Universidad Central De Venezuela
Facultad de Ciencias
Escuela de Computación

Realidad Aumentada en Android para la decoración de espacios interiores

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por
Rafael Ángel Dominguez Di Bisceglie
Tutor:
Prof. Héctor Navarro

Caracas, Mayo de 2017

Universidad Central de Venezuela.
Facultad de Ciencias
Escuela de Computación

ACTA

Quienes suscriben, Miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado con el título **"REALIDAD AUMENTADA EN ANDROID PARA LA DECORACIÓN DE ESPACIOS INTERIORES"**, el cual es presentado por Rafael Dominguez, de Cédula de Identidad 17.400.208 , a los fines de optar al título de Licenciado en Computación, dejamos en constancia lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 19 de Mayo de 2017, a la 2:00 pm, para que su autor lo defendiera en forma pública, en el CCG de la Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, lo cual se realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que les fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de 20 puntos.

En fe de lo cual se levanta la presente acta, en Caracas a los 19 días del mes de Mayo del año 2017.

Prof. Héctor Navarro
Tutor

Prof. Rhadames Carmona
Jurado

Prof. Esmitt Ramirez
Jurado

Agradecimientos

Le agradezco a mi mamá, a Charlie y a Peter, por su apoyo incondicional durante todos estos años.

A mis amigos, Daniel, Leonardo, Jorge, Alejandra, Juan, Andreina, Carlos, Raffaele, Carmen, Gary, Alexander y Teresa, por ofrecerme su incondicional ayuda y apoyo, durante estos años.

A mis profesores, especialmente a los profesores, Walter Hernandez, Mercy Ospina, Eugenio Scalise, Rhadames Carmona y Concettina Di Vasta, por compartir sus conocimientos y ayudarme a convertirme en un buen profesional.

A mi tutor Héctor Navarro, que me ha acompañado a lo largo de casi toda la carrera. Gracias por siempre estar ahí, por compartir tus conocimientos y brindarme tu apoyo a lo largo de todos estos años.

A todos, gracias por haber compartido conmigo esta etapa tan importante en mi vida.

Rafael Dominguez

Dedicatoria

Este Trabajo Especial de Grado, esta dedicado a mi hermano Carlos Augusto y a todas aquellas personas, que la llama de sus vidas se ha extinguido antes de tiempo.

Rafael Dominguez

Resumen

Titulo

Realidad Aumentada en Android para la decoración de espacios interiores

Autor

Rafael Ángel Dominguez Di Bisceglie

Tutor

Prof. Héctor Navarro

Hoy en día con la masificación de los dispositivos móviles y el incremento notorio de sus capacidades de computo, ha cambiado drásticamente la forma en que las personas interactúan con su entorno (ej. vídeos, fotos, geolocalización), otras personas (ej. redes sociales, chats, videoconferencias) e incluso la manera en la que realizan sus actividades diarias (ej. acceso a su información bancaria, compras en línea). Además de estos cambios, los dispositivos móviles han permitido poner en uso otras tecnologías que requieren un mayor poder de cómputo, como es el caso de la realidad aumentada, que en el caso de este trabajo, su aplicación es en el área del diseño de interiores.

El objetivo principal de este trabajo de grado, es la creación de una aplicación móvil de realidad aumentada para la decoración de espacios interiores. Esto se logra, aumentado el ambiente real de un espacio interior con modelos 3D virtuales por medio de la cámara del dispositivo, lo que permite al usuario decorar un espacio con elementos virtuales y observar el resultado en tiempo real. El tipo de realidad aumentada usada, es sin la presencia de marcadores físicos sino apoyándose en técnicas de visión del computador para extraer los rasgos naturales del espacio a decorar, y además, incorporando técnicas modernas como ORB-SLAM para la ubicación y trazado de mapas de forma simultánea, que provee información de la ubicación y orientación de la cámara dentro del espacio interior.

Palabras Claves

Android, realidad aumentada, visión por computador, SLAM, ORB, ORB-SLAM, diseño interiores, computación gráfica, dispositivos móviles.

Índice general

Índice de Figuras

Índice de Tablas

Introducción

Hoy en día con la masificación de los dispositivos móviles (teléfonos inteligentes y tabletas) y el incremento notorio de la capacidad de procesamiento de las CPU y GPU disponibles en los mismos, las personas cuentan con una poderosa herramienta a su disposición.

Los dispositivos móviles han cambiado drásticamente la forma en que las personas interactúan con su entorno (ej. vídeos, fotos, geolocalización), otras personas (ej. redes sociales, chats, videoconferencias) e incluso la manera en la que realizan sus actividades diarias (ej. acceso a su información bancaria, compras en línea). Además de estos cambios, los dispositivos móviles han permitido poner en uso otras tecnologías que requieren un mayor poder de cómputo, como es el caso de la realidad aumentada.

La realidad aumentada busca enriquecer la experiencia de las personas a través de la inclusión de información virtual en el entorno real donde nos desenvolvemos, a través de la estimulación de los sentidos, en el caso los dispositivos móviles generalmente es visual a través de la cámara del dispositivo.

La realidad aumentada ofrece infinidad de nuevas posibilidades de interacción, que hacen que esté presente en muchos y varios ámbitos, como son la arquitectura, el entretenimiento, la educación, el arte, la medicina o las comunidades virtuales.

Por tal motivo, este trabajo se enfoca fundamentalmente en el desarrollo de un software de realidad aumentada para el área de decoración de interiores en los dispositivos móviles, y está estructurado de la siguiente manera:

El Capítulo I, contiene el planteamiento del problema que dio origen al presente trabajo, el objetivo general, los objetivos específicos y la justificación de la solución que se presenta.

En el Capítulo II, se muestra el marco conceptual, en el cual se presentan los fundamentos teóricos investigados que servirán de base para dar soporte al desarrollo de este trabajo.

En el Capítulo III, se especifican las actividades que se llevaron a cabo para el desarrollo de la solución planteada.

En el Capítulo IV, se muestran las pruebas y resultados del rendimiento de la aplicación en diferentes dispositivos.

Finalmente, se presentan las conclusiones, así como también la bibliografía y las referencias digitales consultadas para la elaboración de este trabajo.

Capítulo 1

Problema de Investigación

Este capítulo abarca el planteamiento del problema que se desea resolver con este trabajo, la definición de los objetivos que se desean alcanzar, algunos antecedentes relacionados con el área o que intentan solucionar el mismo problema y por último la justificación de porque resolver este problema.

