UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

Facultad de Ingeniería Eléctrica

División de Estudios de Posgrado

IMPLEMENTACIÓN DE UN MAPA DE PROFUNDIDAD A PARTIR DE ENFOCADOS MÚLTIPLES PARA ROBÓTICA MÓVIL

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Opción en Sistemas Computacionales

Presenta

Roberto Rangel Heras

Doctor en Ciencias Computacionales Leonardo Romero Muñoz

Director de Tesis

Morelia Michoacán, Agosto de 2015

IMPLEMENTACIÓN DE UN MAPA DE PROFUNDIDAD A PARTIR DE ENFOCADOS MÚLTIPLES PARA ROBÓTICA MÓVIL.

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Roberto Rangel Heras

Dr. Leonardo Romero Muñoz

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Agosto 2015

A mi esposa Claudia Paz Servin.

Resumen

En muchas empresas las personas tienen que viajar muy a menudo, lo que hace que la empresa tenga que invertir dinero y tiempo en esos viajes. Otras empresas acuden a las videollamadas o videoconferencias, ahorrando el tiempo y dinero del viaje. Las desventajas de las videollamadas y videoconferencias es que las personas tienen que estar estáticas en un lugar y sólo observan una parte del ambiente donde se realiza la misma. La telepresencia sin embargo, le permite a una persona estar en un lugar sin que físicamente se encuentre ahí, es decir, por medio de un robot que se mueva en el ambiente puede ver y escuchar lo que rodea al robot (si el robot esta equipado con cámara y micrófono). Otra aplicación para la telepresencia puede ser en el ámbito de la vigilancia, donde un sólo usuario puede controlar varios robots y moverlos en el ambiente en busca de anomalías. Además, los robots pueden equiparse con herramientas que les permitan manipular el ambiente, permitiendo al operador realizar tareas a distancia. Se presenta el proyecto de un robot de telepresencia ensamblado con hardware de fácil acceso a los usuarios y software libre. Este robot está conformado por varios módulos de software que se encargan de tareas muy específicas, entre ellas, la obtención de imágenes, el movimiento del robot, etc. En el área de obtención de imágenes, se propone una nueva alternativa para obtener imágenes nítidas a partir de la fusión de varias tomas de un escena variando el paso de enfoque. Una imagen nítida es aquella donde todos los objetos se encuentran enfocados. Esta nueva propuesta se centra en obtener un mapa de decisión, este mapa nos ayuda a decidir qué píxel de una imagen debe ir en la imagen nítida, además al ir variando el paso de enfoque, de ante mano podemos conocer a qué distancia se ven nítidas las imágenes en cada paso de enfoque, esto nos puede ayudar a transformar un mapa de decisión en un mapa de profundidad, dicha información se puede utilizar en el robot para evitar colisiones en el ambiente, evitando así daños al robot. El robot podrá ser operado desde cualquier navegador web, sin necesidad de una aplicación externa, permitiendo así el acceso desde cualquier parte del mundo, siempre y cuando el usuario que quiera acceder al robot cuente con una conexión a internet. También esto permitirá al usuario decidir desde que dispositivo quiere acceder al robot, ya sea, una laptop, computadora de escritorio, tablet o smartphone, por mencionar algunos, permitiendo así el fácil acceso a los usuarios y sin poner limitaciones de software o de hardware.

Palabras clave: Robot, profundidad, enfoque, fusión, filtros.

Abstract

In many companies people have to travel very often, which makes the company has to invest money and time on these trips. Other companies turn to video calls or video conferencing, saving time and travel money. The disadvantages of video calling and video conferencing is that people have to be static in one place and just observe a part of the environment where it is carried out. Telepresence however, allows a person to be in a place without physically be there, that is, by a robot that moves in the environment can see and hear what's around the robot (if the robot is equipped camera and microphone). Another application for telepresence can be in the area of surveillance, where a single user can control multiple robots and move in the environment for abnormalities. In addition, the robots can be equipped with tools to manipulate the environment, allowing the operator to perform tasks remotely. This document describes the design of a telepresence robot hardware assembly easily accessible to users and free software is presented. This robot consists of several software modules that handle very specific tasks, including imaging, the robot motion, etc. In the area of imaging, a new alternative for sharp images from the merger of several takes of a scene changing step focusing is proposed. A clear picture is one where all objects are focused. This new proposal focuses on obtaining a decision map, this map helps us decide which pixel of an image must be in the sharp image also to be changing step focusing, we know beforehand how far are sharp images in each step focusing, this can help us to transform a decision map into a depth map, this information can be used in robot to avoid collisions in the environment, avoiding damage to the robot. The robot can be operated from any Web browser, without the need for external application as well allowing access from anywhere in the world, as long as the user who wants to access the robot counts with a connection to the Internet. This will also allow the user to decide which device you want to access the robot, either a laptop, desktop, tablet or Smartphone, to name a few, allowing easy access to users without limitations put software or hardware.

Contenido

Lista de Figuras

Lista de Tablas

Lista de algoritmos

Lista de Símbolos

ROW	Número total de renglones de una imagen.
COL	Número total de columnas de una imagen.
$I[]$	Arreglo de imágenes desde $i = 1, 2, 3, ..., M$.
n	Número total del arreglo de imágenes.
I_i	Imagen i-ésima del arreglo $I(i)$.
$I_i(r, c)$	Valor del pixel en la posición (r, c) de I_i.
I_i^p	Respuesta al filtro pasa altas de la imagen I_i.
$\|I_i^p(r, c)\|$	Valor absóluto del pixel en la posición (r, c).
I_F	Imagen fusionada.
M_D	Mapa de decisión para formar I_F.
G	Kernel de una función Gaussiana.
G^{0^o}	Derivada direccional de G con respecto al eje x.
G^{90^o}	Derivada direccional de G con respecto al eje y.
G^θ	Derivada direccional de G con un ángulo θ.
θ^N	Lista de ángulos θ de tamaño N.
B	Kernel para realizar los filtros morfológicos.
U_{max}	Umbral máximo utilizado en el algoritmo propuesto.
U_{min}	Umbral mínimo utilizado en el algrotimo propuesto.

Capítulo 1

Introducción

La tecnología en robótica móvil y cámaras digitales ha ido creciendo muy rápidamente en estos últimos años. Antes no era tan sencillo hacerse de un robot móvil para realizar experimentos, las cámaras digitales tampoco eran fáciles de adquirir, además de que eran muy costosas. Hoy en día esta tecnología cada vez es más fácil de adquirir, debido a que los costos se redujeron drásticamente, poniendo al alcance de los investigadores cámaras de última generación y piezas para hacer robots móviles a la medida del problema.

1.1. Planteamiento del problema

Equipar robots con cámaras ha sido cada vez mas cotidiano en nuestro días, al inicio los robots sólo contaban con telémetros láser, sonares, etc. y actualmente se equipan con toda clase de sensores. Dichos sensores sirven para obtener información del ambiente, ya sea recolectar datos para la persona que los opera o para realizar mediciones para sí mismo, como por ejemplo, saber que tan lejos está una pared o si existe riesgo de colisionar con un objeto (para los robots que puedan moverse dentro del ambiente).

La mayoría de los robots obtienen información mediante sensores (también se le puede proporcionar manualmente por la persona que los opera), algunos no visuales (telémetros láser, sonares, etc.) y otros visuales (cámaras) como puede verse en [Dudek11] y [Siegwart04], de tal modo que lo que comenzó con lecturas de distancias (para el caso del láser de rango y los sonares), continúa con cámaras digitales que capturan imágenes a

resoluciones muy altas (ej. 1920x1080 píxeles). Existen varias formas de capturar imágenes, se puede utilizar una sola cámara digital (sistema monocular), cámaras omnidireccionales, y varias cámaras.

Las imágenes tomadas por las cámaras abordo del robot pueden brindarnos mucha información, por ejemplo, obtener texturas, bordes, esquinas, etc. En este documento se centrará en la obtención de un mapa de profundidad (MDP) debido a que es de gran utilidad, pues nos indica que tan lejanos se encuentran los objetos de las cámaras en el mundo real a partir de imágenes.

Para obtener el MDP se diseño una propuesta novedosa a partir de varias imágenes de la misma escena variando el enfoque en la cámara.

1.2. Antecedentes en mapas de profundidad

Obtener una imagen nítida a partir de varias tomas de la misma escena variando el enfoque es una tarea que todavía no tiene una solución cerrada. Existen muchos algoritmos para llevar a cabo esta tarea como se verá en el Capítulo 2.

En teoría, un objeto esta enfocado en una imagen si se ve nítido y una forma de medir el nivel de nitidez de una imagen es aplicando un filtro pasa altas, donde un valor alto de la respuesta a este filtro nos indican que es nítido y un valor bajo nos indica que esta emborronado. Debido a esto, podemos aprovechar la información de varias imágenes cambiando el enfoque para fusionarlas y obtener una sola imagen donde todo se encuentra nítido.

Cuando utilizamos algoritmos para fusionar imágenes, además de obtener una imagen nítida donde todos los objetos se ven bien, también podemos obtener un mapa de decisión (MDD). Un MDD contiene la información acerca de qué píxel pertenece a qué imagen. Por ejemplo si tuviéramos 3 imágenes con diferente enfoque, el MDD estaría lleno de números del 1 al 3, indicando en la posición de un píxel a qué imagen pertenece en la imagen fusionada, siempre y cuando el algoritmo no permita incertidumbre.

La mayoría de los algoritmos que resuelven este problema, se centran únicamente en obtener el MDD y la imagen fusionada, dejando de lado el problema de registro que

tienen las imágenes al variar el enfoque.

Este documento se centrará en la obtención de un MDD para mapearlo a un MDP además de aplicar una heurística para resolver el problema del registro en las imágenes que se adquieran.

1.3. Justificación

Este problema se abordo debido a las numerosas aplicaciones que puede tener en robótica móvil. En especial se decidió trabajar en la obtención de un MDP porque se desarrolló un robot de telepresencia (Paynal) de el cual se hablará en el Apéndice B. Un MDP es de gran utilidad para cualquier robot móvil, debido a que ofrece información acerca de los objetos que tiene en frente. Los telémetros láser también ofrecen información acerca de la distancia a la que se encuentran los objetos con la desventaja de que sólo ofrecen un plano y el MDP ofrece información acerca de toda la escena que pueda capturar la cámara.

1.4. Objetivos de la tesis

Este documento se centra en la obtención de el MDP para brindarle información acerca del ambiente a Paynal.

1.4.1. Objetivo general

Realizar un algoritmo que calcule el MDP sin duda es el objetivo general de esta investigación, sin embargo, se describen los objetivos particulares en la siguiente subsección.

1.4.2. Objetivos particulares

- Obtener el MDP a partir de las imágenes que se capturen con la cámara variando el enfoque.
- Obtener el MDD utilizando la propuesta descrita en este documento.
- Obtener la imagen nítida a partir de las imágenes que se capturen con la cámara variando el enfoque.

- Comparar el MDD y la imagen nítida obtenidas con algunos de los enfoques descritos en el Capítulo 2.

1.5. Descripción de capítulos

La tesis se encuentra organizada de la siguiente manera:

El segundo capítulo presenta el estado del arte, en la obtención de el MDP mediante varias imágenes de la misma escena variando el enfoque. En las capturas de las imágenes cada enfoque nos ayuda a determinar la profundidad del objeto, utilizando información ya conocida sobre qué objeto se ve nítido en qué enfoque.

El tercer capítulo contiene la propuesta implementada para obtener el MDP, se muestra una propuesta novedosa. Se utilizan filtros pasa altas para procesar las imágenes y una heurística de histéresis para construir el MDP.

El cuarto capítulo se trata de los experimentos y resultados que se obtuvieron en la evaluación del los algoritmos contra los que se comparó la propuesta, así como los resultados obtenidos por la propuesta presentada.

En el quinto capítulo, se presentan las conclusiones y trabajos futuros que ofrece este tema de tesis.

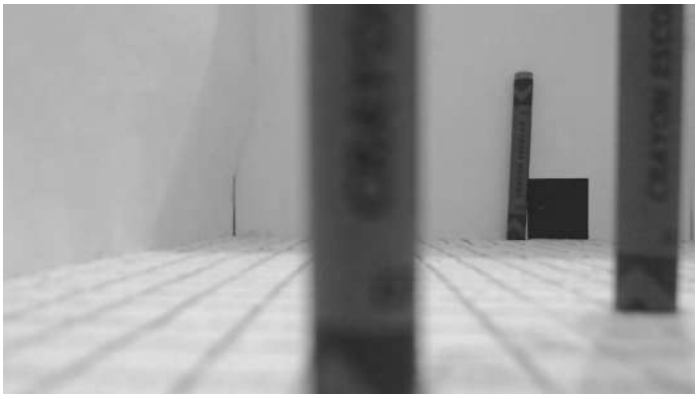
Capítulo 2

Trabajos previos de obtención de el mapa de profundidad en base a enfocado

La fusión de imágenes por enfocado se centra en obtener un MDD y una imagen nítida. Los objetos se enfocan o desenfocan en una imagen debido a que entran o salen del plano de enfoque. Un MDD nos indica qué a imagen pertenece cada píxel para conformar una imagen nítida y una imagen nítida es aquella donde todos los objetos se encuentran enfocados.

Se realizó una revisión bibliográfica para conocer los métodos para obtener un MDP basado en enfocado y el problema relacionado de fusión de imágenes multifoco. La Figura 2.1 muestra un ejemplo de 3 fotografías tomadas con diferente enfoque, donde se pueden ver 3 crayolas y un cuadro negro. En la Figura 2.1a se observa la crayola del fondo y el cuadro negro enfocados, mientras que la parte de enfrente se encuentra borrosa, en la Figura 2.1b, se puede observar que la crayola de en medio se ve mucho más clara que en la fotografía anterior y por último en la Figura 2.1c, se puede apreciar que la crayola de en frente se encuentra bien enfocada.

La Figura 2.2b muestra la fusión de las 3 imágenes anteriores y el MDD utilizado se muestra en la Figura 2.2a donde la zona de color blanco, indica que esa área pertenece a



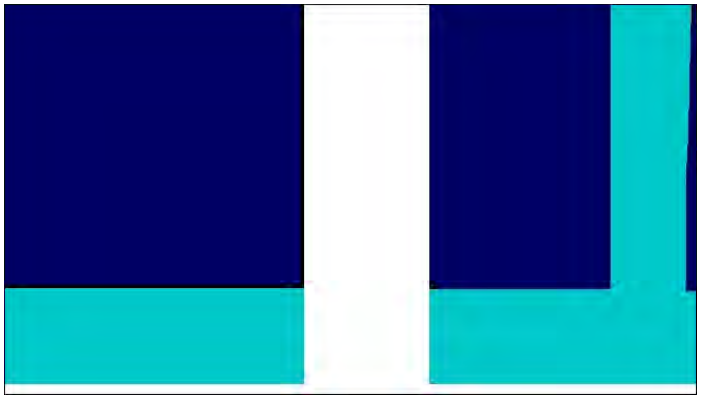
(a) enfoque 1.

(b) enfoque 7.

(c) enfoque 20.

Figura 2.1: Crayolas a diferente enfoque.

la imagen con enfoque 20, la zona azul claro indica que esa área pertenece a la imagen con enfoque 7 y la zona azul oscuro indica que esa área pertenece a la imagen con enfoque 1.



(a) Mapa de decisión sintético.

(b) Imagen nítida sintética.

Figura 2.2: Mapa de decisión e imagen nítida de las crayolas.

Esta fusión se realizó de manera manual decidiendo qué zonas pertenecían a que imagen.

En [Zhong99] se comparan la mayoría de los enfoques que existen para fusionar imágenes y obtener un MDD en base a la descomposición multiescala utilizando la transformada wavelet discreta, la transformación mediante la pirámide laplaciana, entre otros.

En [Asada97] se proponen dos algoritmos para obtener bordes y profundidad a partir de varias imágenes variando el enfoque. A diferencia de otros algoritmos que dependen del tamaño de la ventana local para mejorar la precisión a la hora de medir el emborronamiento de una zona, este algoritmo no necesita ventanas locales, en lugar de esto la intensidad de un píxel se mide a partir del cambio entre las diferentes imágenes multifoco. Adicionalmente el resultado de los análisis es estable debido a que se utiliza un modelo de

ajuste por mínimos cuadrados. Se presentan dos algoritmos: detección de bordes usando un acumulado de las imágenes fuera de foco las cuales representan la distribución espacial del emborronamiento y una estimación de profundidad usando una imagen focal.espacial la cual representa la distribución de intensidad a lo largo del eje del foco. Los resultados de sus experimentos muestran una fluctuación de posición de 0.5 píxeles en la detección de bordes y 0.2 % de error a 2.4 m en la detección de profundidad.

En [Li13] los autores abordan el problema de fusionar imágenes para obtener una imagen donde todo se encuentre enfocado, esta información se puede aprovechar para obtener el MDD y a su vez un MDP. Los autores aplican filtros morfológicos (TOP-HAT [Gonzalez07]) para obtener un tri-mapa de decisión y a partir de este mapa utilizan el algoritmo propuesto en [Wang07] para el empatamiento de la imagen y reducir el tri-mapa de desición a un mapa de decisión.

El desempeño del método propuesto en este documento se comparó con los dos métodos siguientes, los cuales utilizan filtros para fusionar las imágenes multifoco. En [Orozco13] se realiza la fusión de imágenes usando como base un filtro pasa altas, donde se demostró que los mejores resultados se obtienen al utilizar la Ecuación 2.1. Dicho algoritmo será descrito en la siguiente sección.

2.1. Fusión de imágenes usando un filtro pasa altas (FPA)

Dadas dos imágenes I_1 e I_2 con diferentes pasos de enfoques, se puede realizar la fusión de la siguiente manera:

1. Primero se aplican filtros pasa altas a ambas imágenes y se guarda la respuesta en I_1^p e I_2^p. Esto se realiza para obtener las altas frecuencias de las imagenes, debido a que estas nos indican las zonas nítidas.

2. Después se recorren ambas imágenes filtradas píxel a píxel en busca del píxel con mayor valor ($I_1^p(i,j)$ ó $I_2^p(i,j)$).

