
#alg_posicionamiento_dibujar.py

"""
Descripcion
--------------------------------------------------
Importa del modulo alg_posicionamiento_calculos.py
datos y los representa gr'aficamente para que se
puedan interpretar m'as facil.
--------------------------------------------------
"""

"""Modulos Importados"""
import numpy
from scipy.optimize import fsolve
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import cv2
import alg_posicionamiento_calculos.py as caculos
"""_________________"""
t=calculos.devolverPosBalImg()
tamanoImagen=calculos.devolverTamanoImagen()
def dibujoFoto(z=1):
  tx=[]
  ty=[]
  colores=['r']
  for x in t:
    tx.append(x[0])
    ty.append(x[1])

  for c in range(1,len(t)):
    colores.append('g')
  fig=plt.figure(1)
  fig.suptitle('Coordenadas de Balizas en fotografia')
  ax = fig.add_subplot(111)
  ax.set_xlabel('Cordenada X')
  ax.set_ylabel('Cordenada Y')
  plt.scatter(tx,ty,c=colores)
  plt.axhline(0, color='green')
  plt.axvline(0, color='green')
  for b in range(len(t)):
    ax.text(tx[b]+5,ty[b]-5,'B'+str(b))
  plt.show()
dibujoFoto()
puntosdibujarx=[]
puntosdibujary=[]
coloresdibujar=[]
referencias=calculos.devolverPosBal()
posMovReal=calculos.devolverPosReal()
posMovCal=calculos.devolverPosCal()
for x in referencias:
  puntosdibujarx.append(x[0])
  puntosdibujary.append(x[1])
  coloresdibujar.append('g')
```

```
56  puntosdibujarx.append(posMovReal[0])
57  puntosdibujary.append(posMovReal[1])
58  coloresdibujar.append('b')
59  puntosdibujarx.append(posMovCal[0])
60  puntosdibujary.append(posMovCal[1])
61  coloresdibujar.append('r')
62
63  def dibujarMapa(z=1):
64    fig2=plt.figure(2)
65    fig2.suptitle('Posicion Movil respecto a las balizas')
66    ax2=fig2.add_subplot(111)
67    ax2.set_xlabel('Cordenada X')
68    ax2.set_ylabel('Cordenada Y')
69    plt.scatter(puntosdibujarx,puntosdibujary,c=coloresdibujar)
70    for b in range(len(puntosdibujarx)-2):
71      ax2.text(puntosdibujarx[b]+5,puntosdibujary[b]-5, 'B'+str(b))
72    ax2.text(puntosdibujarx[-2]+5,puntosdibujary[-2]-5, 'PosReal')
73    ax2.text(puntosdibujarx[-1]+5,puntosdibujary[-1]-5, 'PosCalculada')
74    plt.show()
75  dibujarMapa()
```

10.4. Programa definitivo para determinar el ancho de banda

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import cv2
import numpy as np

def ancho_bandas(lineas):
        lineas=sorted(lineas)
        anchos=[]
        linea_anterior=lineas[0]
        for linea_actual in lineas[1:]:
                #print linea_actual-linea_anterior
                anchos.append(linea_actual-linea_anterior)
                linea_anterior=linea_actual
        return anchos

def retirar_outliers(anchos):
        percentiles=np.percentile(anchos,[25,50,75])
        Q1=percentiles[0]
        Q3=percentiles[2]
        IQR=Q3-Q1
        limiteinf=Q1-IQR*1.5
        limitesup=Q3+IQR*1.5
        lista_sin_outliers=[]
        for x in anchos:
                if x>=limiteinf and x<=limitesup:
                        lista_sin_outliers.append(x)
        media=np.mean(lista_sin_outliers)
        return lista_sin_outliers,media
def threshold_adaptativo_gauss(imagen):
        thres= cv2.adaptiveThreshold(imagen,255,
 ↪  cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2)
        return thres

def convertir_gris(imagen):
        gray = cv2.cvtColor(imagen,cv2.COLOR_BGR2GRAY)
        return gray

def invertir_negro_blanco(imagen):
        invertida=(255-imagen)
        return invertida

def abrir_imagen_Hz(imagen):
        img = cv2.imread(str(imagen)+'Hz.jpeg')
```

```
        return img

def recuperar_lineas_imagen(imagen):
        lista=[]
        lines = cv2.HoughLines(imagen,1,np.pi,250)
        for linea in lines[0]:
                lista.append(linea[0])
        return lista,lines

def dibujar_lineas_guardar(lines,imagen,nombre_guardar):
        for rho,theta in lines[0]:
                a = np.cos(theta)
                b = np.sin(theta)
                x0 = a*rho
                y0 = b*rho
                x1 = int(x0 + 1000*(-b))
                y1 = int(y0 + 1000*(a))
                x2 = int(x0 - 1000*(-b))
                y2 = int(y0 - 1000*(a))
                cv2.line(imagen,(x1,y1),(x2,y2),(0,255,0),1)
        cv2.imwrite(nombre_guardar,imagen)

listamedias=[]
listavalores=[]

frecuencias=[100,150,200,250,300,350,400,450,500,1000]
for x in frecuencias:
        print 'para la frecuencia '+ str(x)+' Hz:'
        imagen_original=abrir_imagen_Hz(x)
        imagen_gris=convertir_gris(imagen_original)
        imagen_thres_gauss=threshold_adaptativo_gauss(imagen_gris)
        imagen_invertida=invertir_negro_blanco(imagen_thres_gauss)
        lineas_bandas,lines=
 ↪  recuperar_lineas_imagen(imagen_invertida)
        print 'las lineas encontradas son:'
        print lineas_bandas
        lista_ancho_bandas=ancho_bandas(lineas_bandas)
        print 'los anchos totales son:'
        print lista_ancho_bandas
        ancho_sin_outliers,media=
 ↪  retirar_outliers(lista_ancho_bandas)
        print 'los anchos sin outliers son:'
        print ancho_sin_outliers
        print 'la media es:'
        print media
        dibujar_lineas_guardar(lines,imagen_original,str(x)+
 ↪  'Hough.jpg')
```