1.1. Planteamiento del problema

La realidad aumentada es la visión a través de un dispositivo tecnológico, directa o indirecta, de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real.

Uno de los campos de aplicación de esta tecnología es en el área de la decoración de interiores, que es la que se encarga de adornar o distribuir los espacios, como por ejemplo decidir que tipo de muebles se debe colocar. El proceso de decoración de un espacio interior, requiere de imaginación (¿ cómo se verán los elementos ?, ¿ si el tamaño es el adecuado ?, entre otras cosas) y/o de un proceso de modelado digital del ambiente y sus elementos, utilizando herramientas CAD. Esto requiere de tiempo, es más complejo realizarlo y es una aproximación virtual del espacio a decorar. Por lo tanto, contar con una solución móvil que integre la información virtual con la real, accesible y en tiempo real que facilite esta labor sería ideal.

1.2. Objetivos la Investigación

1.2.1. Objetivos Generales

Implementar una aplicación de realidad aumentada para dispositivos móviles con sistema Android para la inserción de modelos 3D en espacios interiores estáticos.

1.2.2. Objetivos Específicos

- Insertar modelos 3D en formato wavefront (.obj) en la escena en tiempo real.
- Permitir aplicar transformaciones de escalamiento, traslación y rotación a los objetos insertados en la escena en tiempo real.
- Registrar y alinear los objetos virtuales en la escena utilizando marcadores generados a partir de las características naturales de la escena en tiempo real.
- Renderizar en tiempo real los objetos virtuales dentro del campo de visión de la cámara del dispositivo móvil a partir del reconocimiento de los marcadores generados.
- Calibrar la cámara del dispositivo móvil.
- Implementar un catálogo de modelos 3D preseleccionados más comunes en los espacios interiores.
- Realizar una recopilación de modelos 3D para ser catalogados.

1.3. Antecedentes

En la actualidad existen varias aplicaciones móviles de realidad aumentada enfocadas al área de decoración interior, la gran mayoría son basadas en marcadores y poseen soporte para las plataformas android, iOS o ambas. Algunas de las aplicaciones mas destacadas son:

- **IKEA Catalog**, es una aplicación móvil gratuita para Android, permite la navegación y visualización del catálogo de productos de IKEA. Esta aplicación es de realidad aumentada basada en marcadores.
- **Decolabs**, es una aplicación móvil gratuita para iOS, permite la decoración de interiores virtualmente usando realidad aumentada basada en marcadores, con un componente social para compartir y evaluar diseños.
- **Curioos Art Print Visualizer**, es una aplicación móvil para Android e iOS, que funciona como una tienda en linea para vender arte, se apoya en la realidad aumentada con marcadores que se colocan en la pared para ver las obras.
- **Augment**, es una aplicación móvil para Android, para la inserción de objetos 3D usando realidad aumentada utilizando marcadores.

También existen una gran cantidad de trabajos de investigación sobre el tema de realidad aumentada. Uno de los principales en el área de realidad aumentada sin marcadores, es el realizado por Iryna Gordon y David G. Lowe, titulado "Scene Modelling, Recognition and Tracking with Invariant Image Features", donde presentaron una sistema completo de realidad aumentada sin marcadores, que no requiere precalibración, conocimiento previo de la geometría de la escena o el uso de marcadores especiales.[10]

1.4. Justificación del problema

La decoración de interiores es una tarea común en la vida cotidiana de las personas, conocer si una decoración es adecuada para un espacio interior no es una tarea simple, la cual requiere de tiempo, imaginación y/o el apoyo en herramientas digitales.

Elaborar una herramienta de realidad aumentada que facilite esta labor, permitiría a los usuarios explorar distintas opciones de decoración en tiempo real y en el espacio interior donde se encuentre, de una manera rápida y sencilla, con lo que pueden tomar decisiones más expeditas como por ejemplo, cual es la decoración que más gusta para un espacio determinado, si se tiene el espacio suficiente, el adquirir o no una pieza para decorar.

Capítulo 2

Marco Conceptual

En este capítulo se describen los fundamentos teóricos que servirán de base para dar soporte al desarrollo de este trabajo. Los temas que se desarrollan a continuación son, procesamiento digital de imágenes, visión por computador, los sistemas de realidad aumentada, la geometría multivista, los sistemas SLAM y ORB-SLAM.

2.1. Procesamiento Digital de Imágenes

2.1.1. Procesamiento de Imágenes

Una **imagen** puede ser definida como una función bidimensional f(x,y), donde x e y son coordenadas espaciales (plano) y la amplitud de f en cualquier par de coordenadas (x,y) se denomina intensidad o nivel de gris de una imagen en ese punto. Cuando los valores de x,y y los valores de intensidad de f son todos finitos y discretos, podemos llamar a una imagen como una imagen digital[26].

El **procesamiento de imágenes** es tratar imágenes usando procesamiento de señales, donde la entrada es una imagen, una serie de imágenes o un vídeo, tales como una fotografía o un cuadro de vídeo y la salida, puede ser una imagen o un conjunto de características o parámetros relacionados con la imagen.

La mayoría de las técnicas de procesamiento de imagen involucran tratar la imagen como una señal bidimensional y aplicar técnicas de procesamiento de señales estándares a la misma. Las imágenes también pueden ser procesadas como señales tridimensionales, donde la tercera dimensión corresponde al tiempo o la eje z.

2.1.2. Imagen digital

Una **imagen digital** f[m,n] es descrita en un espacio 2D discreto a partir de una imagen f(x,y) a través de un proceso de muestro, que es frecuentemente referido como digitalización[26].

La imagen f(x,y) es dividida en N filas y M columnas, representación tipo matricial. La intersección de una fila con una columna es denominada un píxel. El valor asignado a las coordenadas enteras [m,n] $m \in [0, M-1], n \in [0, N-1]$ es f[m,n]. Como se puede ver en la Figura 2.1.