3. Una vez que tengamos el píxel que tenga el mayor valor, se toma de referencia para hacer la imagen fusionada, es decir, si $I_1^p(i,j) > I_2^p(i,j)$ se toma el píxel de la imagen

$I_1(i,j)$, sino se toma el valor de la imagen $I_2(i,j)$.

Algoritmo 1 FPA(I_1, I_2)

1: $I_1^p = filtroPasaAltas(I_1)$
2: $I_2^p = filtroPasaAltas(I_2)$
3: **para** $i = 1$ hasta $i = ROW$ **hacer**
4: **para** $j = 1$ hasta $j = COL$ **hacer**
5: **si** $|I_1^p(i,j)| > |I_2^p(i,j)|$ **entonces**
6: $I_F(i,j) = I_1(i,j)$
7: $M_D(i,j) = 1$
8: **si no**
9: $I_F(i,j) = I_2(i,j)$
10: $M_D(i,j) = 0$
11: **fin si**
12: **fin para**
13: **fin para**
14: REGRESA I_F, M_D

El Algoritmo 1, describe el proceso para obtener una imagen fusionada I_F y M_D a partir de dos imágenes con diferente enfocado. En las líneas 1 y 2 se utiliza la función $filtroPasaAltas$, que recibe una imagen y regresa la respuesta a un filtro pasa altas. También se obtiene un mapa de decisión M_D, en las líneas 7 y 10 se guarda píxel a píxel 1 ó 0, que nos indica a que imagen pertenece cada píxel, en nuestro caso 1 para I_1 y 0 para I_2. Este mapa de decisión se puede mapear a un MDP si se tiene información sobre los pasos de enfoque de la cámara.

La función de $filtroPasaAltas$ que se utilizó en este algoritmo se describe en la Ecuación 2.1. Donde I_i es una imagen de entrada e I_i^p es la imagen de salida. En la ecuación, se puede observar que se le resta a I_i la convolución de I_i con una función gaussiana y esto produce la salida. Se aplica el valor absoluto a la resta debido a que los cambios pueden ser positivos o negativos y al aplicar el valor absoluto ya no se discriminan valores. Para la función gaussiana se utilizó una media de 0.7, mientras que la varianza se calcula

(automáticamente en OpenCV) en base a la ventana de 3x3 que se utiliza.

$$I_i^p = |I_i - I_i * G_1| \tag{2.1}$$

El algoritmo FPA es la base de nuestro método desarrollado, ya que nosotros contamos con información acerca de cada enfoque y podemos mapear un MDD a un MDP fácilmente. Esto se describirá a detalle en el Capítulo 3. El siguiente algoritmo que se comparó con la técnica propuesta utiliza filtros orientados y se describe a continuación.

2.2. Fusión de imágenes propuesto por Rashid Minhas (SFF)

En [Minhas09] la idea principal es utilizar N filtros orientados a diferentes ángulos θ y posteriormente aplicarlos cada uno a cada imagen de entrada, la información de las respuestas a dichos filtros, nos ayuda a obtener un mapa de decisión.

Los filtros orientados se usan para examinar la respuesta al filtro en función del tiempo y fase, en muchas aplicaciones industriales. También son usados en numerosas aplicaciones de visión computacional y procesamiento de imágenes, como son:

1. Compresión de imágenes.
2. Segmentación.
3. Detección de bordes.
4. Texturas y análisis de movimiento.
5. Mejoramiento de imágenes.

La meta principal al encontrar la respuesta de un filtro a diferentes orientaciones, es analizar la salida en varios ángulos.

Para simplificar los cálculos, considere una Gaussiana G en coordenadas cartesianas x y y con media igual a 1 y varianza igual a 1 como la descrita en la Ecuación 2.2.

$$G = e^{-(x^2+y^2)} \tag{2.2}$$

La primera derivada de la Ecuación 2.2 con respecto a x es representada por la Ecuación 2.3 y la misma función derivada respecto a y se muestra en la Ecuación 2.4.

$$G^{0^o} = -2xe^{-(x^2+y^2)}dx \tag{2.3}$$

$$G^{90^o} = -2ye^{-(x^2+y^2)}dy \tag{2.4}$$

La Ecuación 2.3 representa una derivada a 0 grados, mientras que la Ecuación 2.4 representa una derivada a 90 grados. De tal manera que si queremos un filtro G_1 con una orientación θ, lo hacemos con una combinación lineal de 2.3 y 2.4, con lo cual obtendríamos la siguiente ecuación:

$$G^{\theta} = cos(\theta)G^{0^o} + sen(\theta)G^{90^o} \tag{2.5}$$

La Ecuación 2.5 es la derivada direccional de la Ecuación 2.2 con respecto al ángulo θ.

En el Algoritmo 2 se muestra el proceso a seguir para obtener el mapa de decisión y la imagen reconstruida, a partir de los filtros orientados y las imágenes utilizando diferente foco. Las entradas son:

- $I(i)$ $i = 1, .., M$, las M imágenes con diferente enfoque.
- La lista de θ^N son N grados diferentes (0^o, 45^o, 90^o, 135^o, 180^o, 235^o, 270^o y 315^o).

En la línea 2 se obtienen los filtros en base a la lista de ángulos que se le manden como parámetro. En la línea 6 se obtienen las respuestas a los N filtros de cada imagen, posteriormente se queda con el píxel con valor más alto (línea 10) y se guarda en *PI*, posteriormente se aplica una ventana de tamaño *Size*, para sumar la información de los vecinos (línea 17). Esta suma se realiza para ponderar los píxeles por ventanas de tamaño *Size* y así considerar a la vecindad en el siguiente paso. *FI* se obtiene mediante el argumento máximo de *PI* píxel a píxel (línea 24). Por último, el MDD es obtenido al aplicar un filtro

Algoritmo 2 SFF($I[]$, θ^N)

para $o = 1$ hasta $o = N$ **hacer**
  $G^{\theta}[o] = cos(\theta[o])G^{0^o} + sen(\theta[o])G^{90^o}$
fin para
para $i = 1$ hasta $i = M$ **hacer**
  para $o = 1$ hasta $o = N$ **hacer**
    $auxI[o] = I[i] * G^{\theta}[o]$
  fin para
  para $r = 1$ hasta $r = ROW$ **hacer**
    para $c = 1$ hasta $c = COL$ **hacer**
      $PI[i](r, c) = \text{máx}(auxI[](r, c))$
    fin para
  fin para
fin para
para $i = 1$ hasta $i = M$ **hacer**
  para $r = 1$ hasta $r = ROW$ **hacer**
    para $c = 1$ hasta $c = COL$ **hacer**
      $PI[i](r, c) = \sum_{j=r-Size/2}^{r+Size/2} \sum_{k=c-Size/2}^{c+Size/2} PI[i](j, k)$
    fin para
  fin para
fin para
para $i = 1$ hasta $i = M$ **hacer**
  para $r = 1$ hasta $r = ROW$ **hacer**
    para $c = 1$ hasta $c = COL$ **hacer**
      $FI(r, c) = \text{máx}_{argi}(PI[](r, c))$
    fin para
  fin para
fin para
$M_D = filtroMediana(FI)$
$I_F = \text{ReconstruccionDeImagen}(M_D)$
REGRESA I_F, M_D

de mediana a *FI* en la línea 28 para eliminar el posible ruido de sal y pimienta que haya quedado en la imagen. La reconstrucción de la imagen simplemente es componer la imagen fusionada a partir de el MDD.

Además se analizó un tercer algoritmo que utiliza filtros, pero a diferencia de los dos anteriores que utilizan filtros pasa altas, este algoritmo utiliza filtros morfológicos. Cabe destacar que no se comparó este algoritmo con la propuesta debido a que no se llego a implementar en su totalidad.

2.3. Fusión de imágenes propuesto por Shutao Li

En [Li13] la idea principal es fusionar imágenes para obtener una imagen donde todo se encuentre enfocado, esta información se puede aprovechar para obtener el mapa de decisión. Se puede observar el procedimiento que propone el autor en el Algoritmo 3. Este algoritmo recibe una arreglo de imágenes *I[]* y regresa una imagen fusionada I_F. Analizando el algoritmo propuesto por los autores paso a paso, se puede ver que en la línea 3 se aplican filtros morfológicos (TOP-HAT [Gonzalez07]) para obtener el píxel enfocado, donde $I[i] \circ B$ denota una operación de *opening* $((I[i] \ominus B) \oplus B)$ que significa aplicar una erosión a la imagen y al resultado aplicarle una dilatación, por otro lado, $I[i] \bullet B$ denota una operación de *closing* $((I[i] \oplus B) \ominus B)$ que significa aplicar una dilatación a la imagen y al resultado aplicarle una erosión. En la línea 7 simplemente tomamos el valor máximo por píxel de los filtros anteriores. Posteriormente se determina si un píxel está enfocado o no (línea 14), además, existen píxeles con valores muy grandes que definitivamente están enfocados, estos se determinan en la línea 15, en base a un umbral *H*.

Este último algoritmo solamente se analizó hasta la línea 34 debido a que en [Wang07] el algoritmo para el empatamiento de la imagen y reducir el tri-mapa de desición a un mapa de decisión no fue del todo claro. En el siguiente capítulo, se verán algunos experimentos preliminares para comparar los algoritmos FPA, SFF y la propuesta.

Algoritmo 3 AFSL($I[]$)

Entrada: $I(i)(i = 1, .., M)$

Salida: I_F, M_D

para $i = 1$ hasta $i = M$ **hacer**
  $d_b[i] = I[i] - I[i] \circ B$
  $d_d[i] = I[i] \bullet B - I[i]$
  para $r = 1$ hasta $r = ROW$ **hacer**
    para $c = 1$ hasta $c = COL$ **hacer**
      $D[i](r, c) = \text{máx}(d_b[i](r, c), d_d[i](r, c))$
    fin para
  fin para
fin para
para $i = 1$ hasta $i = M$ **hacer**
  para $r = 1$ hasta $r = ROW$ **hacer**
    para $c = 1$ hasta $c = COL$ **hacer**

$$R[i](r, c) = \begin{cases} 1, & \text{if } D[i](r, c) > \underset{m=1:N, m \neq n}{\text{máx}}(D[m](r, c)) \\ 0, & \text{otherwise} \end{cases}$$

$$P^d[i](r, c) = \begin{cases} 1, & \text{if } D[i](r, c) - \underset{m=1:N, m \neq n}{\text{máx}}(D[m](r, c)) > H \\ 0, & \text{otherwise} \end{cases}$$

    fin para
  fin para
  $R^M[i] = median(R)$
  $R^{MS}[i] = median(skelet(R^M[i], totalIte))$
  para $r = 1$ hasta $r = ROW$ **hacer**
    para $c = 1$ hasta $c = COL$ **hacer**

$$R^d[i](r, c) = \begin{cases} 1, & \text{if } R^{MS}[i](r, c) = 1\text{o } P^d[i](r, c) = 1 \\ 0, & \text{otherwise} \end{cases}$$

    fin para
  fin para
fin para

25: **para** $i = 1$ hasta $i = M$ **hacer**

26: **para** $r = 1$ hasta $r = ROW$ **hacer**

27: **para** $c = 1$ hasta $c = COL$ **hacer**

28: $$T[i](r,c) = \begin{cases} 1, & \text{if } R^d[i](r,c) = 1 \text{y} \underset{m=1:N, m\neq n}{\text{máx}}(R^d[m](r,c)) = 0 \\ 0, & \text{if } R^d[i](r,c) = 0 \text{y} \underset{m=1:N, m\neq n}{\text{máx}}(R^d[m](r,c)) = 1 \\ 0.5, & \text{otherwise} \end{cases}$$

29: **fin para**

30: **fin para**

31: **fin para**

32: **para** $i = 1$ hasta $i = M$ **hacer**

33: $\alpha[i] = ImageMatting(T[i], I[i])$

34: **fin para**

35: $I_N[1] = I[M]$

36: **para** $i = 2$ hasta $i = M - 1$ **hacer**

37: **para** $r = 1$ hasta $r = ROW$ **hacer**

38: **para** $c = 1$ hasta $c = COL$ **hacer**

39: $I_N[i](r,c) = \alpha[i](r,c) * I[i](r,c) + (1 - \alpha[i](r,c))I_N[i-1](r,c)$

40: **fin para**

41: **fin para**

42: **fin para**

43: $I_F = I_N[M-1]$

44: REGRESA I_F, M_D

2.4. Concluciones

En este capítulo se analizaron 3 algoritmos basados en filtros que se centran en la obtención de una imagen fusionada y un mapa de decisión. Como pudo verse en la Sección 2.1, la idea es bastante sencilla pero a la vez ofrece buenos resultados, además de que es fácil de implementar. La desventaja de FPA es que utiliza una decisión *Si/No* y esto lleva a que el MDD se vea muy disperso, similar al ruido de sal y pimienta. El siguiente algoritmo que se implementó fue el de la Sección 2.2, dicho algoritmo también es fácil de implementar y es un poco más robusto que FPA, la desventaja es que emplea más operaciones y el tiempo de cálculo se puede triplicar en algunos casos. Por último en la Sección 2.3 se describe un algoritmo que utiliza filtros mofológicos que tambien son fáciles de implementar y ofrecen rápidez, en el artículo muestran resultados muy prometedores, pero debido a la complejidad en el artículo [Wang07] no se pudo completar la implementación.

Por lo tanto las compararciones sólo las realizaremos contra FPA y SFF. En los Capítulos 3 y 4 se pueden ver los resultados obtenidos por los algoritmos antes planteados, así como una comparación de tiempo de ejecución entre cada uno de ellos.

Capítulo 3

Propuesta de determinación de el mapa de profundidad en base a enfocado

En el Capítulo 2 se abordaron algunos de los enfoques existentes para obtener un MDD. En este Capítulo se presentará una nueva alternativa para obtener un MDD mediante enfocado. Se utiliza una variación del algoritmo FPA, utilizando un esquema de histérisis como el que utiliza el algoritmo clásico de Canny para detección de líneas. En general este capítulo se divide de la siguiente manera:

1. Características de la cámara a utilizar.
2. Resolver el problema de registro en las imágenes al cambiar el enfoque.
3. Propuesta de determinación del MDP en base a enfocado.
4. Experimentos preliminares, artificiales y reales en un ambiente controlado.

3.1. Característica de la cámara Logitech C920

Las cámaras Logitech C920 cuentan con la capacidad de variar el enfoque desde 1 hasta 20, donde en el paso 1 se ven nítidos todos los objetos a mas de 30 c.m. y de esa

Figura 3.1: Imagen de la cámara Logitech C920

distancia hacia adelante se ven cada vez mas borrosos a medida que se acercan a la cámara. El enfoque 20 por otro lado nos ofrece una nítidez en los objetos que se encuentren hasta a 3 c.m. de la cámara, lo cual nos ofrece un rango de entre 3 c.m. y 30 c.m. para calcular el MDP en nuestro algoritmo.

El tipo de conexión de la cámara se realiza mediante la interfaz USB, utilizando el protocolo de comunicación USB 2.0. Cuenta con un micrófono estereo integrado. La captura de imagen y video se realiza bajo el aspecto 16:9, en las imágenes soporta resoluciones de 2.0 MP, 3 MP, 6 MP y 15 MP; mientras que en el video soporta resoluciones de 360p, 480p, 720p y hasta 1080p a un máximo de 30 frames por segundo.

La Figura 3.1 muestra como se ve la cámara físicamente, como puede observarse en la figura, la lente se encuentra en el centro y el par estéreo de micrófonos a los lados. Las características completas de la cámara pueden consultarse en [LOG15]. Aquí solamente se han destacado las mas importantes para la investigación realizada.

Una vez conocidas las características de la cámara se realizaron pruebas para conocer el desempeño a la hora de tomar las imágenes y se hicieron pruebas variando el enfoque. El primer detalle que se observó al variar el enfoque además de que se emborronaban los objetos, fue que los mismos parecían hacerse más grandes o más pequeños, lo que nos llevó a resolver el problema de registro en las imágenes.

Figura 3.2: Fotografía de la caja utilizada

3.2. Resolviendo el problema de registro de las imágenes

Antes de resolver el problema que se presento del registro de imágenes se observaron otros problemas, entre ellos la iluminación, el tamaño de los objetos debido al cambio de enfoque, el ruido interno de la cámara, etc. En esta sección se tratarán los problemas presentados así como las soluciones propuestas.

Después de ver los problemas, se decidió que para realizar las pruebas reales, era conveniente construir un ambiente controlado. Dicho ambiente comprendería una caja de cristal con un lado descubierto, a esta caja se le pegaron hojas blancas para evitar que entrará demasiada luz, pero permitiendo que la iluminación fuera adecuada, evitando que las luces de fondo afectaran las muestras.

La Figura 3.2 muestra la caja de cristal descrita anteriormente, se pueden ver que a los lados patrones de calibración de cámaras, del lado derecho un patron asimétrico de círculos y del lado izquierdo un tablero de ajedrez.

3.2.1. Ruido de la cámara

Se tomaron 100 imágenes sucesivas de una misma escena para observar que tanto ruido tenía la cámara al tomar una foto tras otra. A primera vista no se veía ruido, pero cuando se analizaron a detalle las tomas sucesivas se encontró ruido entre una toma y otra. Esto se puede deber a varios factores, entre ellos la iluminación del ambiente y el ruido inherente al sensor de imagen. Las pruebas se realizaron de día y sin luz artificial.

Como se puede observar en la Figura 3.3, se presentan las primeras dos tomas sucesivas un $\frac{1}{30}$ de segundo después la primera de la segunda. A simple vista la Figura 3.3a y 3.3b son idénticas, pero en la Figura 3.3c se puede ver el ruido inherente al cambio de la iluminación del ambiente. Ya que a simple vista no se nota, se ecualizó la imagen y el resultado se muestra en la Figura 3.3d, donde se aprecia mejor el ruido. La media de la diferencia de las 100 imágenes fue de 2.29592 y la desviación estándar de 0.841003, el valor absoluto de la máxima diferencia fue de 39.

El siguiente problema que se abodo es el problema de registro en las imágenes al variar el enfoque.