11. Bibliografía

Referencias

[Asim, 2015] Asim, F. (2015). Androsensor. [Imagen]. Obtenido de: https://play.google.com/store/apps/details?id=com.fivasim.androsensor&hl=es.

[ByteLight, 2015] ByteLight (2015). Indoor positioning. [Imagen].Obtenido de: http://www.acuitybrands.com/solutions/services/bytelight-services-indoor-positioning.

[Cole, 2014] Cole, J. (2014). Shutter effect. [Imagen]. Obtenido de: http://petapixel.com/assets/uploads/2014/10/3192314056_e0df39ed3c_z.jpg.

[Docs.opencv.org, 2016a] Docs.opencv.org (2016a). Miscellaneous image transformations. [Online]. Obtenido de: http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html [Acceso 1 de junio de 2016].

[Docs.opencv.org, 2016b] Docs.opencv.org (2016b). Opencv: Image thresholding. [Online]. Obtenido de: http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html [Acceso 1 de junio de 2016].

[DuckMa, 2015] DuckMa (2015). ibeacon demo droidcon2015. [Imagen]. Obtenido de: https://play.google.com/store/apps/details?id=com.duckma.conference.

[Fisher, 2004a] Fisher, R. (2004a). Gaussian smoothing. http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

[Fisher, 2004b] Fisher, R. (2004b). Mean filter. http://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm.

[Fisher, 2016] Fisher, R. (2016). Gauss2. [Imagen]. Obtenido de: http://homepages.inf.ed.ac.uk/rbf/HIPR2/figs/gauss2.gif.

[IndooAtlas.com, 2016] IndooAtlas.com (2016). Indooratlas. [Online]. Obtenido de: https://www.indooratlas.com/ [Acceso 9 de junio de 2016].

[IndoorNavigation, 2015] IndoorNavigation (2015). Vlc indoor positioning. [Imagen]. Obtenido de: http://www.indoornavigation.de/perch/resources/vlc-indoor-positioning.jpg.

[Kuo et al., 2014] Kuo, Y.-S., Pannuto, P., Hsiao, K.-J., and Dutta, P. (2014). Luxapose: Indoor positioning with mobile phones and visible light. Technical report, University of Michigan.

[Luo et al., 2014] Luo, P., Ghassemlooy, Z., Minh, H. L., Khalighi, A., Zhang, X., and Yu, C. (2014). Experimental demonstration of an indoor visible light commu-

nication positioning system using dual-tone multi-frequency technique. Technical report, Northumbria University, Ecole Central Marseille, Beijing University of Posts & Telecommunication, National University of Singapore.

[Mautz, 2012] Mautz, R. (2012). Indoor positioning technologies. Technical report, ETH Zurich.

[Pinard, 2011] Pinard, P. T. (2011). Hough transform. [Imagen]. Obtenido de: http://ebsd-image.org/documentation/reference/ops/hough/op/houghtransform.html.

[Ravi et al., 2006] Ravi, N., Shankar, P., and Frankel, A. (2006). Indoor localization using camera phones. Technical report, Rutgers University.

[RevistaCompetitividad, 2016] RevistaCompetitividad (2016). Nuevas tendencias. [Imagen]. Obtenido de: http://revistacompetitividad.com/wp-content/uploads/2016/01/nuevas-tendencias1.png.

[Smith, 2016] Smith, C. (2016). Google lenovo project tango hands on 8. [Imagen]. Obtenido de: http://cdn.bgr.com/2016/02/mwc-2016-google-lenovo-project-tango-hands-on-8.jpg?w=624.

[Stenius and Yoo, 2016] Stenius, P. and Yoo, S. (2016). Principle of magnetic positioning technology. [Imagen]. Obtenido de: http://www.lgcnsblog.com/wp-content/uploads/2016/03/Principle-of-magnetic-positioning-technology.png.

[Tistory, 2010] Tistory (2010). Wifi positioning system. [Imagen]. Obtenido de: http://cfile24.uf.tistory.com/image/183991024BB215FF5BB389.

[Wikipedia, 2016] Wikipedia (2016). Canny edge detection. [Online]. https://en.wikipedia.org/wiki/Canny_edge_detector [Acceso 17 de julio de 2016].