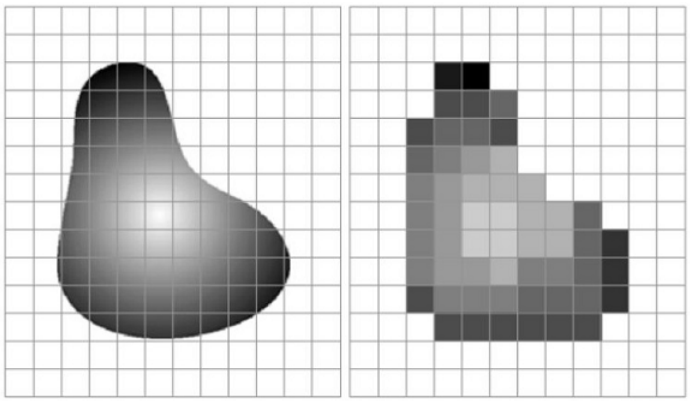


Figura 2.1: Digitalización de una imagen continua

Los valores asignados a cada píxel, es el promedio de intensidad en el píxel, redondeado al valor entero más cercano. El proceso de representar la amplitud de una señal 2D en una coordenada dada como un valor entero con L distintos niveles de grises, es usualmente referido como cuantización de la amplitud o simplemente cuantización.

El número de distintos niveles de gris es usualmente una potencia de 2. Eso significa que, $L = 2^B$ donde B es el número de bits en la representación binaria de los niveles de intensidad o gris. Cuando B es mayor a 1, se habla de que la imagen está en escala de grises y cuando B es igual 1, se dice que es una imagen binaria. En una imagen binaria solo existen dos niveles de grises a los cuales se puede referir, por ejemplo como "negro" y "blanco" o "0" y "1".

2.1.3. Procesamiento Digital de Imágenes

El **procesamiento digital de imágenes** es el uso de una computadora para realizar procesamiento de imágenes digitales. El procesamiento digital de imágenes tiene varias ventajas sobre el procesamiento de imágenes. Permite aplicar una mayor cantidad de algoritmos a los datos de entrada y puede evitar problemas como la acumulación de ruido y distorsión de la señal durante el procesamiento[26].

Dado que una imagen está definida en dos dimensiones (o más en algunos casos) el procesamiento digital de imágenes puede ser modelado como un sistema multidimensional (sistemas donde existen más de una variable independiente).

2.1.4. Algunas relaciones básicas entre los píxeles

En las imágenes digitales existen algunas relaciones importantes entre los píxeles. Entre ellas tenemos los vecinos de un píxel, adyacencia, conectividad, regiones y frontera, las cuales se explicarán a continuación[26].

Vecinos de un píxel

Un píxel p en las coordenadas (x,y) tiene 4 vecinos, 2 horizontales y 2 verticales cuyas coordenadas están dadas por,

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

A este conjunto de píxeles se llaman **4-vecinos de p**, se denota como $N_4(p)$. Cada píxel está a una distancia unitaria de (x,y) y algunos de los vecinos de p están fuera de la imagen digital si (x,y) está en el borde de la imagen.

Los cuatro vecinos diagonales de p tienes coordenadas

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

y se denota como $N_D(p)$. Estos puntos juntos con los 4-vecinos, se llaman **8-vecinos de p**, se denota como $N_8(p)$. Como se mencionó anteriormente es posible que algunos de los vecinos estén fuera de la imagen si (x,y) está en el borde de la imagen.

Adyacencia, Conectividad, Regiones y Frontera

Sea V el conjunto de valores de intensidad usados para definir la adyacencia. En una imagen binaria, $V = \{0\}$ si nos estamos refiriendo a la adyacencia de los píxeles con valor 0. En una imagen en escala de grises, la idea es la misma, solo que V puede contener más elementos. Por ejemplo para el rango de valores de intensidad de [0,15], el conjunto V puede ser cualquier subconjunto de estos 16 valores. Consideramos 3 tipos de adyacencias:

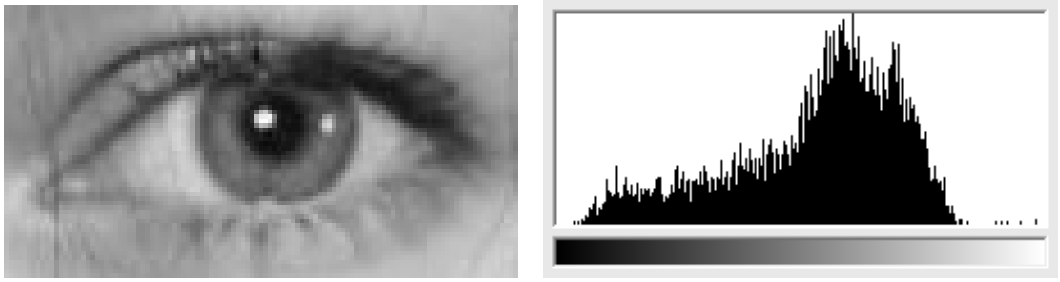
1. **4-adyacencia**, dos píxeles p y q con valores que pertenecen a V son 4-adyacentes si q está en el conjunto $N_4(p)$.
2. **8-adyacencia**, dos píxeles p y q con valores que pertenecen a V son 8-adyacentes si q está en el conjunto $N_8(p)$.
3. **m-adyacencia** (adyacencia mixta), dos píxeles p y q con valores que pertenecen a V son m-adyacentes si
   a) q pertenece a $N_4(p)$ o
   b) q pertenece a $N_D(p)$ y el conjunto $N_4(p) \cap N_D(p)$ no tiene un píxel cuyo valor pertenezca a V.

Un **camino** desde un píxel p con coordenadas (x,y) a un píxel q con coordenadas (s,t) es una secuencia de píxeles distintos con coordenadas

$$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$$

donde $(x_0, y_0) = (x, y), (x_n, y_n) = (s, t)$ y los píxeles (x_i, y_i) y (x_{i-1}, y_{i-1}) son adyacentes para $1 \leq i \leq n$. En este caso n es la longitud del camino. Si $(x_0, y_0) = (x_n, y_n)$, es un **camino cerrado**.

Sea S un subconjunto de píxeles de una imagen. Dos píxeles p y q están conectado en S, sí existe un camino entre ellos que consista solamente de píxeles de S. Para cualquier píxel p en S, el conjunto de píxeles que están conectado a él en S, se llama una **componente conexa** de S. Si solamente tiene una componente conexa, entonces el conjunto S se llama **conjunto conexo**.



(a) Imagen (b) Histograma

Figura 2.2: Histograma de una Imagen

Sea R un subconjunto de píxeles de una imagen. Llamamos a R una **región** de la imagen, si R es un conjunto conexo. Dos regiones R_i y R_j son adyacentes si su unión forma un conjunto conexo. Las regiones que no son adyacentes se llaman **disjuntas**. Solo se consideran la 4-adyacencia y la 8-adyacencia cuando se refieren a regiones.