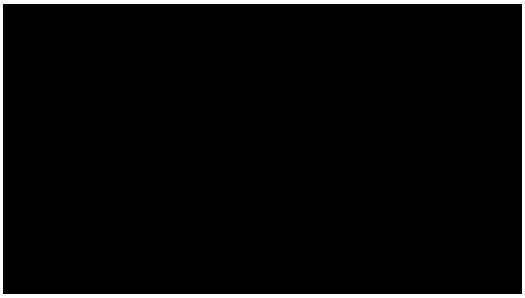
3.2.2. Problema de crecimiento de los objetos al variar el enfoque

El principal problema de variar el foco es que los objetos en la imagen se hacen más grandes o más chicos, además de emborronarse o aclararse, dependiendo de la variación del foco. Para calcular el mapa de decisión, se requiere que los objetos en la imagen conserven el mismo tamaño, es decir, si se aumenta el foco, sólo debería emborronarse, pero deberían conservar el mismo tamaño.

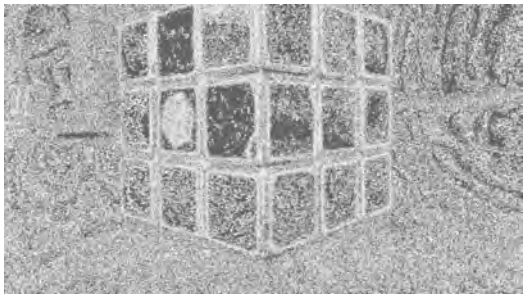
Para resolver este problema, se calcularon 19 matrices de transformación proyectiva, se utilizaron estas matrices en los pasos de enfoque del 1 al 19, debido a que el paso 20 es el que hace los objetos más grandes dentro de la imagen y es el que se usó de referencia. Para calcular una matriz de transformación homográfica, se requieren al menos 4 puntos en la imagen origen y 4 puntos en la imagen destino. La Figura 3.4 muestra un ejemplo de aplicar una matriz de este tipo. Donde el área contenida por las líneas rojas (Figura 3.4a), se transforma en la Figura 3.4b, donde se pueden ver los tomos de la fotografía como si se

(a) Imagen de un cubo de rubik

(b) Imagen del mismo cubo unas milésimas después



(c) Diferencia entre ambas imágenes.



(d) Diferencia entre ambas imágenes ecualizada.

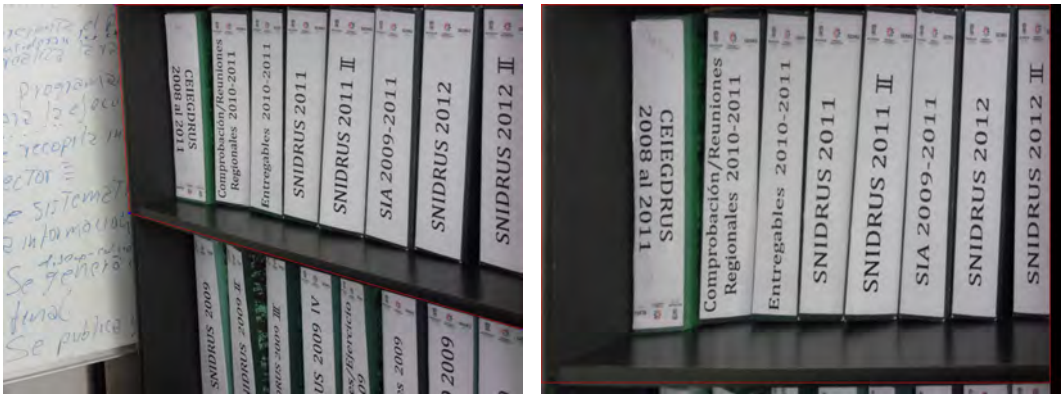
Figura 3.3: Prueba de tomas sucesivas

hubieran tomado de frente. De la misma manera se buscaron 4 puntos en una imagen con enfoque 20 para mapearlos a 4 puntos en una imagen con paso 19, 18,..., 1. Con el fin de obtener 19 matrices de transformación proyectiva. Para calcularlas se siguieron estos pasos:

1. Se tomaron 20 tomas controladas de una imagen conocida, 4 cuadros negros cerca de las esquinas con fondo blanco como se observa en la Figura 3.5.

2. Manualmente se pusieron los centros de 4 cuadros de la imagen y se utilizó una variación del algoritmo de K-medias para obtener el centro exacto de cada cuadro en la imagen.

3. Para cada cuadro, se ajustaron los centros a una línea recta con el método de mínimos cuadrados totales (MCT) [Romero14]. La Figura 3.6 muestra 4 líneas rectas de cada esquina hacía el centro óptico de la cámara. Estas líneas ya fueron ajustadas por



(a) Fotografía de unos tomos de lado.

(b) Fotografía al aplicar la transformación proyectiva

Figura 3.4: Ejemplo transformación proyectiva

Figura 3.5: Imagen de un patrón conocido

MCT y las "x"son los puntos obtenidos por el paso anterior. La Figura 3.7 muestra el acercamiento de cada cuadro, así como la línea correspondiente calculada por MCT.

4. Se utilizó la proyección ortogonal de cada centro de los cuadros hacía la línea obtenida y con estos valores se calcularon las matrices de transformación proyectiva.

Al utilizar las matrices de transformación proyectiva en los pasos de enfoque del 1 al 19, se pudo apreciar que los objetos de la imagen ya no crecían o decrecían, solamente se emborronaban.

La Figura 3.8 muestra dos tomas del patrón con diferente enfocado sin aplicar

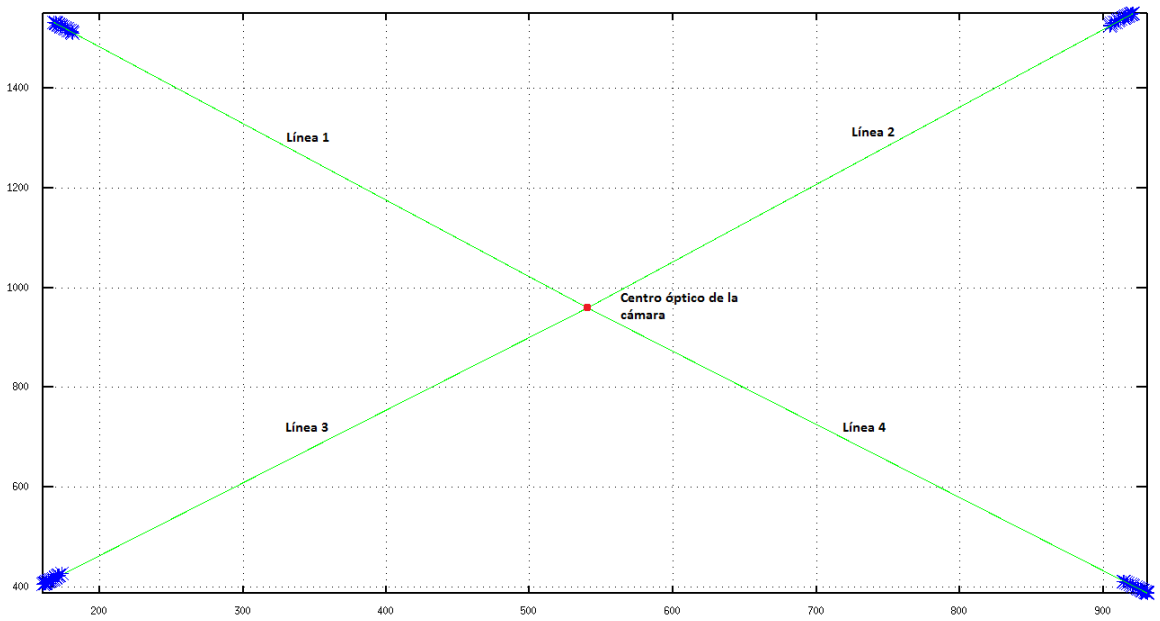
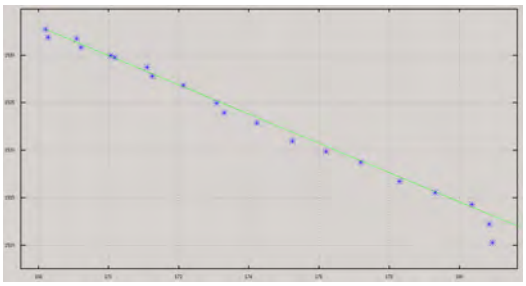


Figura 3.6: Comparación entre el centro óptico de la cámara y las líneas encontradas.

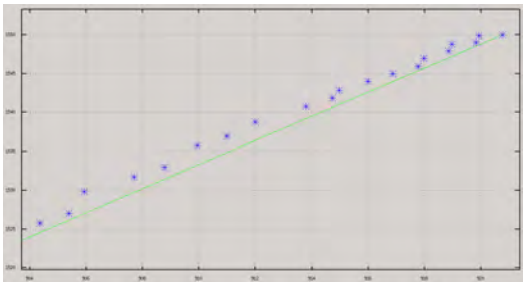
ninguna transformación proyectiva. En la Figura 3.8a se utilizó un enfoque 2, mientras que en la Figura 3.8b se utilizó un enfoque 19. La diferencia entre ambas imágenes se puede observar en la Figura 3.8c, en ella se nota claramente el corrimiento que sufre la imagen de un enfoque a otro.

La Figura 3.9 muestra las mismas tomas de la Figura 3.8, pero esta vez aplicando la matriz de proyección obtenida, se puede notar en la Figura 3.9b que conserva el mismo tamaño que la Figura 3.9a, mientras que en la Figura 3.9c se puede ver que la diferencia entre las mismas ya no tiene corrimiento y lo que se ve es la zona que se emborrona al cambiar el enfoque.

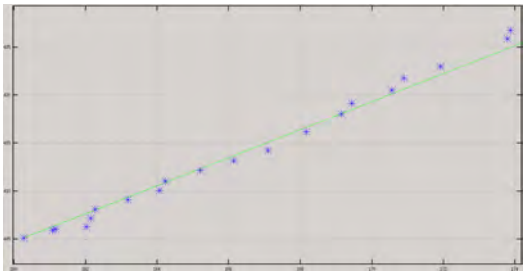
Una vez realizado esto, se procedió a implementar la propuesta de este documento de tesis.



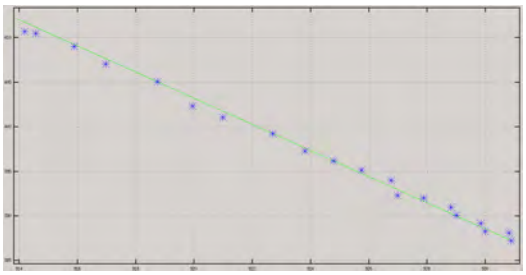
(a) Acercamiento de la Figura 3.6 en la esquina superior izquierda.



(b) Acercamiento de la Figura 3.6 en la esquina superior derecha.



(c) Acercamiento de la Figura 3.6 en la esquina inferior izquierda.



(d) Acercamiento de la Figura 3.6 en la esquina inferior derecha.

Figura 3.7: Acercamiento de cada esquina de las líneas obtenidas.

3.3. Implementación de la propuesta de determinación de el mapa de profundidad en base a enfocado

En una etapa preliminar se calibraron y rectificaron las cámaras como puede verse en el Apendíce A. Para obtener mejores resultados, se recortaron las imágenes quedándose solamente con las áreas válidas.

La propuesta se describe a grandes rasgos en los siguientes pasos:

1. Se toman imágenes de la misma escena variando el foco. Sean $I(i)$, $i = 1, .., 20$ imágenes con pasos de enfoque consecutivos. En la imagen $I(1)$ se ven nítidos los objetos lejanos, mientras que en $I(20)$ se ven nítidos los objetos muy cercanos a la cámara (3 c.m.). El tiempo que tarda la cámara en obtener las 20 imágenes a una resolución de 1920x1080 píxeles es de 45 a 60 segundos, dependiendo de los procesos que se encuentren ejecutando en la máquina en ese momento. Mientras que para una resolución

(a) Imagen de un patrón conocido con un enfoque 2

(b) Imagen de un patrón conocido con un enfoque 19

(c) Diferencia entre ambas imágenes

Figura 3.8: Tomas del patrón conocido antes de aplicar transformación proyectiva.

de 640x320 píxeles el tiempo que tarda es de 20 segundos. Cabe destacar que para disminuir el ruido inherente al sensor se acumulan 3 imágenes con un mismo paso de

(a) Imagen de un patrón conocido con un enfoque 2

(b) Imagen de un patrón conocido con un enfoque 19

(c) Diferencia entre ambas imágenes

Figura 3.9: Tomas del patrón conocido después de aplicar transformación proyectiva.

enfoque y se promedian.

2. Se toman las primeras dos imágenes y se obtienen las respuestas de un filtro pasa

altas de ambas. Se utiliza el filtro pasa altas descrito en el Capítulo 2.

3. Se calculan dos mapas de diferencias, para las dos respuestas obtenidas en el paso anterior. En esta etapa es donde se utiliza el esquema de histérisis.

4. Se realiza la fusión de las imágenes utilizando los mapas de diferencias de la etapa anterior y la respuesta se utiliza para realizar una nueva fusión con la tercer imagen. Se repiten los pasos del 2 al 4 hasta llegar a la imagen 20.

A continuación se describen a detalle cada uno de estos pasos.

3.3.1. Secuencia de imágenes con diferente enfoque

Para llevar a cabo esta tarea, se utilizó OpenCV, se hizo un programa para variar el foco por medio de una barra deslizante, para comprobar el funcionamiento del código. Posteriormente se quitó la barra para automatizar dicho proceso. Al variar el foco, se desechan las primeras tomas de la cámara, considerando el tiempo que tarda el motor en ajustarse cada paso. Utilizar imágenes sucesivas garantiza que el cambio será suave, es decir, se va emborronando lentamente de una imagen a otra.

El Algoritmo 4 muestra los pasos que se han de seguir para fusionar las 20 imágenes. Recibe de entrada un arreglo de imágenes *I[]* que contiene todas las imágenes que se tomaron. Después va fusionando de dos en dos imágenes, como se muestra en la línea 4. Esta función se ve a detalle en el algoritmo 5. Donde I_1, I_2, i son parámetros de entrada, mientras que I_F y M_D son parámetros de entrada/salida.

Algoritmo 4 FPAH(*I[]*)

1: $I_1 = I(1)$
2: **para** $i = 2$ hasta $i = 20$ **hacer**
3: $I_2 = I(i)$
4: FusionAux(I_1,I_2,i,I_F,M_D)
5: $I_1 = I_F$
6: **fin para**
7: REGRESA I_F, M_D

El Algoritmo 5 muestra los pasos a seguir para fusionar 2 imágenes por medio de las respuestas a filtros pasa altas y dos mapas de diferencias. Los parámetros que recibe esta función son dos imágenes I_1 e I_2, el índice i que nos indica la posición de la imagen que estamos fusionando y por último I_F y M_D que son la imagen fusionada y el MDD respectivamente, estos dos últimos son valores de entrada/salida, llevan el acumulado de la fusión hasta el índice i. Los filtros pasa altas, los mapas de diferencias y la fusión utilizando dichos mapas se describen más adelante.

Algoritmo 5 FusionAux(I_1, I_2, i, I_F, M_D)

1: $I_1^p = filtroPasaAltas(I_1)$
2: $I_2^p = filtroPasaAltas(I_2)$
3: $mapaDiff_1 = obtenMapaDiff(I_1^p, I_2^p)$
4: $mapaDiff_2 = obtenMapaDiff(I_2^p, I_1^p)$
5: $fusion(I_1, I_2, mapaDiff_1, mapaDiff_2, i, I_F, M_D)$

3.3.2. Filtros pasa altas

Una vez guardadas las 20 tomas, se procedió a aplicar un filtro pasa altas a cada par de imágenes de entrada como se describe en el Algoritmo 5 líneas 1 y 2. El filtro pasa altas que se aplicó se describe en la ecuación 3.1, donde I es la imagen de entrada e I^p es la salida del filtro pasa altas; G_1 es un kernel gaussiano utilizando $\sigma = 0.795$ (en x y y) y un tamaño de kernel de 5x5; y G_2 es un kernel gaussiano utilizando $\sigma = 3.91$ (en x y y) y un tamaño de kernel de 9x9. $I_i * G_1$ denotan una convolución entre la imagen y el kernel respectivamente. Los parámetros anteriores se fijaron a prueba y error, quedándose con los que se obtuvieron mejores resultados.

Si bien la primera parte de la Ecuación 3.1 es un filtro pasa altas ($|I_i - I_i * G_1|$), posteriormente se suaviza esta respuesta para ponderar a los vecinos cercanos de los valores de altas frecuencias. Las dos salidas de los filtros se guardaron para la siguiente etapa, que es obtener un mapa de diferencia.

$$I_i^p = (|I_i - I_i * G_1|) * G_2 \tag{3.1}$$

3.3.3. Mapas de diferencias

A partir de las salidas de los filtros pasa altas, los mapas de diferencias se obtienen con el Algoritmo 6.

En la línea 1, se escriben las variables que se utilizarán a lo largo del algoritmo. Primero se restan las salidas de los filtros $I_1^p - I_2^p$, esta resta se realiza píxel a píxel y esta salida la guardamos en una matriz llamada *diff*. Después de eso se recorren todos los píxeles de *diff* en busca de un valor que sea mayor o igual a *uMax* para aplicar un esquema de histéresis. Se pone en una pila el píxel que cumpla la condición y se examinan los píxeles de arriba, abajo, izquierda y derecha (si existen) y si el valor del píxel es mayor que *uMin* se agrega a la pila y se continua con el proceso mientras haya píxeles en la pila. Todos los píxeles que se agreguen a la pila se marcan como visitados, para no volver a visitarlos en el futuro. Además, se marca una zona por cada píxel que cumpla la condición del *uMax* y se guardan las posiciones de todos los píxeles que cumplan con la histéresis en *mask* con dicha zona. Además se guarda el valor máximo de cada zona en un arreglo de máximos (línea 33). Por último, para cada zona obtenida en el proceso de histéresis, se le asigna el valor máximo correspondiente (línea 42), con lo cual obtenemos la salida del algoritmo, el mapa de diferencia *mapaDiff*.