Supongamos que una imagen contiene K regiones disjuntas, R_k $k = \{1,2,...,K\}$, ninguna toca el borde de la imagen. Sea R_u la unión de todos las K regiones y sea $(R_u)^c$ su complemento. Llamamos a todos los puntos en R_u primer plano y a todos los puntos en $(R_u)^c$ el fondo de una imagen.

La **frontera** de una región R es el conjunto puntos que son adyacentes a puntos en el complemento de R. Esta definición a veces es conocida como el **borde interno** de la región para distinguirlo del **borde externo**, el cual es el borde correspondiente al fondo.

2.1.5. Histograma de una imagen

Un **histograma** es una representación gráfica de una distribución de frecuencias.

Un **histograma de una imagen** es un histograma de los distintos valores de gris de una imagen. Si la imagen es multicanal, se puede obtener un histograma de cada canal por separado.[26]

```
Algoritmo. Cálculo de un histograma.

Entrada. A: imagen de ancho x alto
Salida. Histograma: arreglo [0,...,255] de entero
Algoritmo:
        Histograma[]:= 0
        para y:= 0, ..., alto-1 hacer
                para x:= 0, ..., ancho-1 hacer
                        Histograma[A(x,y)]:= Histograma[A(x,y)]+1
```

Los histogramas son una herramienta importante en el análisis de imágenes, ya que nos ayudan a decidir cuál es el procesamiento adecuado para mejorar la calidad de una imagen, tanto cualitativamente (qué operación aplicar) como cuantitativamente (en qué cantidad). En la Figura 2.3 se tiene un ejemplo de como se puede usar un histograma para mejorar una imagen.

En principio, una buena imagen debe producir un histograma más o menos uniforme y distribuido en todo el rango de valores.

(a) Imagen (b) Histograma

Figura 2.3: Ejemplo de una imagen con mucho brillo.

2.1.6. Principales tipos de procesamientos de imágenes

Operaciones de procesamiento global, cada píxel es tratado de forma independiente, ya sea con una o con varias imágenes.

Transformaciones locales, se considera la vecindad local de los píxeles.

Transformaciones geométricas, se modifica el tamaño y forma de la imagen.

Transformaciones lineales, Fourier, wavelets, etc.

2.1.7. Operaciones de procesamiento global

Una operación de procesamiento global, es aquella que involucra una o más imágenes, donde la operación se realiza píxel por píxel. Los tipos de operaciones de procesamiento global son, las aritméticas, lógicas y otras transformaciones generales[26].

- **Aritméticas**, sumar, restar, multiplicar, máximo, etc.
  - Unarias, una sola imagen y un valor constante.
  - Binarias, con dos imágenes.
- **Lógicas**, and, or, xor, etc.
  - Unarias, una sola imagen y un valor constante.
  - Binarias, con dos imágenes.
- **Otras transformaciones generales**
  - Transformaciones de histograma.
  - Transformaciones de color.
  - Binarización, etc.

Las operaciones de procesamiento global, las podemos expresar matemáticamente como una función, que tiene una imagen de entrada A y una imagen resultante R de la siguiente manera.

$$R(x, y) = f(A(x, y))$$

Por ejemplo, sumar un valor constante, $R(x, y) = 100 + A(x, y)$, la cual se muestra en la figura 2.4.

Figura 2.4: Ejemplo de una operación de suma, la imagen de la izquierda es la imagen original, la de la derecha es la resultante de sumarle 100 a la original.

Transformaciones de histograma

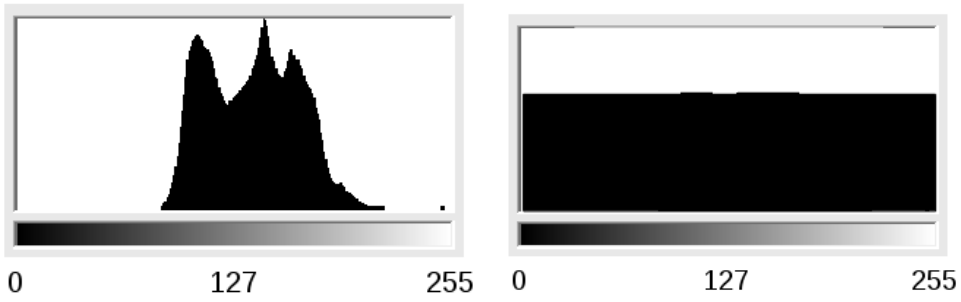
Ajuste lineal o estiramiento, es una transformación que expande el rango de los niveles de intensidad de una imagen, para abarcar el rango completo de intensidad. Esta transformación se realiza en 3 pasos.

1. Buscar el valor mínimo m del histograma .
2. Buscar el valor máximo M del histograma .
3. Aplicar la transformación $f(v) = (v - m) * maxI/(M - m)$, donde maxI es el máximo nivel de intensidad, v son los niveles de intensidad, que pertenecen al rango [0,maxI].

Ecualización del histograma, es una transformación definida de forma que el histograma resultante se distribuye uniformemente en todo el rango de grises.

```
Algoritmo. Cálculo de la función de ecualización del histograma.
Entrada. Histograma: arreglo [0,...,255] de entero
```

Figura 2.5: Estiramiento de un histograma.



(a) Imagen (b) Histograma

Figura 2.6: Ecualización de un histograma.

```
                                    np: entero (número total de píxeles = mx*my)
Salida. f: arreglo [0,...,255] de byte

Algoritmo:
        f[0]:= 0
        acumulado:= Histograma[0]
        para i:= 1, ..., 254 hacer
                f[i]:= acumulado*255/np
                acumulado:= acumulado + Histograma[i]
        finpara
        f[255]:= 255
```

Umbralización de imágenes

Es una técnica que nos permite separar los objetos de una imagen que nos interesen del resto, a esto se le conoce como segmentación. La umbralización de imágenes nos permite binarizar una imagen, es decir, construir dos segmentos. La asignación de un píxel a uno de los dos segmentos (0 y 1) se consigue comparando su nivel de gris g con un cierto valor umbral preestablecido t. La imagen final es muy sencilla de calcular ya que para cada píxel sólo hay que realizar una comparación numérica. La regla de cálculo correspondiente T_g es:

$$T_{global}(A) = \begin{cases} 0, & \text{si } g < t \\ 1, & \text{si } g \geq t \end{cases} \tag{2.1}$$

La umbralización son métodos de segmentación completos, es decir cada píxel pertenece obligatoriamente a un segmento y sólo uno.

Transformaciones de color

Una transformación a color, es aquella donde la transformación no necesariamente es igual para todos los canales (R,G,B).

$$R(x,y).R = f1(A(x,y).R, A(x,y).G, A(x,y).B)$$
$$R(x,y).G = f2(A(x,y).R, A(x,y).G, A(x,y).B)$$
$$R(x,y).B = f3(A(x,y).R, A(x,y).G, A(x,y).B)$$

Estas transformaciones permiten aplicar las misma operaciones explicadas anteriormente, pero con distintos parámetros para cada canal. Además permiten cambiar el modelo de color y aplicar la función en ese modelo. Entre algunas de las transformaciones de color tenemos:

- **Escala de grises sencilla**, $R(x,y) := (A(x,y).R + A(x,y).G + A(x,y).B)/3$.
- **Escala de grises real**, $R(x,y) := 0{,}21A(x,y).R + 0{,}72A(x,y).G0{,}07A(x,y).B$.
- **Escala de color**, a partir de una imagen en escala de gris convertirla en una imagen en escala de cierto color dado.
- **Color falso**, es una transformación cuyo objetivo es hacer más visibles las pequeñas variaciones del nivel de gris.

2.1.8. Transformaciones Locales

Una transformación local es aquella en donde el valor de un píxel depende de la vecindad local de ese píxel. Estas transformaciones tienen sentido porque existe una relación de vecindad entre los píxeles de una imagen [26].

Un tipo interesante de transformación local son las convoluciones discretas. Una **convolución discreta** es una transformación en la que el valor del píxel resultante es una combinación lineal de los valores de los píxeles vecinos en la imagen. Ejemplo

$$R(x,y) = \frac{1}{4} * A(x-1,y-1) + \frac{1}{4} * A(x,y-1) + \frac{1}{4} * A(x-1,y) + \frac{1}{4} * A(x,y)$$

Otra forma de ver la convolución, es como una matriz de coeficientes de la combinación lineal.

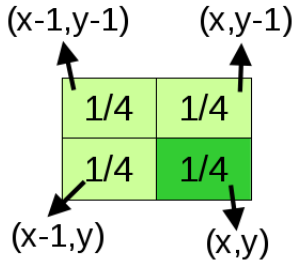


Figura 2.7: Ejemplo de una máscara de convolución

La matriz de coeficientes es conocida como **la máscara o núcleo (kernel) de convolución**. Para aplicar una máscara de convolución se aplica el siguiente algoritmo.

Sea M una máscara de convolución.
Se puede definir como array [−k...k, −p...p] de real

Algoritmo. Cálculo de una convolución.
Denotamos la convolución como: R:= M.A
Entrada. A: imagen de maxX x maxY
M: array [−k...k, −p...p] de real
Salida. R: imagen de maxX x maxY
Algoritmo:
para cada píxel (x, y) de la imagen A hacer
$R(x,y) = \sum_{i=-k}^{k} \sum_{j=-p}^{p} M(i,j) * A(x+i, y+j)$

Sobre una imagen se pueden aplicar sucesivas operaciones de convolución, $M3 \times (M2 \times M1)$. La combinación de convoluciones es equivalente a una sola convolución.

$$M2 \times (M1 \times A) = M \times A$$

Análogamente, algunas convoluciones se pueden obtener combinando otras mas simples, **núcleos separables**.

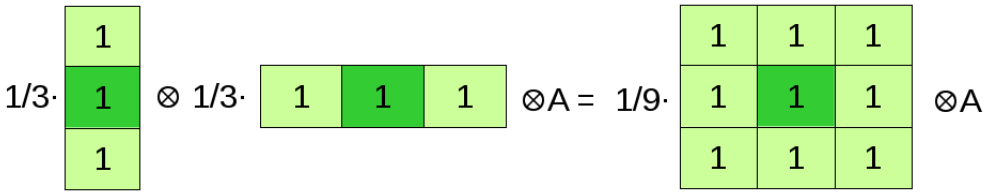


Figura 2.8: Ejemplo de un núcleo separable

Propiedades de una convolución

1. **Asociativa**, $M2 \otimes (M1 \otimes A) = (M2 \otimes M1) \otimes A$.
2. **Conmutativa**, $M2 \otimes M1 \otimes A = M1 \otimes M2 \otimes A$.
3. **Distributiva**, $A \otimes (M1 + M2) = (A \otimes M1) + (A \otimes M2)$.

Operadores importantes

Aplicando distintos operadores de convolución es posible obtener diferentes efectos:

- **Suavizado o difuminación de la imagen**, reducir contrastes abruptos en la imagen.
- **Perfilado**, resaltar los contrastes, lo contrario al suavizado.
- **Bordes**, detectar zonas de variación en la imagen.
- **Detección de cierto tipo de características**, como esquinas, segmentos, etc.

Operadores de suavizado

Media, es el operador más simple de suavizado, sus parámetros son el ancho y el alto de la región en la que se aplica y la posición del centro. Normalmente, el ancho y el alto son impares y el centro es el píxel central [26].

En algunos casos puede ser interesante aplicarlo en alguna dirección solamente, horizontal, vertical o en cualquier dirección.

1	1	1
1	1	1
1	1	1

Figura 2.9: Máscara de media aritmética de 3x3.

Gaussiano, es un operador de media ponderada, donde los pesos toman la forma de una campana de Gauss. Normalmente el suavizado gaussiano se aplica en dos dimensiones. Los pesos de la máscara dependen de la distancia al píxel central.[26]

Para 1D, la función de la campana de Gauss es:

$$f(x) = e^{\frac{-x^2}{s^2}}$$

Para 2D, la función de la campana de Gauss es:

$$f(x,y) = e^{\frac{-(x^2+y^2)}{s^2}}$$

La varianza, s^2, indica el nivel de suavizado. Se mide en píxeles.