Las Figuras 3.10a, 3.10b y 3.10c muestran un ejemplo gráfico de como se lleva a cabo este procedimiento. En la Figura 3.10a podemos observar una matriz llena de números, el número 10 esta marcado con rojo, debido a que es el primer número que cumple con la condición del umbral máximo (en este ejemplo umbralMaximo = 5 y umbralMinimo = 2). En la Figuras 3.10b se observan todos los píxeles que cumplieron con la condición del umbral mínimo, además se guarda el valor máximo de la zona. Por último en la Figura 3.10c se llena toda la zona que cumplió con la condición con el valor máximo.

Una vez que tenemos los mapas de diferencias de las imágenes, podemos proceder obtener el mapa de decisión y fusionar la imagen.

Algoritmo 6 obtenMapaDiff(I_1^p, I_2^p)

1: $pila, maximos, maux, diff, mask, uMin, uMax, maximo, k$
2: Se inicializa *pila* vacia y *maximos* vacio. $k = 1$ (Se comienza por la zona 1).
3: *maux* y *mask* se inicializan en 0. (Matrices de ROWxCOL).
4: $umbralMaximo = 0.75$ y $umbralMinimo = 0.25$
5: $diff = I_1^p - I_2^p$
6: **para** $r = 0$ hasta $r < ROW$ **hacer**
7: **para** $c = 0$ hasta $c < COL$ **hacer**
8: **si** $maux(r, c) == 0$ **entonces**
9: **si** $diff(r, c) >= uMax$ **entonces**
10: $pila.push(r, c)$
11: $maximo = diff(r, c)$
12: $mask(r, c) = k$
13: **mientras** !$pila.vacia$ **hacer**
14: $[raux, caux] = pila.pop()$
15: **si** $maux(raux, caux) > 0$ **entonces**
16: $maux(raux, caux) = 255$
17: **para** $rr = -1$ hasta $rr <= 1$ **hacer**
18: **para** $cc = -1$ hasta $cc <= 1$ **hacer**
19: **si** $cc! = 0 \&\& cc! = 0$ **entonces**
20: $rraux = raux + rr, ccaux = caux + cc$
21: **si** $diff(rraux, ccaux) > uMin$ **entonces**
22: **si** $diff(rraux, ccaux) > maximo$ **entonces**
23: $maximo = diff(rraux, ccaux)$
24: **fin si**
25: $pila.push(rraux, ccaux)$
26: $mask(rraux, ccaux) = k$
27: **fin si**
28: **fin si**
29: **fin para**
30: **fin para**
31: **fin si**
32: **fin mientras**

33: $maximos[k] = maximo$
34: $k = k + 1$
35: **fin si**
36: **fin si**
37: **fin para**
38: **fin para**
39: **para** $r = 0$ hasta $r < N$ **hacer**
40: **para** $c = 0$ hasta $c < M$ **hacer**
41: **si** $mask(r, c)! = 0$ **entonces**
42: $mapaDiff(r, c) = maximos[mask(r, c) - 1]$
43: **fin si**
44: **fin para**
45: **fin para**
46: REGRESA $mapaDiff$

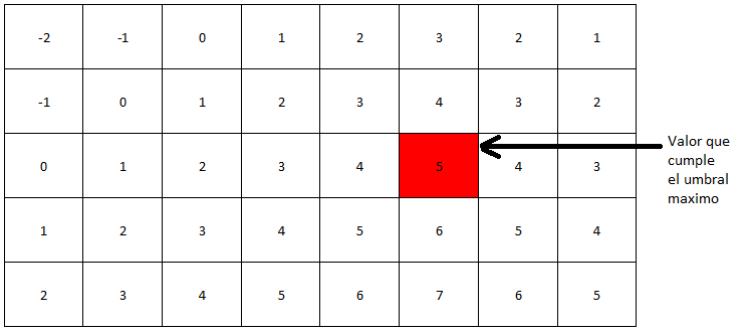
3.3.4. Obtención del mapa de decisión e imagen nítida

Una vez que tenemos los dos mapas de diferencias que se pueden ver en el Algoritmo 5 líneas 3 y 4, se fusionan las imágenes de entrada en base al Algoritmo 7.

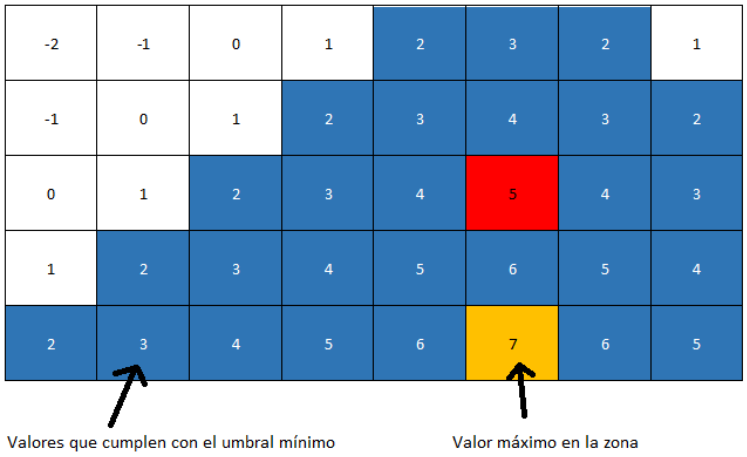
En el Algoritmo 7 se puede observar que se utilizan los mapas de diferencias para crear el mapa de decisión (M_D) y la imagen fusionada (I_F). En la línea 5 se puede ver que los valores de los mapas de diferencias que valgan 0 se descartan, esto porque en esas zonas no se obtuvo información. En las líneas 7 y 11 se llenan los valores del mapa de decisión, note que sólo cuando $iaux = 2$ (línea 10) se le asigna valor a M_D en la línea 11, esto debido a que cuando $iaux$ es mayor que 2, el mapa de decisión ya contiene información de las imágenes anteriores y debe conservarse esa información. En las líneas 8 y 13 se asignan los valores a I_F dependiendo del caso. La imagen fusionada queda en I_F y el mapa de decisión queda en M_D, debido a que son variables de entrada/salida.

Algoritmo 7 fusion($I_1, I_2, mapaDiff_1, mapaDiff_2, iaux, I_F, M_D$)

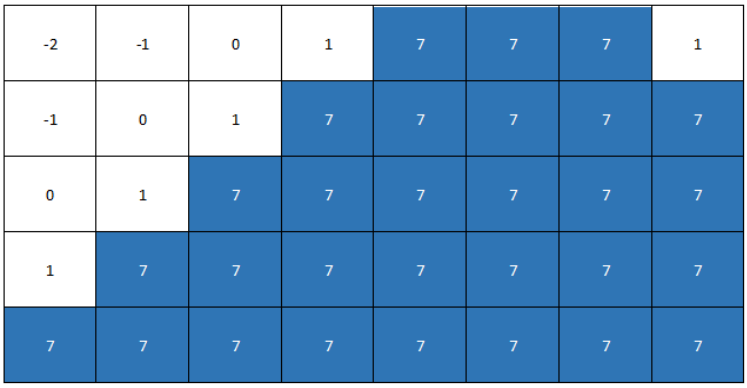
```
1: para i = 0 hasta i < ROW hacer
2:   para j = 0 hasta j < COL hacer
3:     diff_1 = mapaDiff_1(i, j)
4:     diff_2 = mapaDiff_2(i, j)
5:     si diff_1 != 0 && diff_2 != 0 entonces
6:       si diff_2 > diff_1 entonces
7:         M_D(i, j) = iaux
8:         I_F(i, j) = I_2(i, j)
9:       si no
10:        si iaux == 2 entonces
11:          M_D(i, j) = 1
12:        fin si
13:        I_F(i, j) = I_1(i, j)
14:      fin si
15:    fin si
16:  fin para
17: fin para
```



(a) Buscando un píxel que cumpla con el umbral máximo.



(b) Llenando la zona con los píxeles que cumplan el umbral mínimo.



(c) Asignando el valor máximo a toda la zona.

Figura 3.10: Proceso de llenado de un mapa de diferencias.

(a) Imagen con el lobo de atrás nítido. (b) Imagen con el lobo de frente nítido.

Figura 3.11: Lobos de prueba

3.4. Experimentos preliminares

Se hicieron varias pruebas para comprobar la efectividad del algoritmo, tanto artificiales como reales. En las pruebas artificiales se utilizó una fotografía con dos lobos en ella [Orozco13]. En la primera se emborronó todo el lobo de atrás, simulando que se encuentra fuera de foco y en la segunda, se emborrono todo el lobo de enfrente. Por otro lado, en las pruebas reales, se utilizó el ambiente controlado descrito en la Sección 3.1, se fue variando el foco de la cámara y se hicieron las tomas, posteriormente se evaluaron los algoritmos para conocer su desempeño.

3.4.1. Pruebas artificiales

En esta subsección se mostrarán los resultados obtenidos de las pruebas artificiales, las imágenes usadas como se comentó anteriormente, fueron de dos lobos de tamaño 512x512.

En la Figura 3.11a puede observarse que el lobo de atrás se encuentra enfocado y el lobo de enfrente desenfocado. Mientras que en 3.11b es al revés.

En la Figura 3.12a se encuentra la imagen original, donde ambos lobos se encuentran enfocados. En la Figura se encuentra el mapa de decisión real de dicha imagen. Donde

(a) Fotografía de los lobos sin distorsión.

(b) Mapa de decisión original.

Figura 3.12: Lobos nítidos y mapa original de decisión

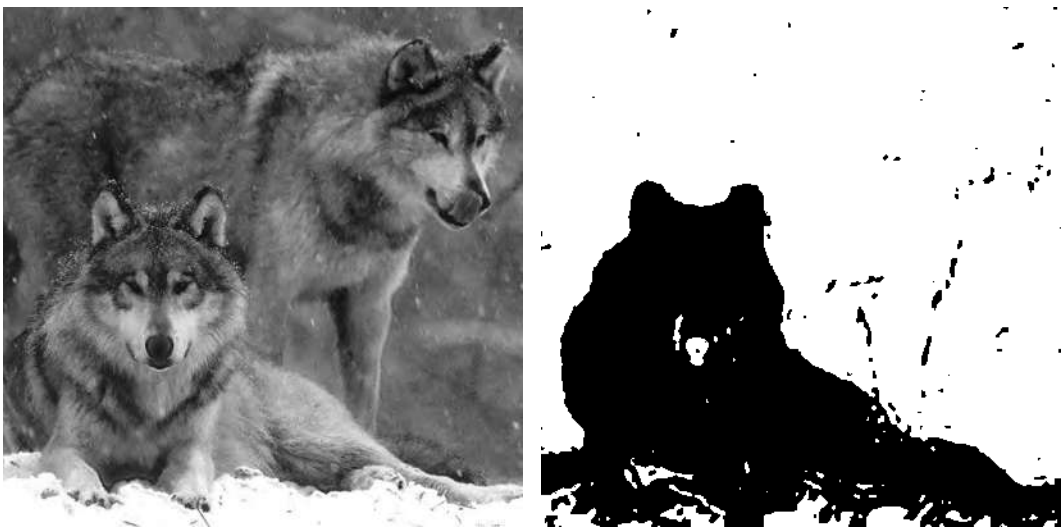
(a) Lobos fusionados usando el método FPA

(b) Mapa de decisión usando el método FPA

Figura 3.13: Lobos fusionados usando FPA y su mapa de decisión

el color negro nos indica que esa zona pertenece a una imagen y el blanco nos indica que esa zona pertenece a la otra.

La Figura 3.13a, muestra la fusión usando el método FPA descrito en la Sección 2.1, en la Figura 3.13b se muestra el mapa de decisión obtenido. Se puede observar en



(a) Lobos con fusión usando el método SFF (b) Mapa de decisión usando el método SFF

Figura 3.14: Lobos fusionados usando SFF y su mapa de decisión

esta figura que el mapa de decisión sale muy disperso. Aunque la imagen fusionada se ve bastante bien a simple vista. El porcentaje de error que se obtiene de realizar esta fusión es del 15.913 %.

En la Figura 3.14 se muestran la fusión y el mapa de decisión obtenidos respectivamente, esta vez usando el método SFF descrito en la sección 2.2. Como se puede ver los resultados mejoraron bastante comparados con el resultado anterior, pero aún se siguen teniendo zonas que no deberían estar ahí. El porcentaje de error que se obtiene de realizar esta fusión es del 3.56789 %, como se puede ver baja drásticamente con respecto al anterior, pero las zonas siguen sin estar bien definidas.

Por último en la Figura 3.15 se muestra el resultado del algoritmo propuesto, del lado derecho la imagen fusionada y del lado izquierdo el mapa de decisión, como se puede apreciar la diferencia con el mapa original es muy pequeña. Se puede ver que las zonas están bien definidas, pero con algunos errores en los bordes. El porcentaje de error de este mapa fue de 0.632477 %. Aquí el error fue de 1658 píxeles de 262,144, por lo que el resultado fue realmente bueno.

La Tabla 3.1 muestra el tiempo que tardo cada método para dar el resultado que se mostró antes. Como puede observarse, el método más rápido fue el FPA, seguido de la

(a) Lobos con fusión usando el algoritmo propuesto

(b) Mapa de decisión usando el algoritmo propuesto

Figura 3.15: Lobos fusionados por el algoritmo propuesto y su mapa de decisión

propuesta y por último el SFF-

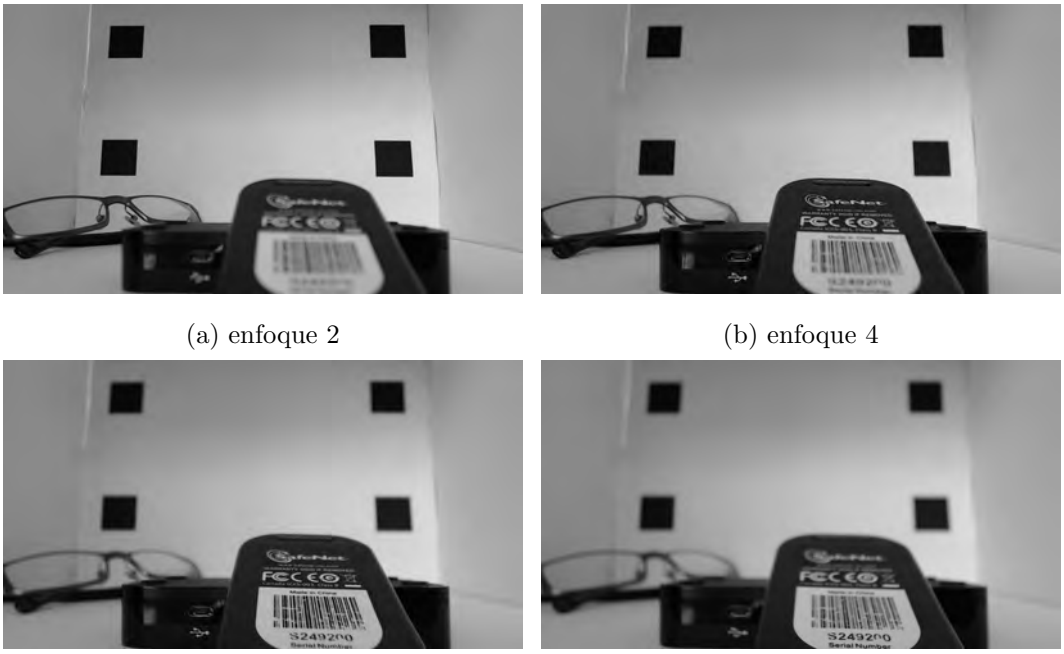
Método	**Tiempo en ms.**
FPA	257
SFF	660
Propuesta	458

Tabla 3.1: Tiempo que tardaron los métodos para obtener el mapa de decisión y la imagen fusionada, para 2 imágenes artificiales.

3.4.2. Pruebas reales

Las pruebas reales se hicieron en un ambiente controlado con imágenes de 1860x1036 píxeles.

En la Figura 3.16 se pueden observar varios objetos dentro de la caja de cristal, unos lentes, un hub usb y un llavero de banco con un código de barras y un número de serie (NetKey), además, se observa en el fondo el patrón utilizado para obtener las matrices de transformación proyectiva. En la Figura 3.16a se puede notar que los objetos cercanos a la cámara se encuentran fuera de foco, mientras que los lentes y el fondo se ven más claros.



(a) enfoque 2 (b) enfoque 4

(c) enfoque 6 (d) enfoque 8

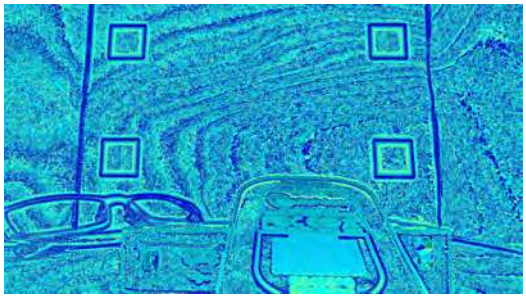
Figura 3.16: Tomas de las pruebas reales

En la Figura 3.16d se observa que casi todo se encuentra desenfocado y sólo la parte más cercana a la cámara se encuentra enfocada, esto es con un enfoque 8. Se utilizaron imágenes variando el enfoque de la 2 a la 8, pero sólo se muestran del paso 2, 4, 6 y 8 debido a la similitud que hay en los pasos intermedios.

La Figura 3.17 muestra los resultados obtenidos usando FPA. En la Figura 3.17a se puede ver la fusión de las 8 tomas, como puede observarse es un resultado muy bueno, debido a que la gran mayoría se ve enfocada, mientras que la Figura 3.17b se puede ver el mapa de decisión generado. Las zonas oscuras indican que el píxel se encuentra lejos, mientras que las zonas más claras indican que el píxel se encuentra cercano. A pesar que la fusión es buena, el mapa de decisión no sale como se esperaba. En la Figura 3.20a se pueden ver algunos detalles de la fusión original al hacer un acercamiento, en la zona que se acercó, se pueden ver las letras el NetKey y algunas de ellas tienen ruido a su alrededor.

En la Figura 3.18b se pueden observar los resultados obtenidos por el método SFF. En la Figura 3.18a se observa la fusión de las 8 imágenes, como se ve en la figura, los

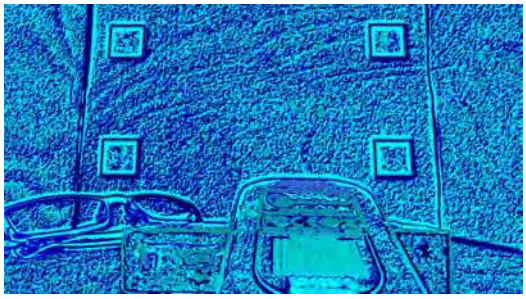
(a) Resultado usando el método FPA



(b) Mapa de decisión usando el método FPA

Figura 3.17: Prueba real usando FPA

(a) Resultado de la fusión usando el método SFF



(b) Mapa de decisión usando el método SFF

Figura 3.18: Prueba real usando SFF

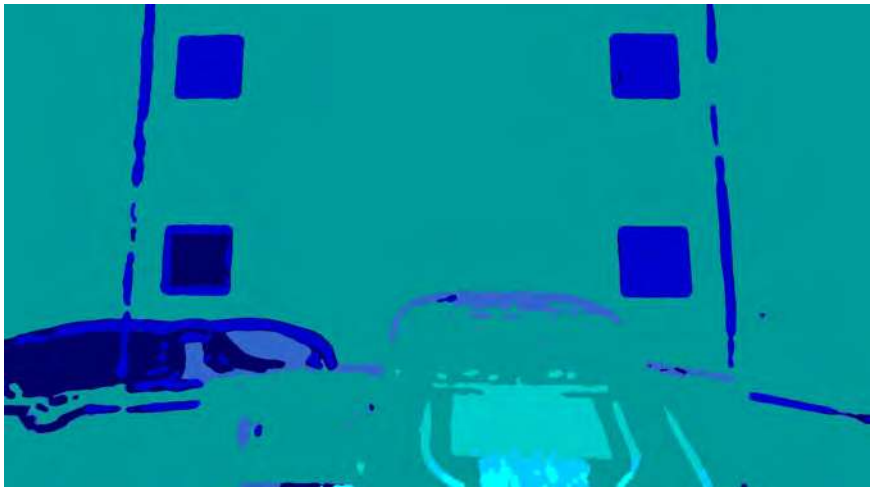
resultados obtenidos no son muy buenos. En la Figura 3.20b se puede ver mas de cerca que los resultados alcanzados en una prueba real no fueron los esperados.

En la Figura 3.19 muestra el resultado usando el algoritmo propuesto, el cual alcanza resultados muy buenos. Como se puede ver en la Figura 3.20c, las letras salen muy claras y se ven mucho mejor que el detalle utilizando FPA.

La Figura 3.20 muestra un acercamiento de las figuras anteriores sobre las letras del NetKey, para observar mejor las diferencias entre los algoritmos.

La Tabla 3.2 muestra el tiempo que tardo cada método. Como puede observarse en la tabla, el método más rápido fue el FPA, después la propuesta y por último SFF. Cabe destacar que no se considera el tiempo que se tarda en obtener las imágenes. Los experimentos con las 20 imágenes con diferente enfoque se verán en el Capítulo 4.

(a) Resultado de la fusión usando el algoritmo propuesto

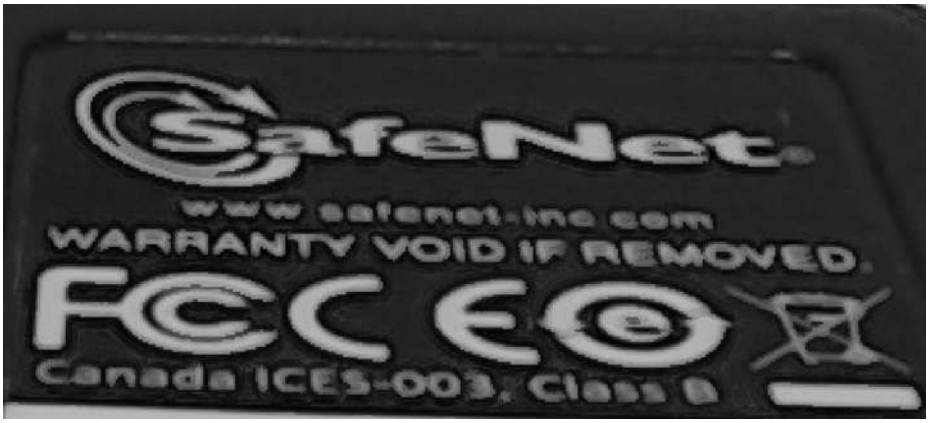


(b) Mapa de decisión utilizando el algoritmo propuesto

Figura 3.19: Prueba real, fusión usando el algoritmo propuesto

Método	**Tiempo en s.**
FPA	5
SFF	14
Propuesta	13

Tabla 3.2: Tiempo que tardaron los métodos para obtener el mapa de decisión y la imagen fusionada, para 7 imágenes reales.



(a) Acercamiento en las letras de la Figura 3.17a.

(b) Acercamiento en las letras de la Figura 3.18a.

(c) Acercamiento en las letras de la Figura 3.19a.

Figura 3.20: Prueba real, acercamiento en la zona de las letras

3.5. Concluciones

Capítulo 4

Experimentos y resultados

4.1. Prueba del módulo de visión 3D del robot

Para evaluar la visión del robot, se compararon el MDD y el mapa de decisión. Primero se evaluaron los mapas de disparidad y después los mapas de decisión. Las pruebas fueron realizadas con una resolución de la cámara un poco más baja, debido a que el tiempo de ejecución era de 2 o 3 segundos en las pruebas y se buscaba rapidez al obtener el MDD y el mapa de decisión, la resolución final de las imágenes después de recortarlas en el paso de la rectificación quedó de: 502x343 píxeles. A continuación se muestran los resultados obtenidos de los mapas de disparidad.

La Figura 4.1 muestra dos tomas de la misma escena desde las cámaras izquierda y derecha. Se pueden observar en las imágenes un tablero de ajedrez, un telefono celular y un borrador al frente y de fondo un monitor y algunas otras cosas. Las cámaras fueron calibradas y rectificadas. De este par de imágenes se calcularon los mapas de disparidad utilizando OpenCV. La Figura 4.2 muestra los mapas de disparidad obtenidos por los diferentes algoritmos que ofrece OpenCV.

Como puede observarse en la Figura 4.2a se puede ver el MDD usando el algoritmo BM, se muestra con colores oscuros la disparidad baja y con colores claros la disparidad alta. Se puede ver que a pesar de que el dodecaedro esta hasta al frente, muchas zonas las muestra con color oscuro. En la Figura 4.2b se muestra el MDD usando el algoritmo SGBM,

(a) Imagen izquierda (b) Imagen derecha

Figura 4.1: Imágenes tomadas con la cámara izquierda y derecha

aquí se puede apreciar que los resultados son un poco mejores, ya que las zonas están un poco más definidas, pero aún así existen zonas donde la disparidad no es la correcta. En la Figura 4.2c se muestre el MDD obtenido usando el algoritmo VAR, se puede apreciar que las zonas no están muy bien definidas en comparación con el algoritmo SGBM. El mejor resultado que se obtuvo en estas pruebas de un MDD fue utilizando SGBM. Una vez decidido que algoritmo se iba a utilizar en la visión estéreo, se procedió a evaluar la visión por enfocado.

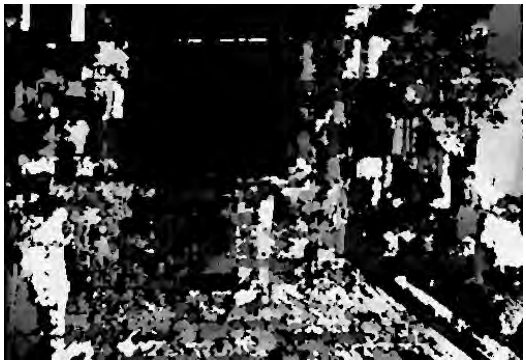
La Figura 4.3 muestra una escena tomada con la cámara izquierda del arreglo estéreo. En las tres imágenes se puede observar que es la misma toma del par estéreo pero se aplicaron las matrices de transformación homográficas como se vio en el Capítulo 3.

Se fusionaron las imágenes desde el enfoque 1 al 20 y los resultados de todos los esquemas se muestran en las Figuras 4.4, 4.5 y 4.6.

Como puede verse en la Figura 4.4a la fusión de la imagen usando FPA es muy buena, pero en la Figura 4.4b se puede ver que el mapa de decisión sale muy disperso, por lo que no se puede usar esta información para mapear a un mapa de profundidad, debido a que quedaría igual muy disperso.

Como puede verse en la Figura 4.5a la fusión de la imagen usando SFF no es muy buena y esto también se puede ver en la Figura 4.5b, donde el mapa de decisión muestra que la mayoría de las cosas pertenecen a la zona mas cercana a la cámara, lo cual no es cierto. Por lo tanto este algoritmo tampoco se utilizará para obtener el mapa de profundidad.

(a) Mapa de disparidad usando BM



(b) Mapa de disparidad usando SGBM

(c) Mapa de disparidad usando VAR

Figura 4.2: Mapas de disparidad obtenidos usando los 3 métodos que ofrece OpenCV.

Como puede verse en la Figura 4.6a la fusión de la imagen usando el algoritmo propuesto es muy buena y la Figura 4.6b corrobora el resultado obtenido, mostrando un mapa de decisión mas uniforme que los demás, donde la mayoría de los objetos pertenecen a la zonas que marca el mapa, incluso la base utilizada se puede ver bien definida en la parte de abajo. La parte más blanca de hasta el frente indica que la base es lo que se encuentra mas cercano a la cámara, después se puede ver el tono más claro de gris que delimita la siguiente zona de la base. Esta información si es útil para obtener un mapa de profundidad, pero aún así se pueden ver algunos errores de las zonas, por ejemplo, la parte del fondo debería pertenecer a la zona negra, pero se ve casi toda gris.

El tiempo que tardaron los métodos en fusionar las 20 imágenes y calcular el mapa de decisión se muestra en la siguiente tabla.

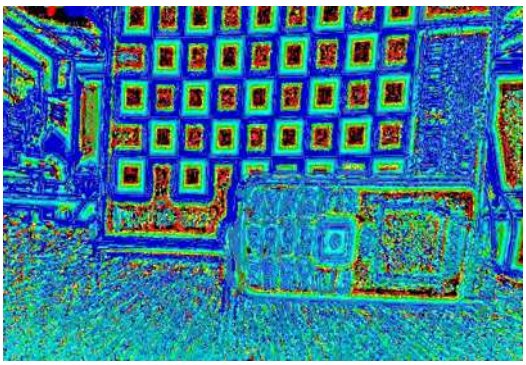
(a) enfoque 1

(b) enfoque 5

(c) enfoque 13

Figura 4.3: Pasos de enfoque 1, 5 y 13

(a) Fusión usando FPA

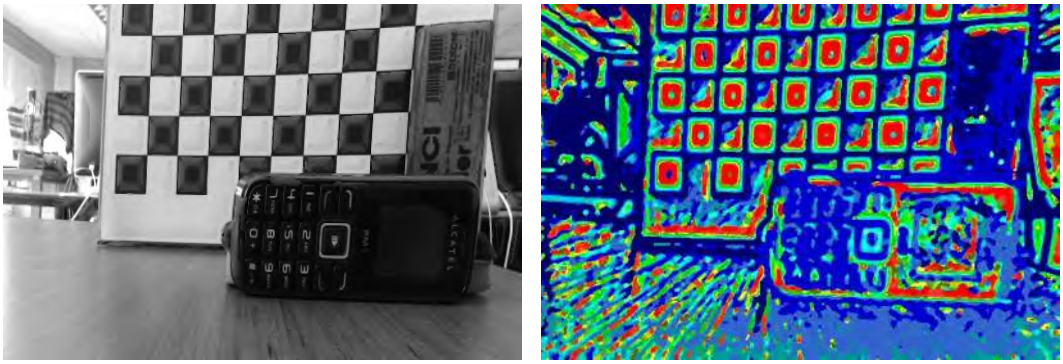


(b) Mapa de decisión usando FPA

Figura 4.4: Resultado final usando FPA

Método	**Tiempo en ms.**
FPA	788.08
SFF	3233.56
Propuesta	2775.27

Tabla 4.1: Tiempo que tardaron los métodos para obtener el mapa de decisión y la imagen fusionada, para 20 imágenes reales.



(a) Fusión usando SFF (b) Mapa de decisión usando SFF

Figura 4.5: Resultado final usando SFF

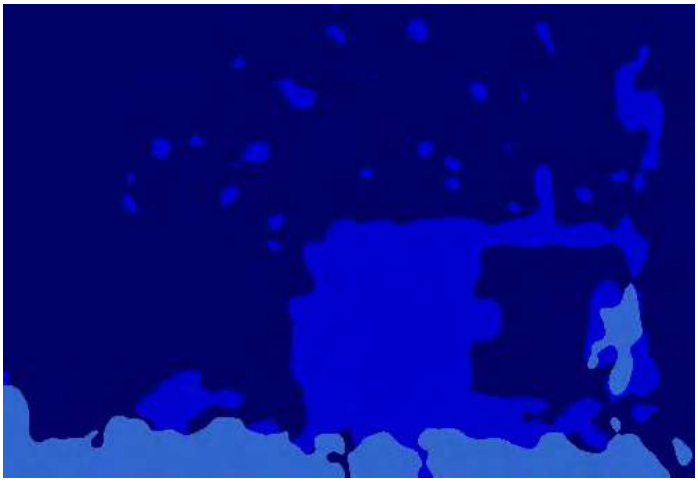
Como puede verse en la Tabla 4.1, el método mas rápido es el FPA, seguido de nuestra propuesta y por último el SFF. La Tabla 4.2 muestra el tiempo que tardaron los algoritmos programados en OpenCV para calcular el mapa de disparidad, como puede observarse, el algoritmo más rápido es el de BM, seguido de SGBM y por último el algoritmo de VAR.

Algoritmo	**Tiempo en ms.**
BM	49.77
SGBM	158.73
VAR	306.22

Tabla 4.2: Tiempo que tardaron los algoritmos programados en OpenCV para obtener el mapa de disparidad.

La siguiente tarea sería combinar ambos esquemas para obtener más información acerca de la profundidad, pero esto se dejará como un trabajo futuro.

(a) Fusión usando el algoritmo propuesto



(b) Mapa de decisión usando el algoritmo propuesto

Figura 4.6: Resultado final usando el algoritmo propuesto

Capítulo 5

Conclusiones

5.1. Conclusiones Generales

A lo largo del proyecto presentado en esta tesis, se tuvieron algunas dificultades, algunas fueron cuestión de tiempo, otras de implementación, pero en general los objetivos planteados fueron cumplidos en su gran mayoría.

En la visión por enfocado la cercanía de los objetos no es un problema, debido a que se cuentan con 20 diferentes pasos de enfoque (en las cámaras que se utilizaron). Esto nos brinda información acerca de la lejanía de los objetos, desde 3 c.m. hasta 30 c.m. ya que después de esa distancia, todos los objetos se ven iguales. La única desventaja de utilizar esta técnica es que tomar las 20 imágenes nos lleva mucho tiempo, hasta 54 segundos utilizando una resolución de 1920x1080 píxeles. Pero una vez obtenidas las muestras la fusión se lleva a cabo relativamente rápido, hasta en 13 segundos utilizando la misma resolución.

5.2. Trabajos Futuros

La mayoría de los trabajos futuros que se muestran a continuación, se dieron bajo la necesidad de tiempo. Algunos fueron surgiendo como ideas de posibles alternativas a los problemas que se fueron presentando durante el proyecto.

1. Proponer un método para obtener los parámetros de los filtros pasa altas y de los

mapas de decisión que se calculen de manera automática, en lugar de obtenerlos a prueba y error.

2. Integrar el sistema ya desarrollado con *The Robot Operating System* (ROS)[1].

3. Utilizar algoritmos que resuelven el problema del *Simultaneous Localization and Mapping* [SLA15] para construir un mapa y que el robot se mantenga localizado.

4. Utilizar indicaciones más simples para el robot, por ejemplo, ir al punto X y que el robot sea capaz de trazar una ruta y seguirla sin perderse.

5. Integrar el Filtro de Kalman Extendido para mejorar la presición a la hora de mantener localizado al robot.

6. Incluir estadísticas de conexión de usuarios, horarios de conexión e impacto social en la interfaz correspondiente al súper usuario, esto con el fin de conocer la aceptación del público en general por el prototipo diseñado.

7. Integrar los resultados obtenidos de la visión por enfocado con la visión estéreo.

8. Integrar las lecturas del telémetro láser junto con las de la visión 3D propuesta para obtener mejor presición.

9. Incluir un medidor de nivel de baterías en el robot, para conocer la carga exacta de las mismas y no correr el riesgo de que se termine a mitad de una presentación.

10. Proponer un nuevo algoritmo para calcular un MDD que no dependa de tantos parámetros.

11. Terminar el usuario administrador el cual tendrá el control completo del robot, además de que le permitirá ver en tiempo real quien se encuentra al robot como operador y quién se encuentra como observador. Además podrá acceder a las mimas funciones que un operador y si esté lo decide, estará facultado para desconectar a cualquier usuario que haga mal uso del robot o que trate de permanecer mucho tiempo conectado a el, esto con el fin de que se pueda tener para fines educativos y cualquier persona en

[1] http://www.ros.org/

cualquier parte del mundo puedan acceder al robot y observar de lo que es capaz, de esta manera, aseguraremos el buen uso del robot y evitaremos que alguien haga mal uso del mismo.

Apéndice A

Calibración de un par estéreo de cámaras.

La vista humana es un ejemplo de la visión estéreo, nuestros ojos serían como un par de cámaras separados por una distancia fija, por lo tanto percibimos profundidad a partir de las pequeñas diferencias que el cerebro encuentra entre lo que ven los ojos. De igual manera un par de cámaras nos pueden dar la información sobre la misma escena tomada con diferente ángulo. La Figura A.1 muestra dos cámaras con ejes ópticos paralelos, separados una cierta distancia.

Figura A.1: Par de cámaras alineadas horizontalmente.

Este capítulo está basado en el libro de [Bradski08] y [Prince12]. Si al lector le interesa ahondar más en algún tema mencionado a continuación, se le recomienda acudir a cualquiera de los dos libros.

A grandes rasgos, para obtener un mapa de profundidad utilizando la visión

estéreo, se requieren 4 pasos:

1. Remover la distorsión radial y tangencial de los lentes. Esto se realiza utilizando un patrón conocido (como un tablero de ajedrez) y relacionar los puntos en el espacio con los puntos en la imagen. Esto nos lleva a un sistema de ecuaciones que podemos resolver para remover las distorsiones antes mencionadas. Esto se abordará con más detalle en la Sección A.1 de calibración de cámaras.