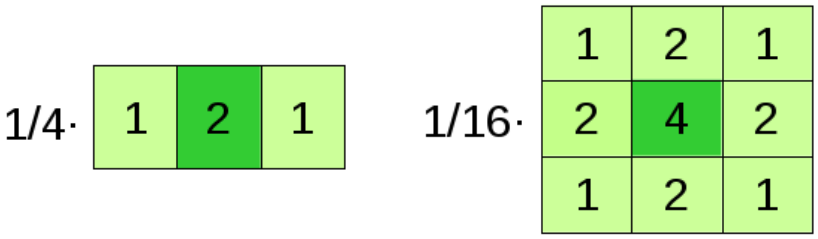
- **Varianza grande**, campana más ancha, más suavizado.
- **Varianza pequeña**, campana más estrecha, menos suavizado.

Para calcular la máscara gaussiana se debe, calcular la función correspondiente (1D o 2D), discretizar en el rango, discretizar en el valor y calcular el multiplicador. Una forma mas expedita de calcular la máscara en 1D es a través del triángulo de Pascal, donde las filas del triangulo forman discretizaciones de la campana de Gauss.[26]

Una propiedad interesante de este filtro es, que es separable. Se puede obtener un suavizado 2D aplicando dos máscaras gaussianas unidimensionales, una horizontal y otra vertical, como se puede apreciar en la figura 2.11.

Operadores de Borde

El principal objetivo de estos operadores es resaltar las transiciones de intensidad. Los usos de los operadores de borde son diversos e incluyen aplicaciones desde impresión electrónica e imágenes médicas hasta inspecciones industriales[26].



(a) Máscara de 1x3 (b) Máscara de 3x3

Figura 2.10: Ejemplo de máscaras gaussianas

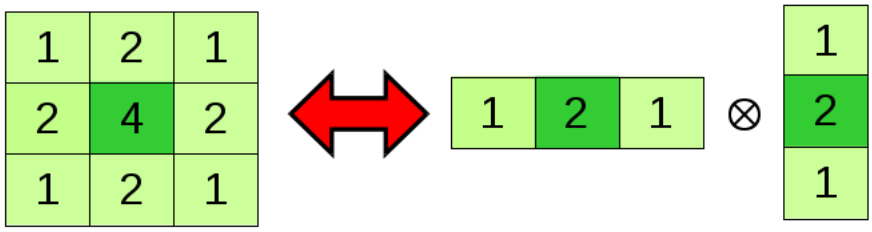


Figura 2.11: Suavizado Gaussiano usando 2 máscaras 1D.

En la sección anterior, se mostró que el suavizado de una imagen, puede ser logrado a través de un promedio de los píxeles en una vecindad. Dado que el promedio es análogo a una integral. Es lógico concluir, que los operadores de borde pueden realizarse a través de la diferenciación de los píxeles. Fundamentalmente, la magnitud de la respuesta del operador, es proporcional al grado de discontinuidad de la intensidad de la imagen en algún punto en donde el operador es aplicado. De este modo, los operadores de borde resaltan los bordes y otras discontinuidades (como el ruido) y remueven importancia a las áreas con leves variaciones de intensidad[26].

Las derivadas de una función digital están definidas en términos de diferencias. Existen varias formas de definir estas diferencias. Sin embargo, se requiere que cualquier definición que se use para la primera derivada, cumpla con lo siguiente:

1. Debe ser cero en áreas de intensidad constante.
2. Debe ser distinto de cero al inicio de una variación de intensidad.
3. Debe ser diferente de cero a lo largo de una curva.

De manera similar, cualquier definición que se use para la segunda derivada debe cumplir con lo siguiente:

1. Debe ser cero en áreas de intensidad constante.
2. Debe ser distinto de cero al inicio y al final de una variación de intensidad.
3. Debe ser cero a lo largo de curvas de pendientes constante.

Una definición básica de la derivada de primer orden de una función f(x) es la diferencia.

(a) Media de 11x11 (b) Gaussiano de 21x21

Figura 2.12: Comparación entre suavizado de media y gaussiano.

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Se define la derivada de segundo orden de $f(x)$ como la diferencia

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

En el caso de las funciones bidimensionales $f(x, y)$, la derivada de la función es un vector que apunta en la dirección de máxima variación de $f(x, y)$ y cuyo módulo es proporcional a dicha variación, a este vector se le denomina gradiente.

$$\nabla f(x,y) = (\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y})$$

Su magnitud se calcula como:

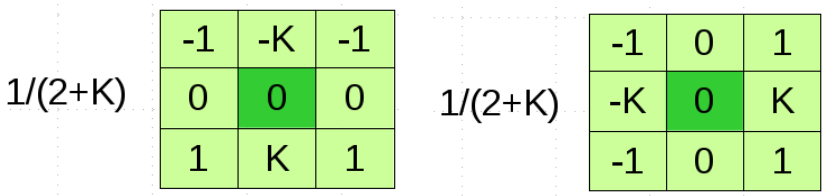
$$|f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$$

Su dirección se calcula como:

$$\theta = tan^{-1}\left(\frac{\frac{\partial f(x,y)}{\partial x}}{\frac{\partial f(x,y)}{\partial y}}\right)$$

Existen algunos operadores de borde estándar, tales como, Prewitt, Sobel, Frei-Chen y Laplace.

- Prewitt, Sobel y Frei-Chen, son operadores basados en el gradiente de una función, se pueden describir de manera conjunta como se muestra en la Figura 2.13.



(a) Gradiente columna (b) Gradiente fila

Figura 2.13: Máscaras para los operadores Prewitt, Sobel y Frei-Chen.

En el operador Prewitt ($K = 1$) se involucran a los vecinos de filas / columnas adyacentes para disminuir el ruido, por lo tanto, proporciona una buena detección para bordes verticales y horizontales.

El operador Sobel ($K = 2$), a diferencia del operador de Prewitt, proporciona una mejor detección para los bordes diagonales.

El operador Frei-Chen ($K = \sqrt{2}$), el gradiente es el mismo para bordes verticales, horizontales y diagonales.