2. Ajustar los ángulos y distancias entre las cámaras. Es decir, se colocan las cámaras alineadas, ya sea vertical u horizontalmente, como se muestra en la Figura A.1. Al igual que en la calibración, se puede usar un patrón para alinearlas correctamente, es decir, que queden perfectamente paralelas, ya que cuando se hace el arreglo de las cámaras, generalmente no quedan perfectamente alineadas. Este proceso de alineación por software es conocido como rectificación y se verá a detalle en la Sección A.2

3. Una vez realizada la rectificación se puede obtener el mapa de disparidad (Sección A.3). En general, el mapa de disparidad se obtiene observando el corrimiento de los píxeles de la imagen en la cámara derecha con respecto a los píxeles correspondientes de la imagen en la cámara izquierda en la misma fila (para pares estéreo horizontales), es decir, $d_x = x^d - x^i$. Donde x^d y x^i se refieren a la filas derecha e izquierda respectivamente y d_x denota la disparidad. Así los píxeles asociados a objetos cercanos tendrán una gran disparidad y los asociados a objetos lejanos tendrán una disparidad muy pequeña.

4. Conociendo el arreglo geométrico de las cámaras (separación entre ellas), entonces podemos convertir el mapa de disparidad en distancias de profundidad por triangulación. Este paso es llamado re-proyección, y la salida es un mapa de profundidad, donde a cada píxel se le asocia la profundidad correspondiente al objeto asociado a dicho píxel.

La Figura A.2 muestra a grandes rasgos los pasos que deben seguirse hasta antes de obtener el mapa de disparidad, que incluyen, calibrar las cámaras, rectificarlas y recortar la

imagen para quedarse únicamente con las áreas válidas. A continuación se verán con detalle cada uno de los pasos mencionados.

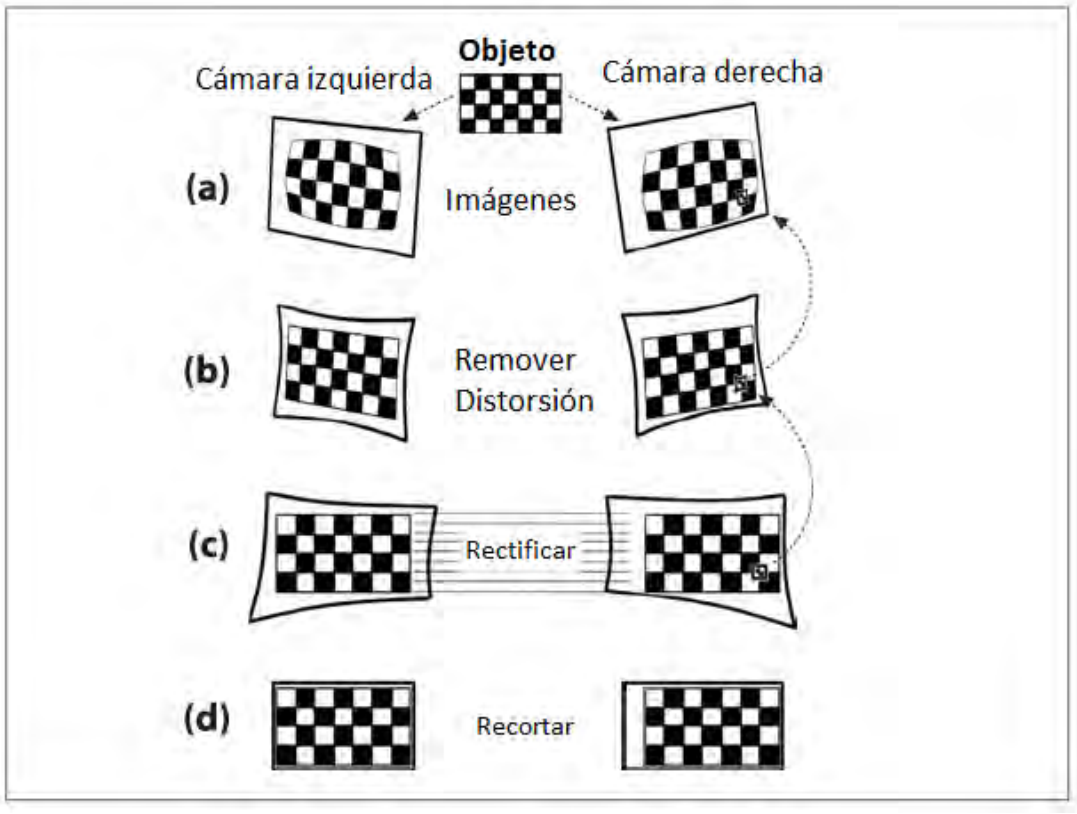


Figura A.2: Proceso hasta antes de obtener un mapa de disparidad [Bradski08].

A.1. Calibración de cámaras

Antes de aprender a calibrar una cámara, debemos saber como se modela una cámara, conocer los tipos de distorsión que presentan y posteriormente cómo calibrarla. A continuación se describe el modelo de una cámara.

A.1.1. Modelo de la cámara de Pin-hole

El modelo de una cámara convencional se puede apreciar en la Figura A.3. Las ecuaciones que modelan una cámara son las siguientes:

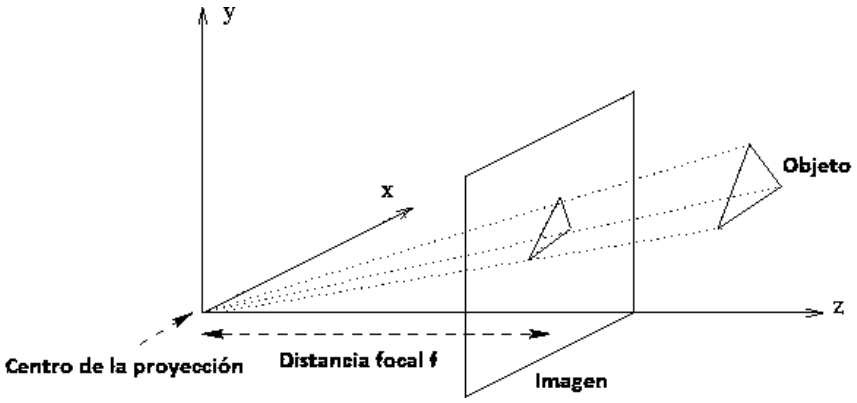


Figura A.3: Modelo de cámara convencional.

$$x = \left(\frac{X}{Z}\right) f_x + c_x,\ y = \left(\frac{Y}{Z}\right) f_y + c_y \tag{A.1}$$

Donde:

- (X, Y, Z) son las coordenadas del objeto en el sistema de referencia de la cámara.
- (x, y) son las coordenadas del objeto proyectado en la imagen con respecto al sistema de referencia del plano de la imagen.
- f_x, f_y son las distancias focales en los ejes x y y respectivamente,
- c_x, c_y son las coordenadas del centro óptico, el punto de intersección del eje óptico con el plano de la imagen en el sistema de referencia del plano de la imagen.

Note que la cámara tiene dos distancias focales, una para el eje x y la otra para el eje y, esto debido a las características específicas de los fotoreceptores del plano de la imagen [Prince12].

La Ecuación A.1 se puede pasar a coordenadas homogéneas, de esta forma se obtiene una expresión matricial y las operaciones con matrices son más fáciles de visualizar. Dicha representación quedaría como sigue:

$$q = MQ,\quad q = \begin{bmatrix} x \\ y \\ w \end{bmatrix},\quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix},\ Q = \begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} \tag{A.2}$$

Aquí q representa una posición en el plano de la imagen, Q una posición en 3 dimensiones del objeto que se proyecta al punto q en la imagen y M es la matriz de parámetros intrínsecos de la cámara. Las coordenadas reales x_r, y_r de la imagen, se obtienen como $x_r = x/w$, $y_r = y/w$.

A.1.2. Distorsión de los lentes

En teoría es posible definir lentes que no introducirán distorsión, en la práctica esto no es así debido a que los lentes no son perfectos. Esto se debe principalmente a razones de manufactura. Además es difícil mecánicamente alinear exactamente los lentes y el receptor de imagen. La distorsión radial se crea como resultado de la forma del lente y la distorsión tangencial se crea del proceso de ensamblado de la cámara completa.

La distorsión radial se aprecia más en los píxeles de las orillas de una imagen, la distorsión es 0 en el centro óptico de la imagen y se incrementa en la periferia. En la práctica, esta distorsión es pequeña y puede caracterizarse con algunos de los primeros términos de una función polinomial. En general, la localidad radial de un punto en la imagen puede ser re escalado de acuerdo a la siguiente ecuación:

$$x_{corregida} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{A.3}$$

$$y_{corregida} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{A.4}$$

Donde: (x, y) es la localidad original del punto distorsionado y $(x_{corregida}, y_{corregida}$ es la nueva localidad como resultado de la corrección. k_1, k_2 y k_3 son los coeficientes del polinomio, usualmente se utilizan 2 ó 3. Por último, r es la distancia radial del centro óptico de la imagen (c_x, c_y) al punto (x, y).

La distorsión tangencial se debe a defectos de fábrica, resultado de que los lentes no están exactamente paralelos al plano de la imagen. Esta distorsión es caracterizada como mínimo por dos parámetros adicionales, p_1 y p_2, con la siguiente ecuación:

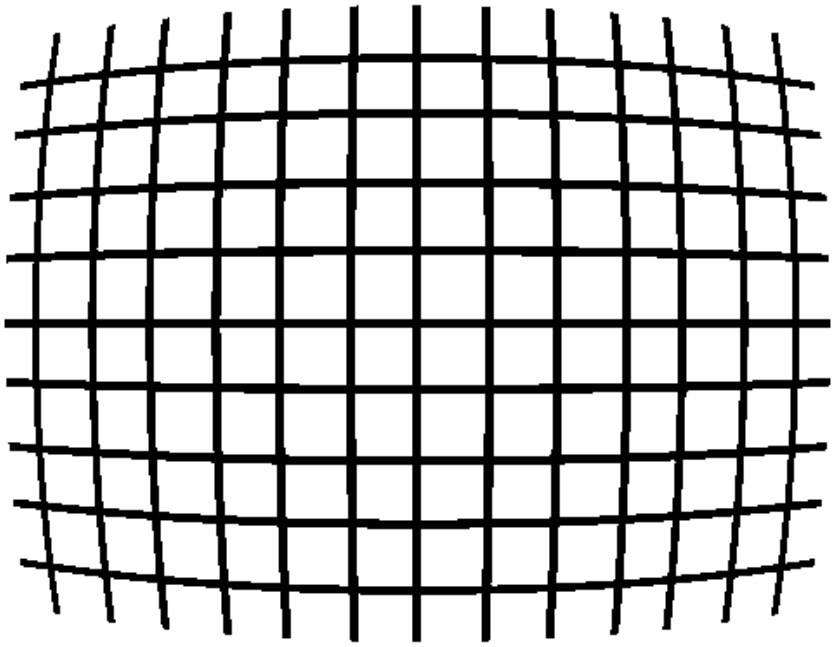


Figura A.4: Ejemplo de la distorsión radial en una cámara.

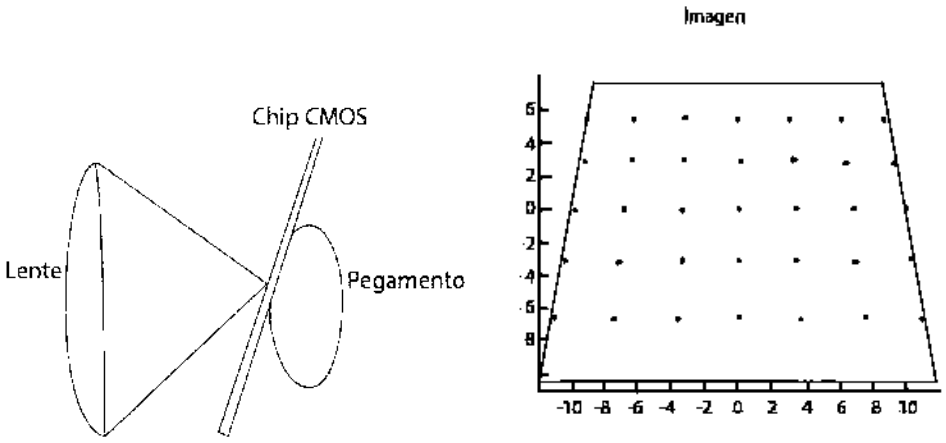
$$x_{corregida} = x + [2p_1y + p_2(r^2 + 2x^2)] \tag{A.5}$$

$$y_{corregida} = y + [p_1(r^2 + 2y^2) + 2p_2x] \tag{A.6}$$

Por lo tanto en total se requieren 5 coeficientes de distorsión $(k_1, k_2, k_3, p_1, p_2)$. Las Figuras A.4 y A.5 muestran un ejemplo de la distorsión radial y tangencial respectivamente. En la primera podemos observar una cuadrícula con distorsión en las orillas, mientras que en la segunda se ilustra lo que podría ocurrir al pegar el chip CMOS sin alinearlo correctamente con el eje óptico del lente (Figura A.5a) y el plano de la imagen que se tendría de ser así (Figura A.5b).

A.1.3. Parámetros extrínsecos de la cámara

Los parámetros extrínsecos de la cámara son la transformación de rotación y traslación que se deben aplicar a un objeto para que quede en coordenadas de la cámara, es decir, si tenemos un punto de un objeto P_o en coordenadas (X, Y, Z) con respecto a un



(a) Vista interna del ensamblaje de la cámara.

(b) Plano de la imagen con distorsión tangencial.

Figura A.5: Ejemplo de la distorsión tangencial en una cámara [Bradski08].

marco de referencia global, que rotación y que traslación deben aplicarse a P_o para que quede expresado con respecto al marco de referencia de la cámara. La ecuación que modela esta transformación es:

$$P_c = R(P_g - T) \tag{A.7}$$

Donde:

- P_g es un punto $(X_g, Y_g, Z_g)^T$ con respecto a un marco de referencia global.
- P_c es el punto P_g expresado en coordenadas (X_c, Y_c, Z_c) con respecto al marco de referencia de la cámara.
- R es una matriz ortonormal de dimensión $3x3$ que modela las transformaciones de rotación.
- T es un vector de dimensión $3x1$ que modela una traslación.

Si cambiamos a coordenadas homogéneas, el punto P_g se expresará como $P_g^H = (X_g, Y_g, Z_g, 1)^T$ y el punto P_c^H como (X_c, Y_c, Z_c, W_c).

$$P_c^H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_g \\ y_g \\ z_g \\ 1 \end{bmatrix} \tag{A.8}$$

$$P_c^H = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} P_g^H \tag{A.9}$$

Combinando esta ecuación con la Ecuación A.2, podemos obtener una ecuación reducida que integra los parámetros extrínsecos e intrínsecos (dejando aparte los parámetros de distorsión).

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_g \\ y_g \\ z_g \\ 1 \end{bmatrix} \tag{A.10}$$

En la literatura se describen muchos métodos para calcular los parámetros intrínsecos y extrínsecos de una cámara, sea por separado o combinando los parámetros en una sola matriz. En este trabajo se utilizan los algoritmos de calibración implementados en OpenCV, para facilitar el desarrollo de la tesis.

OpenCV [OpenCV215] utiliza un algoritmo basado en el método de Zhang [Zhang00] para detectar las distancias focales y los centros ópticos, pero utiliza el método basado en Brown [Brown71] (en OpenCV 1) y el método de Bouguet [Bouguet15] (en OpenCV 2) para obtener los parámetros de los coeficientes de distorsión radial y tangencial.

Para calibrar la cámara OpenCV utiliza una rutina llamada *calibrateCamera*(), esta función calcula los parámetros intrínsecos de la cámara y los coeficientes de distorsión, a partir de un patrón como un tablero de ajedrez. La función se puede ver en [OpenCV215] y es la descrita en la siguiente sección.

A.1.4. Calibración de la cámara con OpenCV

OpenCV proporciona la siguiente función en C++ para calibrar una cámara:

double calibrateCamera(InputArrayOfArrays objectPoints, InputArrayOfArrays imagePoints, Size imageSize, InputOutputArray cameraMatrix, InputOutputArray distCoeffs, OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs, int flags=0, TermCriteria criteria=TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30, DBL_EPSILON))

Esta función regresa el error de re-proyección y en un inicio los parámetros de entrada/salida pueden parecer muchos pero al describirlos pueden ya no parecer tan abrumadores.

- objetctPoints: Este parámetro es de entrada y se refiere a los puntos 3D del objeto en cuestión con respecto al sistema de referencia del patrón utilizado, se manda una matriz de $N * Kx3$, donde N es el número de tomas del objeto, K el número de puntos del patrón y 3 corresponde a las coordenadas en X, Y y Z.

- imagePoints: Este parámetro es de entrada y se refiere a las esquinas encontradas en las imágenes en coordenadas X, Y y es una matriz de $N * Kx2$, donde cada punto corresponde a un punto 3D del parámetro anterior.

- imageSize: Es el tamaño de la imagen en píxeles.

- cameraMatrix: Es la matriz de los parámetros intrínsecos de la cámara, puede ser una matriz de entrada/salida, si se proporciona como entrada se utiliza para calcular los coeficientes de distorsión y si no, se calcula y se regresa. Es una matriz de $3x3$.

- distCoeffs: Es la matriz de los coeficientes de distorsión de la cámara, puede ser una matriz de entrada/salida, si se proporciona como entrada se utiliza para calcular la matriz de los parámetros intrínsecos de la cámara y si no, se regresa. Es una matriz de $5x1(k_1, k_2, p_1, p_2, k_3)$ en ese orden.

- rvecs: Es una matriz de salida que contiene los vectores de rotación para cada objeto 3D.

- tvecs: Es una matriz de salida que contiene los vectores de traslación para cada objeto 3D.

- flags: Son las banderas que se pueden utilizar para calibrar la cámara. Por defecto se inicializa en 0.

- criteria: Es una variable que indica cuando debe detenerse el algoritmo, por defecto se inicializa con un número de iteraciones de 30 como máximo y un umbral de épsilon de DBL_EPSILON.

A.1.5. Ejercicio. Calibrar una cámara y mostrar resultado

Un buen ejercicio para entender mejor esto es calibrando una cámara. Para esto se usará una cámara Logitech C920 y un patrón de calibración.

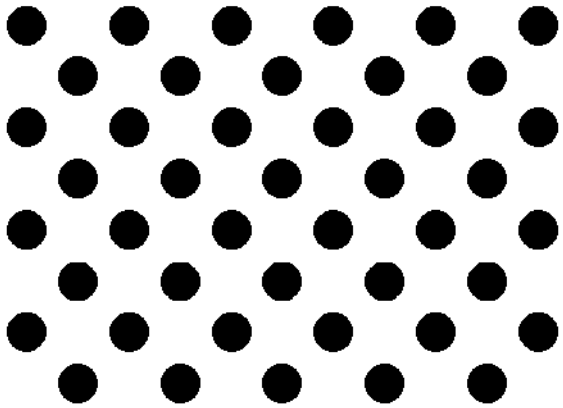


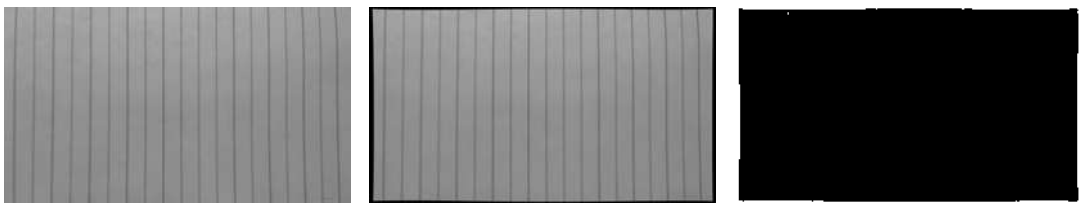
Figura A.6: Patrón utilizado para calibrar las cámaras.

En la Figura A.6 se puede ver un patrón de círculos asimétrico usado para calibrar la cámara, ya que se obtienen mejores resultados que al utilizar un tablero de ajedrez, como se puede ver en [Harguessa14].

La Figura A.7a muestra una fotografía de líneas rectas tomada con la cámara antes de la calibración, como se puede observar en la esquina superior izquierda, existe una ligera curvatura.

En la Figura A.7b se puede apreciar la misma fotografía removiendo la distorsión, se puede notar que a pesar de que la distorsión es pequeña el cambio es notorio.

La Figura A.7c muestra la resta entre la fotografía de la Figura A.7a y la fotografía en la Figura A.7b, se le aumento el contraste y el brillo para poder visualizar mejor dicha



(a) Imagen original I (b) Imagen sin distorsión I_s (c) $I - I_s$

Figura A.7: Resultados de la calibración

diferencia. Como se puede observar en el centro, la diferencia es menor y a medida que se aleja del mismo la diferencia va creciendo. Note que el centro óptico de la cámara no es exactamente el centro de la fotografía.

Los parámetros intrínsecos de la cámara se muestran en la Tabla A.1. En la primera fila se pueden apreciar la distancia focal en x (primera columna) y el centro óptico en x (tercera columna). La imagen es de 1920x1080 píxeles, si la lente estuviera perfectamente alineada, el centro óptico en x sería de 960. En la segunda fila segunda columna, se puede ver la distancia focal en y y en la tercera columna el centro óptico en y, que si fuera perfecto sería de 540.

1878.41	0.	988.13
0.	1947.24	573.79
0.	0.	1.

Tabla A.1: Parámetros intrínsecos de la cámara.

Los coeficientes de distorsión de la cámara se muestran en la Tabla A.2. Como se puede apreciar, los coeficientes son bajos, debido a que no existe mucha distorsión radial y tangencial.

El promedio de error de proyección obtenido fue de 1.06 %.

k_1	0.24
k_2	-0.65
p_1	0.00158
p_2	-0.00201
k_3	0.24

Tabla A.2: Coeficientes de distorsión de la cámara.

A.2. Rectificación

Una vez calibradas las cámaras, el siguiente paso es la rectificación de las mismas, es decir, físicamente las cámaras deberían estar alineadas horizontalmente, de tal forma que la imagen se vea a la misma altura en ambas cámaras. Como esto generalmente no es posible, debido a ligeras rotaciones y traslaciones, debe aplicarse la rectificación.

Para rectificar las cámaras, utilizaremos una función de OpenCV llamada *StereoRectify*, pero antes necesitaremos utilizar una función llamada *StereoCalibrate*.

La función *StereoCalibrate* se utiliza no solo para calibrar las cámaras, también se puede usar por separado para obtener las matrices de rotación y traslación antes mencionadas, se recomienda que primero se calibren las cámaras por separado y una vez hecho esto, se alimente esta función con los valores obtenidos para obtener mejores resultados.

A.2.1. Calibración estéreo con OpenCV

double stereoCalibrate(InputArrayOfArrays objectPoints, InputArrayOfArrays imagePoints1, InputArrayOfArrays imagePoints2, InputOutputArray cameraMatrix1, InputOutputArray distCoeffs1, InputOutputArray cameraMatrix2, InputOutputArray distCoeffs2, Size imageSize, OutputArray R, OutputArray T, OutputArray E, OutputArray F, TermCriteria criteria=TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30, 1e-6), int flags=CALIB_FIX_INTRINSIC)

Al igual que la función *calibrateCamera* mostrada en la sección anterior, los parámetros pueden ser muchos, pero analizándolos se ve que se pueden entender fácilmente.

- objetctPoints: Este parámetro es de entrada y se refiere a los puntos 3D del objeto

en cuestión con respecto al sistema de referencia del patrón utilizado, se manda una matriz de $N*Kx3$, donde N es el número de tomas del objeto, K el número de puntos del patrón y 3 corresponde a las coordenadas en X, Y y Z.

- imagePoints1 e imagePoints2: Son parámetros de entrada y se refieren a las esquinas encontradas en las imágenes (en la cámara 1 y 2 respectivamente) en coordenadas X, Y y es una matriz de $N * Kx2$, donde cada punto corresponde a un punto 3D del parámetro anterior. Para que un punto sea considerado válido, deberá estar tanto en imagePoints1 como en imagePoints2, deberá descartarse o se le introducirá mucho ruido a la función.

- cameraMatrix1: Son los parámetros intrínsecos de la cámara 1, se pueden proporcionar si se tienen, sino, los puede regresar la función.

- distCoeffs1: Vector de coeficientes de distorsión de la cámara 1, de igual manera, si se tienen se obtienen mejores resultados al proporcionarlos.

- cameraMatrix2: Son los parámetros intrínsecos de la cámara 2, se pueden proporcionar si se tienen, sino, los puede regresar la función.

- distCoeffs2: Vector de coeficientes de distorsión de la cámara 2, de igual manera, si se tienen se obtienen mejores resultados al proporcionarlos.

- imageSize: Es el tamaño de las imágenes, todas deben tener el mismo tamaño.

- R: Es la matriz de rotación entre las dos cámaras. Este es un parámetro de salida.

- T: Es el vector de traslación entra las dos cámaras. Este es un parámetro de salida.

- E: Es la matriz esencial. Este es un parámetro de salida.

- F: Es la matriz fundamental. Este es un parámetro de salida.

- criteria: Es la condición de paro para el algoritmo.

- flags: Son las banderas utilizadas para especificar que matrices serán utilizadas de entrada(s) y cuáles de salida.

Ahora tenemos lo necesario para aplicar la función *StereoRectify*. Esta función recibe la mayoría de los parámetros que dan de salida/entrada *StereoCalibrate* y nos permite alinear las cámaras horizontal o verticalmente, según sea el caso.

El procedimiento a seguir es el siguiente:

- Obtenemos las matrices de rotación (R) y traslación (T) de los sistemas de coordenadas de las cámaras (*StereoCalibrate*).
- Obtenemos las matrices de transformación de rectificación para las dos cámaras y las matrices de re-proyección (*R1, R2, P1 y P2*) utilizando *StereoRectify*, además obtenemos las áreas válidas para las imágenes.
- Re-mapeamos las imágenes utilizando la función *remap*.
- Cortamos las imágenes de acuerdo a las áreas válidas.

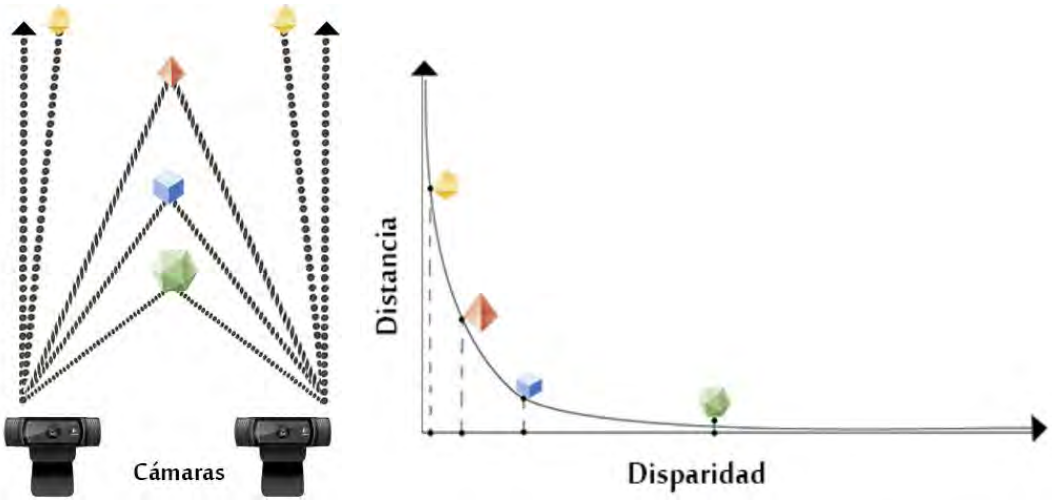
A.3. Mapa de disparidad

El cálculo del mapa de disparidad se hace a partir de las diferencias entre las dos imágenes, en nuestro caso, izquierda y derecha, esto es $d = x^l - x^r$. La Figura A.8a muestra dos cámaras que observan los mismos objetos desde diferentes ángulos y en la Figura A.8b se muestra una gráfica de la relación entre la distancia y la disparidad.

Para encontrar el mapa de disparidad los algoritmos más utilizados son el *Block matching algorithm*, *Semiglobal matching* [Hirschmuller08] y el *Variational Method* [Kosov08]. Se probarán los 3 y dichas pruebas se mostrarán en el Capítulo 4. Una vez que obtenemos el mapa de disparidad podemos utilizarlo para obtener el mapa de profundidad, tema de la siguiente sección.

A.4. Triangulación

Una vez que tenemos las imágenes sin distorsión, alineadas horizontal o verticalmente (según sea el caso) y la medición del mapa de disparidad, podemos obtener el mapa de profundidad por triangulación.



(a) Ilustración de objetos. (b) Gráfica de distancia contra disparidad.

Figura A.8: Relación entre distancia y disparidad

Como se vio en la Figura A.8, el mapa de disparidad es inversamente proporcional a la profundidad, se puede apreciar que hay una relación no lineal entre estos dos términos. Cuando la disparidad es cercana a 0, la profundidad tiende a infinito, es decir los objetos están muy alejados. Por otro lado, cuando la disparidad es muy grande, la profundidad tiende a 0. En consecuencia, la visión estéreo es utilizada para objetos que están relativamente cerca de las cámaras.

La Figura A.9 muestra como se podría usar la triangulación para obtener la profundidad de un objeto utilizando dos cámaras. Donde Z es la profundidad hacía el objeto, f es la distancia focal de las cámaras (se asume la misma en este ejemplo), x^l y x^r es el objeto proyectado en la imagen izquierda y derecha respectivamente, O_l y O_r son los centros de las cámaras izquierda y derecha respectivamente, T es la distancia física entre las cámaras, c_x^{left} y c_x^{right} son los centros ópticos de las cámaras izquierda y derecha en x. En [Bradski08] utilizan triángulos similares para obtener la Ecuación A.11.

Podemos ver que la profundidad es inversamente proporcional a la disparidad. La profundidad se puede derivar fácilmente usando triángulos similares y la Ecuación A.11 nos da la distancia $Z_{(i,j)}$ a la cual está el píxel (i, j).

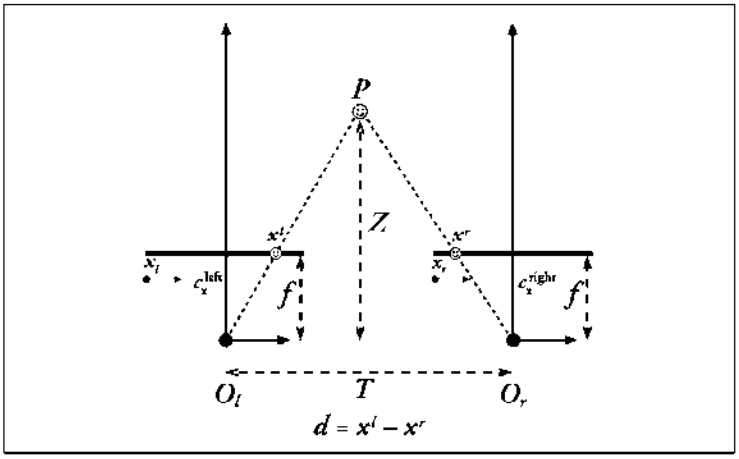


Figura A.9: Ilustración del arreglo de dos cámaras y la profundidad.

$$Z_{(i,j)} = \frac{Tf}{(x_l - x_r)} \tag{A.11}$$

Donde:

- T es la distancia entre los centros de las cámaras en el eje x.
- f es la distancia focal de la cámara (asumiendo que es la misma para ambas).
- $x^l - x^r$ es la disparidad de (i, j) píxel.

En OpenCV se usa la función *reprojectImageTo3D* para re-proyectar un mapa de disparidad en una imagen en 3D.

Apéndice B

Robot de telepresencia

Nuestro robot, contará con visión estéreo y visión por enfocado, además de utilizar un telémetro láser, para obtener información sobre el entorno, las cuales serán usadas en la detección de colisiones, evitando así daños al robot. Citando las leyes de la robótica de Isaac Asimov [Asimov50]:

1. Un robot no hará daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.

2. Un robot debe obedecer las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1^a Ley.

3. Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1^a o la 2^a Ley.

Nuestro robot sólo será capaz de seguir una variación de la tercera ley.

Además de esto, el usuario final podrá ver lo que ve el robot desde una interfaz desde un navegador web, dicha interfaz contará con 3 tipos de usuarios diferentes:

1. Observador: El cual como su nombre lo indica, tendrá la facultad de ver solamente lo que ven las cámaras.

2. Operador: Este usuario podrá manipular al robot para moverlo en el ambiente, además podrá ver lo que las cámaras están observando y el audio del ambiente, podrá proyectar

su propia imagen y voz (si está equipado) al robot, contará con un límite de tiempo (definido por el Súper usuario) para que no agote las baterías del robot.

El robot de telepresencia contará con un esquema de locomoción diferencial, utilizando motores de pasos para las ruedas y para el sistema de inclinación de las cámaras. Las placas del robot son en su mayoría de acrílico y utilizará un Odroid-U3[1] para las tareas de alto nivel(como el manejo de las cámaras, micrófono, etc.) y un Arduino Mega[2] para las tareas de bajo nivel (como mover motores, obtener lecturas del telémetro láser, etc.).

En los capítulos anteriores se abordaron temas como calibración, rectificación, mapa de disparidad, enfocado, etc. En este capítulo, se centrará en utilizar los desarrollos anteriores en el robot de telepresencia. Para hacerlo se hizo una interfaz para el robot y una para el usuario que se conecte al robot, pero primero se describirán brevemente las partes que conforman al robot.

B.1. Hardware del robot de telepresencia

A continuación se listan los componentes físicos que constituyen el robot de telepresencia.

1. Dos motores de pasos de 200 pasos/revolución a 12 V.
2. Dos motores de pasos para el sistema de giro de las cámaras 2 y 3.
3. Tres controladores para los motores de pasos.
4. Un servomotor.
5. Dos ruedas locas.
6. Dos ruedas para el movimiento.
7. Dos baterías plomo-ácido de 6 V., 6 Ah.
8. Tres cámaras Logitech C920.

[1]Es un ordenador de placa única [Odr15]
[2]Placa con un microcontrolador [Ard15]

9. Un microcontrolador Arduino Mega.
10. Un Odroid U3.
11. Un hub USB.
12. Una bocina.
13. Un display.
14. Una tarjeta de red inalámbrica.

En general son las partes que constituyen el robot de telepresencia, dejando de lado los tornillos, rondanas, bases de acrílico, etc. Una vez ensambladas todas estas partes, se procedió a programar la interfaz para el usuario final.

B.2. Interfaz para el usuario

La interfaz servirá para comunicarse con el robot y enviarle comandos. Para el desarrollo de la interfaz se utilizó lenguaje HTML5 [HTM15], estilos CSS3 [CSS15], lenguaje JavaScript [JS15] y programación en PHP5 [PHP15], así como algunas librerías de código abierto como el framework Bootstrap[BOO15] y una librería basada en WebRTC[WEB15].

La interfaz está compuesta por un menú principal en la parte superior y en la parte bajo el menú el cuerpo. La Figura B.1 muestra la pantalla de inicio del cliente se puede ver en la parte superior el menú y en la parte inferior se encuentra un carrusel de imágenes. A continuación se describe el menú.