- Laplace, es un operador isotrópico (invariante a las rotaciones) basado en la segunda derivada de una función. El laplaciano de una imagen resalta las regiones con cambios rápidos de intensidad. Este operador generalmente es aplicado luego de que la imagen ha sido suavizada para disminuir el ruido. El operador de laplace se define como:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

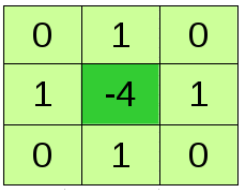


Figura 2.14: Máscara de Laplace

Detección de esquinas

Una esquina en una imagen A, es un píxel p tal que dos lados con diferentes direcciones se intersectan. También puede definirse como un punto para el que hay dos direcciones de bordes dominantes y diferentes en una vecindad local del punto. Entre los detectores de esquinas mas comunes están el de Harris y Stephens y FAST[7, 11], como se muestra en la Figura 2.15

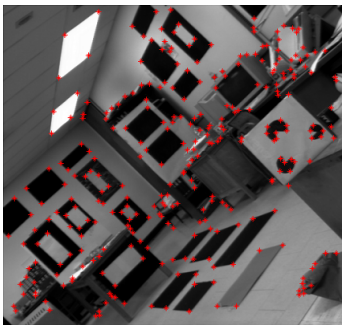


Figura 2.15: Esquinas de una imagen luego de aplicar un detector de esquinas FAST.

El **detector de Harris y Stephens**, también conocido como detector de Harris, es un operador que utiliza las derivadas de primer orden de una imagen A, suavizada con el operador de Gauss(L).

$$G(p,\sigma) = \begin{bmatrix} L_x^2(p,\sigma) & L_x(p,\sigma)L_y(p,\sigma) \\ L_x(p,\sigma)L_y(p,\sigma) & L_y^2(p,\sigma) \end{bmatrix}$$

Para un píxel p, los autovalores λ_1, λ_2 de la matriz G, representan los cambios de intensidad en direcciones ortogonales en la imagen A. En vez de calcular estos autovalores, se considera el siguiente indicador "cornerness measure".

$$H(p,\sigma,a) = det(G) - a \cdot Tr(G)$$

Con un parámetro a pequeño y mayor a 0.

El **detector FAST** (*feature from accelerated segment test*), es un operador que identifica una esquina, considerando los valores de la imagen en un círculo digital alrededor del píxel p. Para ser considerado una esquina, el valor del píxel central necesita ser más oscuro (o claro) comparado con más de 8 píxeles subsecuentes en este círculo (11 para realmente identificar una esquina y no solamente un píxel irregular en un borde recto) y "similar.ª los valores de los píxeles restante en el círculo[7, 27, 28].

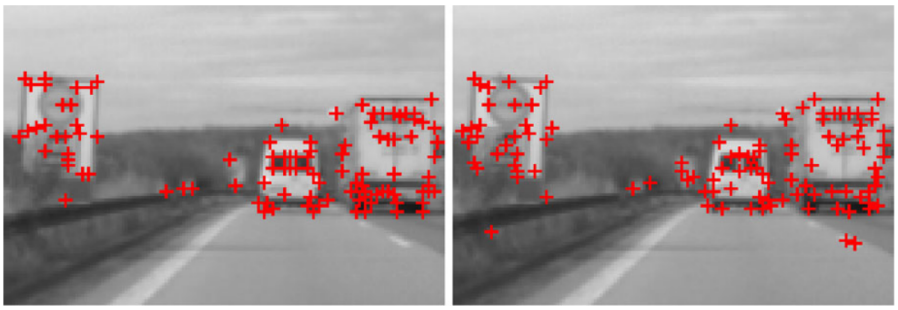


Figura 2.16: Comparación entre el detector de Harris (izquierda) y FAST (derecha)

2.1.9. Transformaciones Geométricas

Una transformación geométrica, es aquella donde el valor del píxel depende de otro píxel (o varios) cuya posición es calculada a través de un par de funciones f_1 y f_2. El tamaño de la imagen de salida no necesariamente corresponde al de la imagen de entrada [26].

$$R(x,y) = A(f_1(x,y), f_2(x,y))$$
$$f_1, f_2 : NxN \rightarrow R$$

Donde f_1, es la posición X de la imagen original para el píxel resultante (x,y). Y f_2, es la posición Y de la imagen original para el píxel resultante (x,y).

Entre las transformaciones geométricas tenemos, las básicas, las afines, la bilineal y la perspectiva.

Las **transformaciones básicas** están conformadas por las transformaciones de desplazamiento y recorte, reflejos y rotaciones exactas (90, 180, 270).

Las **transformaciones afines**, están conformadas por las transformaciones de identidad, escalado, rotación, traslación, inclinación (vertical y horizontal). Estas transformaciones permiten establecer una correspondencia entre dos quadrilateros y conservan el paralelismo de las líneas. Todas estas transformaciones se pueden expresar de forma matricial, de la siguiente manera:

$$R(x,y,1) = A\left(\begin{bmatrix} c_11 & c_12 & c_13 \\ c_21 & c_22 & c_23 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right)$$

En la Tabla 2.1 se muestra las distintas matrices de transformación para cada tipo junto con un ejemplo. La aplicación más inmediata y frecuente de las transformaciones afines es extraer y redimensionar un área de interés, dándole una forma predefinida de antemano, a esta aplicación se le conoce con el nombre de normalización.

Las **transformaciones bilineal y perspectiva**, son una generalización de las transformaciones afines. Estas transformaciones establecen una correspondencia entre líneas pero sin preservar el paralelismo y además conservan los radios cruzados de cuatro puntos colineales[26]. Las transformaciones bilineal y de perspectiva se definen de la siguiente manera:

- Transformación Bilineal

$$R(x,y) = A\left(\begin{bmatrix} c_11 & c_12 & c_13 & c_14 \\ c_21 & c22 & c_23 & c_24 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ x*y \\ 1 \end{bmatrix}\right)$$

- Transformación Perspectiva

$$R(x,y) = A(x'/z', y'/z'), \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} c_11 & c_12 & c_13 \\ c_21 & c_22 & c_23 \\ c_31 & c_32 & c_33 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

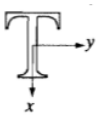
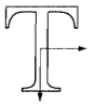
Nombre Transformación	**Matriz Afín**	**Ejemplo**
Identidad	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Escala	$\begin{bmatrix} e_x & 0 & 0 \\ 0 & e_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Rotación	$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Traslación	$\begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$	
Inclinación Vertical	$\begin{bmatrix} 1 & 0 & 0 \\ -i_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Inclinación Horizontal	$\begin{bmatrix} 1 & -i_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	

Tabla 2.1: Matrices de para las diferentes transformaciones afines.[26]

(a) (b)

Figura 2.17: Ejemplo de una transformación bilineal(izquierda) y perspectiva(derecha).

2.2. Visión por Computador

La visión por computador es un campo de estudio dentro del procesamiento digital de imágenes, que tiene como objetivo adquirir, procesar, analizar y comprender imágenes a fin de producir información numérica o simbólica [17].