1. Botón RobotFIE. Al pulsarse este botón únicamente se redirecciona a la pantalla principal.
2. Botón Inicio. Al pulsarse este botón al igual que el botón anterior, se redirecciona a la pantalla principal.
3. Botón Conectarse. Este botón le muestra al usuario un menú desplegable como se muestra en la Figura B.2. Las opciones que se pueden ver aquí son: Administrador,

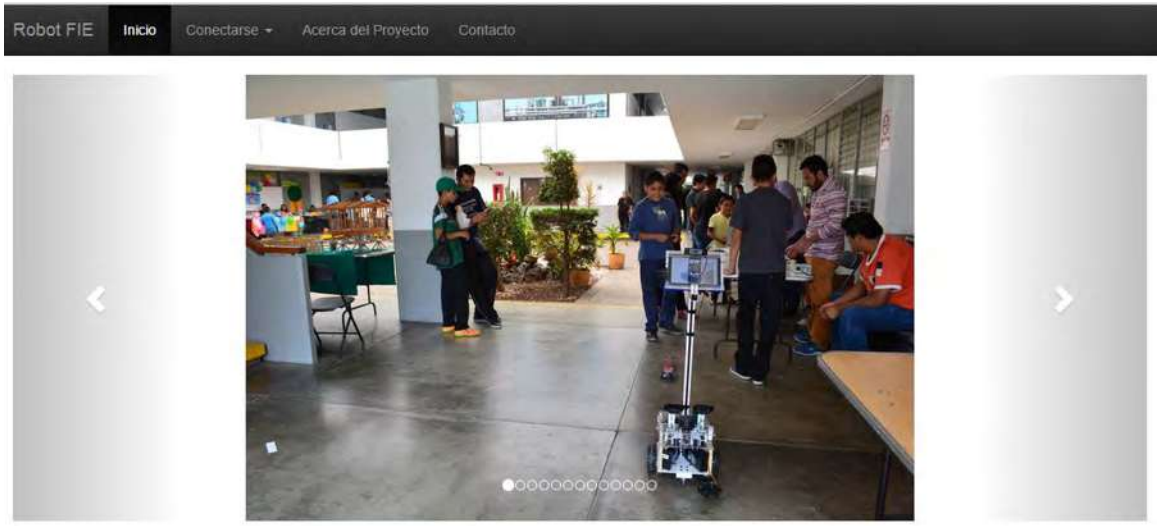


Figura B.1: Pantalla principal del cliente

Operador, Observador y un apartado de *Estatus* que muestra los usuarios conectados y una breve reseña acerca del robot FIE. Los primeros tres botones se verán a detalle más adelante.

4. Botón Acerca del Proyecto. Como su nombre lo indica, al ser pulsado este botón se muestra una reseña acerca del proyecto.

5. Botón Contacto. Este botón muestra algunos datos de contacto para las personas que estén interesadas en el proyecto.

Como se mencionó al inicio del capítulo, la interfaz tendrá diferentes tipos de usuarios. Al igual que en sistemas operativos, los tipos de usuarios serán utilizados para definir los privilegios que tendrá cada uno de ellos. A continuación se describirán los tres tipos de usuarios, comenzando por el usuario observador.

B.2.1. Usuario observador

El usuario observador únicamente tendrá el privilegio de observar las cámaras inferiores del robot, es decir, las cámaras 2 y 3; esto porque la cámara de arriba se utilizará exclusivamente para el operador o el administrador. Para acceder a esta interfaz, el

Figura B.2: Menú Conectarse.

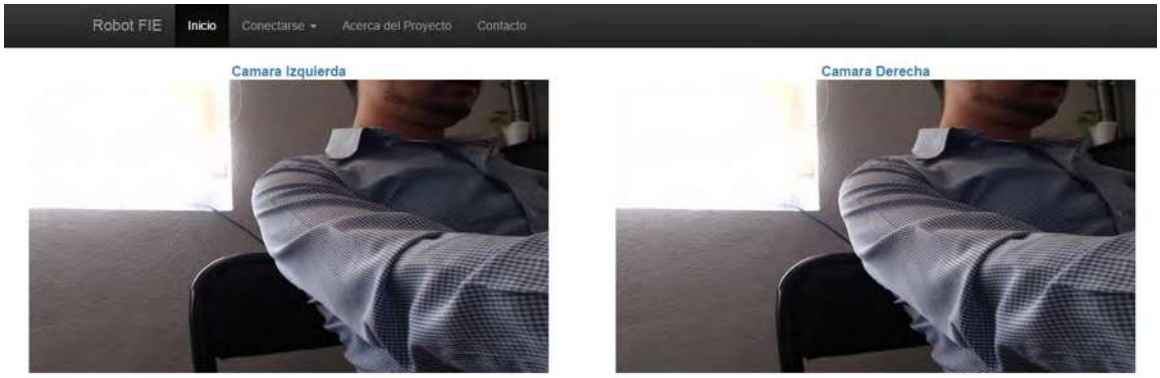


Figura B.3: Pantalla para el usuario solo ver

usuario deberá dar clic en el botón Conectarse y posteriormente en Observador.

La Figura B.3 muestra la interfaz que se desarrolló para aquellos usuarios que sólo quieran ver lo que observan las cámaras 2 y 3 del robot. En esta interfaz no es posible comunicarse con el robot, pero se lleva un registro de los usuarios que se conecten al robot sólo para observar.

Como puede observarse en la figura anterior, es una pantalla muy básica. Se describe a continuación el usuario operador.

B.2.2. Usuario operador

Para conectarse a esta interfaz el usuario deberá dar clic en el botón Conectarse y después en Operador. Con esto tendrá acceso automático a la interfaz, sólo podrá haber un operador a la vez, pero se podrán conectar múltiples usuarios a esta interfaz teniendo deshabilitadas las opciones siempre y cuando ya se encuentre un operador activo.

La Figura B.4 muestra la interfaz creada para dicho usuario. En la parte inferior del menú se puede observar un apartado que muestra los usuarios en línea. En el ejemplo se puede ver que se encuentra conectado el usuario robotFIE, esté representa al robot de telepresencia. Si se encuentra más usuarios conectados tipo operador aparecerán en este listado. La interfaz ofrece también la posibilidad de que los usuarios se llamen entre sí. Abajo del listado de usuarios se pueden ver los botones que podrá utilizar el operador para mover al robot.

El usuario operador, será capaz de tener un control completo sobre el robot, impidiendo que otros usuarios que lo quieran operar lo hagan, esto con la finalidad de que sólo un usuario operador pueda utilizarlo a la vez. Se diseñó una interfaz de comunicación, donde el usuario debe presionar un botón para marcar al robot (véase Figura B.5) y así acceder a la cámara frontal superior y a los micrófonos. Una vez establecida la conexión, el usuario podrá observar lo que la cámara del robot esté observando y escuchar lo que se escuche por el micrófono. Además, como se mencionó antes, tendrá la capacidad de mover al robot en el ambiente con comandos sencillos de avanzar y rotar. Para avanzar, se propusieron dos modos de avance. El primero, cada que el usuario le indique al robot que debe avanzar, el robot avanzará 5 cm hacia adelante. El segundo, el usuario le puede indicar al robot que avance y este no se detendrá hasta que el usuario lo ordene o hasta que encuentre un objeto con el cual vaya a colisionar, de ahí la importancia del sistema para detectar colisiones. Para rotar por otro lado, el usuario podrá indicar cuantos grados debe girar el robot y en qué sentido, es decir, izquierda o derecha.

La Figura B.6 muestra una ampliación a la figura anterior, de los botones para el control de movimiento del robot. Los primeros botones que se muestran son los de Adelante y Atrás seguidos de un cuadro de texto, donde el operador puede indicar la cantidad en

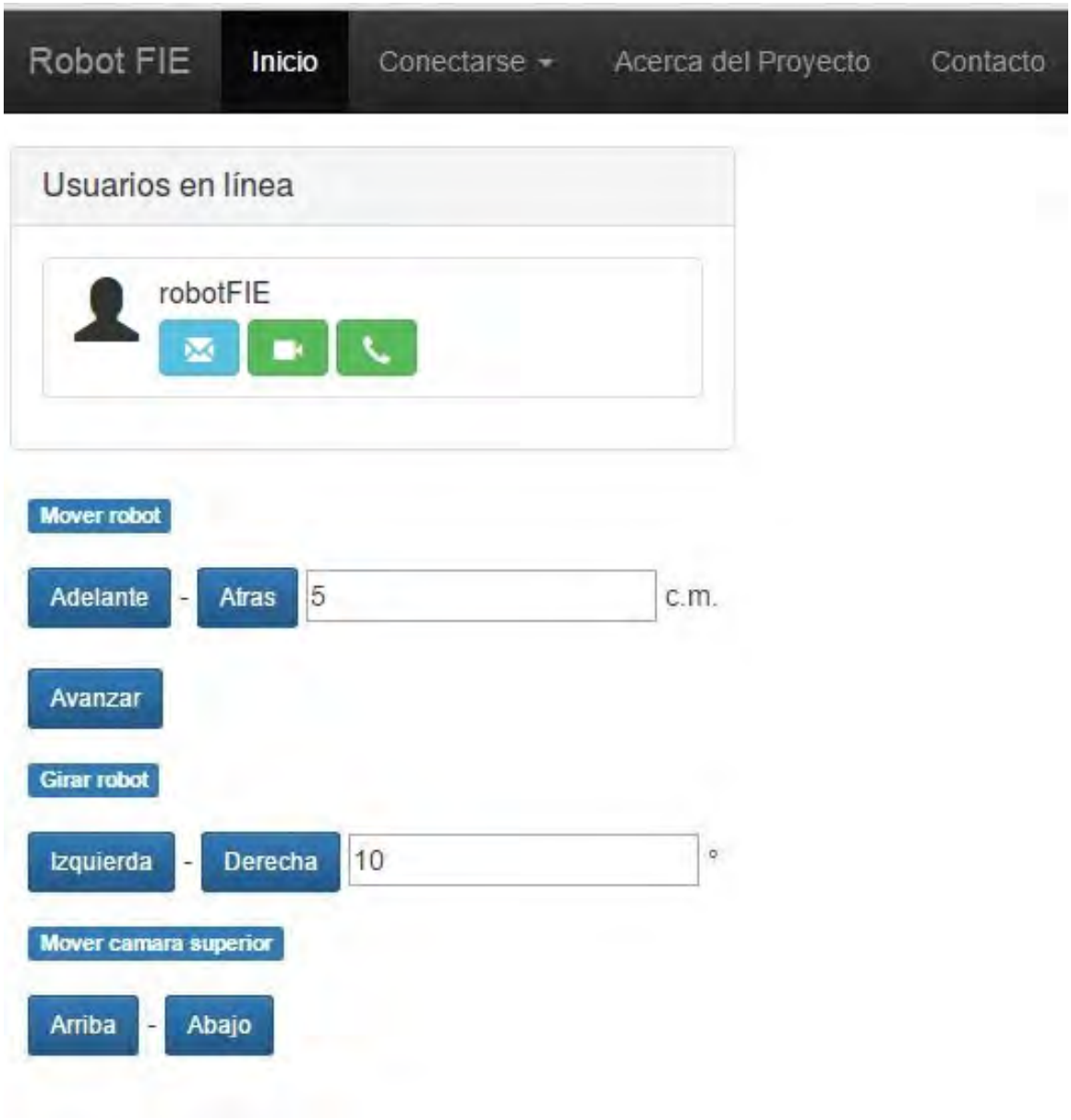


Figura B.4: Pantalla para el usuario operador

Figura B.5: Acercamiento a los botones disponibles para el operador

centímetros que desea que el robot avance. El siguiente botón es el de avanzar, que como ya se mencionó anteriormente sirve para que el robot avance hacia adelante indefinidamente. Los siguientes dos botones sirven para rotar, izquierda y derecha respectivamente; también se cuenta con un cuadro de texto para el operador pueda indicar la cantidad en grados que desea que el robot rote, ya sea a la izquierda o a la derecha. Los últimos botones sirven para mover la cámara superior, arriba y abajo respectivamente; en esto botones no se cuenta con un cuadro de texto, debido a que los parámetros para subir y bajar la cámara están fijos a 3^o.

Como pudo observarse, esta pantalla esta mucho mas completa que la pantalla para el usuario observador. La última opción para que se conecten los usuarios es la pantalla del usuario administrador, que se describe a continuación.

Figura B.6: Botones para el control de movimientos del robot

B.2.3. Usuario administrador

Este usuario solamente se encuentra en el menú de usuarios, pero se sigue trabajando en la implementación del mismo, por lo que se agregará a trabajos futuros.

B.3. Módulos de audio y vídeo para el robot

La emisión y recepción de vídeo se hizo de dos maneras, la primera fue conectando el robot directamente con el Cliente, es decir PeerConnection (PC), esta forma permite que las dos computadoras se enlacen y puedan compartir información, en este caso, audio, vídeo y texto; la segunda fue de forma Cliente-Servidor (CS), donde el cliente hace la petición de una imagen y el servidor le envía una imagen. En el esquema de CS, no fue posible compartir el micrófono del robot, debido a que no todos los navegadores soportaban el protocolo lo cual fue un inconveniente ya que se busca compartir el audio y vídeo desde cualquier navegador, por lo tanto se decidió utilizar la primera opción (PC) para compartir el audio y vídeo de la cámara 1 y la segunda opción (CS) se utilizará sólo para el usuario observador.

Para compartir el audio y vídeo de la cámara 1 mediante la opción PC, se utilizó una librería basada en WebRTC. Se diseñó la interfaz web utilizando una plantilla de Bootstrap para que la interfaz detecte desde que dispositivo se accede a la web y se ajuste en caso de que sea un Smarthphone, computadora o tablet. Esto permite que el usuario pueda acceder desde el dispositivo que desee, sin tener que programar una interfaz por cada dispositivo diferente.

B.4. Módulo para el control de movimientos del robot

Para controlar el movimiento del robot, se creó una interfaz que se comunica al puerto serial del microcontrolador Arduino Mega que se encuentra en el robot. Se utiliza el protocolo de comunicación serial para manipular los motores de pasos, los servo-motores. El control de movimiento es controlado utilizando Ajax desde la interfaz del cliente, para comunicarse con el servidor y darle instrucciones. La interfaz gráfica se pudo apreciar en la

sección del usuario operador B.2.2.

B.5. Módulo de mediciones de distancia utilizando el telémetro láser

Este módulo se encuentra en desarrollo y no se presentará formalmente. Se probó el telémetro láser y se realizaron algunas pruebas, pero debido a la falta de tiempo no se logró integrar con los otros sistemas ya desarrollados. Se presentará como propuesta la integración de este módulo al desarrollo completo en los trabajos futuros.

Referencias

[Ard15] Arduino. http://www.arduino.cc/, 2015. Consulta: Febrero 2015.

[Asada97] Asada, N., Fujiwara, H., y Matsuyama, T. Edge and depth from focus. *International Journal of Computer Vision*, 1997.

[Asimov50] Asimov, I. *I, Robot.* Gnome Press, 1ª edón., 1950.

[BOO15] Bootstrap. http://getbootstrap.com/, 2015. Consulta: Junio de 2015.

[Bouguet15] Bouguet, J. Matlab calibration tool. J.Y. Bouguet, http://www.vision.caltech.edu/bouguetj/calib_doc/, 2015. Consulta: Enero 2015.

[Bradski08] Bradski, G. y Kaehler, A. *Learning OpenCV.* O'Reilly Media, Inc, 1ª edón., 2008. ISBN 978-0-596-51613-0.

[Brown71] Brown, D. C. Close-range camera calibration. *Photogrammetric Engineering*, págs. 855–866, 1971.

[CSS15] Css3. http://www.w3schools.com/css/css3_intro.asp, 2015. Consulta: Junio de 2015.

[Dudek11] Dudek, G. y Jenkin, M. *Computational Principles of Mobile Robotics.* Cambridge University Press, 2ª edón., 2011. ISBN 978-0-521-87157-0.

[Gonzalez07] Gonzalez, R. C. y Woods, R. E. *Digital Image Processing.* Prentice Hall, 3ª edón., 2007. ISBN 978-0131687288.

[Harguessa14] Harguessa, J. y Strange, S. Infrared stereo calibration for unmanned ground vehicle navigation. *Unmanned Systems Technology XVI*, 2014.

[Hirschmuller08] Hirschmuller, H. Stereo processing by semiglobal matching and mutual information. *PAMI*, 2008.

[HTM15] Html5. http://www.w3schools.com/html/html5_intro.asp, 2015. Consulta: Junio de 2015.

[JS15] Js. http://www.w3schools.com/js/, 2015. Consulta: Junio de 2015.

[Kosov08] Kosov, S. *Multi - View 3D Reconstruction with Variational Method*. Proyecto Fin de Carrera, Saarland University, 2008.

[Li13] Li, S., Kang, X., Hu, J., y Yang, B. Image matting for fusion of multi-focus images in dynamic scenes. *Information Fusion*, 2013.

[LOG15] Logitech c920. http://support.logitech.com/product/hd-pro-webcam-c920, 2015. Consulta: Agosto de 2015.

[Minhas09] Minhas, R., Mohammed, A. A., Wu, Q. J., y Sid-Ahmed, M. A. 3d shape from focus and depth map computation using steerable filters. *6th International Conference, ICIAR 2009*, 2009.

[Odr15] Odroid. http://www.hardkernel.com/main/main.php, 2015. Consulta: Febrero 2015.

[OpenCV215] OpenCV2. Calib 3d. http://docs.opencv.org/modules/calib3d/doc/, 2015. Consulta: Enero 2015.

[Orozco13] Orozco, R. I. *Fusión de imágenes multifoco por medio de filtrado de regiones de alta y baja frecuencia*. Proyecto Fin de Carrera, Universidad Michoacana de San Nicolás de Hidalgo, 2013.

[PHP15] Php5. http://www.w3schools.com/php/, 2015. Consulta: Junio de 2015.

[Prince12] Prince, S. J. *Computer vision: models, learning and inference*. 2012.

[Romero14] Romero, L., García, M., y Gómez, C. A tutorial on the total least squares method for fitting a straight line and a plane. *INTERNATIONAL CONGRESS ON MECHANICAL AND ELECTRIC ENGINEERING*, 2014.

[Siegwart04] Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. *Introduction to Autonomous Mobile Robots*. 3ª ed$^{\text{ón}}$., 2004. ISBN 978-0-262-01535-6.

[SLA15] Slam. https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping, 2015. Consulta: Junio de 2015.

[Wang07] Wang, J. y Cohen, M. F. Optimized color sampling for robust matting. *Computer Vision and Pattern Recognition*, 2007.

[WEB15] Webrtc. http://www.webrtc.org/, 2015. Consulta: Junio de 2015.

[Zhang00] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, págs. 1330–1334, 2000.

[Zhong99] Zhong, Z. A categorization of multiscale-descomposition-based image fusion schemes with a performance study for a digital camera application. *Proceedings of the IEEE*, págs. 1315–1326, 1999.