El rango de aplicaciones va desde los sistemas de visión industriales, los cuales inspeccionan las botellas en una línea de producción, hasta la investigación en inteligencia artificial y computadoras que pueden entender el mundo a su alrededor. Algunos ejemplos de sistemas de visión por computadora son:

- Procesos de control, ej. un robot industrial.
- Navegación, ej. un vehículo autónomo o robot móvil.
- Detección de eventos, ej. conteo de personas.
- Organización de información, ej. bases de datos para indexar imágenes o secuencia de imágenes.
- Modelado de objetos o ambientes, ej. análisis de imágenes médicas o modela topológico.
- Inspección Automática, ej. aplicaciones de manufactura.

2.2.1. Tareas comunes

Cada una de las aplicaciones descritas anteriormente emplean un conjunto de tareas de visión por computador, problemas de medición o procesamiento más o menos bien definidos, los cuales pueden ser solucionados usando una variedad de métodos. Algunas de las tareas utilizadas usualmente son el reconocimiento, análisis de movimiento, reconstrucción de escenas y restauración de una imagen[7].

Reconocimiento

La tarea de reconocimiento, consiste en determinar si la imagen contiene o no algún objeto, característica o actividad específica. Existen diferentes variedades de reconocimiento, entre las cuales están:

- **Reconocimiento de objetos o clasificador de objetos**, es el reconocimiento de uno o varios objetos aprendidos o especificados previamente, en conjunto con su posición 2D en la imagen o su pose 3D dentro de la escena.
- **Identificación**, es el reconocimiento de una instancia individual de un objeto. Ej. identificación de la cara de una persona específica.
- **Detección**, los datos de las imágenes se digitalizan para una condición específica. Ej. detección de posibles células anormales.

Análisis de Movimiento

La tarea de análisis de movimiento, se define como la estimación del movimiento a partir de una secuencia de imágenes para producir un estimado de la velocidad, ya sea en cada punto de la imagen o en una escena 3D, o de la cámara que genera las imágenes. Ejemplos de esta tarea se tienen:

- **Localización**, es la estimación del movimiento rígido 3D (rotación y traslación) de la cámara a partir de una secuencia de imágenes, producidas por la cámara.
- **Rastreo**, es el seguimiento de los movimientos de un pequeño conjunto de puntos de interés u objetos en la secuencia de imágenes.
- **Flujo Óptico**, es el cálculo para cada punto de la imagen, de cómo ese punto se mueve con respecto al plano de la imagen.

Reconstrucción de escenas

La tarea de reconstrucción de escenas, es dada una o (generalmente) más imágenes de una escena, o un vídeo, el objetivo de esta tarea es producir un modelo 3D de la escena. En el caso más simple, el modelo puede ser una nube de puntos 3D. Los métodos más sofisticados producen una superficie 3D completa del modelo.

Restauración de imágenes

La tarea de restauración de imágenes, consiste en la eliminación del ruido de las imágenes. El enfoque más sencillo para la eliminación del ruido es usando diferentes tipos de filtros como los de paso bajo o el de mediana. Los métodos más avanzados asumen un modelo de como se debería ver las estructuras locales de la imagen, la cual las distingue del ruido. Primero analizando los datos de la imagen en términos de las estructuras locales de la imagen, como líneas o bordes, y luego controlando el filtrado de acuerdo a la información local del paso anterior, con lo que se logra un mejor resultado que los métodos más sencillos.

2.2.2. Arquitectura

La arquitectura de un sistema de visión por computador es altamente dependiente de la aplicación. Algunos sistemas son independientes que resuelven un problema específico de medición o de detección, mientras otros constituyen un subsistema de un diseño más grande. La implementación específica de un sistema de visión por computador también depende de si su funcionalidad está predefinida o alguna parte de ella se puede aprender o modificar durante su uso. Mucha de sus funciones son únicas según la aplicación.[17, 7] Sin embargo, existen funciones que se encuentran en muchos sistemas.

Adquisición de una imagen, generación de una imagen digital a través de un o más sensores como las cámaras, tomógrafos, radares, cámaras de ultrasonido, etc. Dependiendo del tipo de sensor, los datos pueden ser imágenes ordinarias 2D, un volumen 3D o una secuencia de imágenes.

Preprocesamiento, preprocesamiento de la data antes de ser procesada por un método de visión para el computador para garantizar que se satisfagan ciertas asunciones asumidas por el método. Ejemplos:

- Remuestreo, a fin de garantizar que el sistema de coordenadas de la imagen es correcto.
- Reducción del ruido, para garantizar que el ruido de los sensores no introduzcan información erronea.
- Mejora del contraste, con el fin de garantizar que la información relevante pueda ser detectada.

Extracción de características, extracción de características en distintos niveles de complejidad de la imagen. Ejemplos:

- Líneas, bordes y picos.
- Puntos de interés locales como esquinas, regiones o puntos.
- Otras características más complejas relacionadas con texturas, forma o movimiento.

Detección/Segmentación, decidir que puntos o regiones de interés de una imagen son relevantes para un procesamiento posterior. Ejemplos:

- Selección de un conjunto específicos de puntos de interés.
- Segmentación de una o más regiones de interés en una imagen que contienen un objeto específico de interés.
- Segmentación de una imagen en una escena jerárquica compuesta por primer plano, grupo de objetos.

Procesamiento de alto nivel, procesamiento de un pequeño (generalmente) conjunto de datos, como por ejemplo, conjunto de puntos o una región que contiene un objeto específico. Ejemplos:

- Estimación de parámetros específicos de la aplicación, tales como la pose o dimensiones de un objeto.
- Reconocimiento de una imagen, clasificar un objeto detectado en diferentes categorías.
- Alineación de una imagen, comparar y combinar dos vistas diferentes de un mismo objeto.

Toma de decisiones, realiza la toma de decisiones requerida por una aplicación. Por ejemplo:

- Aprobar/Rechazar en aplicaciones de inspección automatizadas.
- Corresponde/No corresponde en aplicaciones de reconocimiento.
- Marcar para una revisión humana posterior en aplicaciones de reconocimiento, médico, militar y seguridad.

2.2.3. Detección y Descripción de Características

Características de una imagen

Una **característica**, es una parte interesante de una imagen, que su exacta definición de que cons-tituye una característica, depende generalmente del problema a tratar o del tipo de aplicación. Las características son usadas como punto de partida y como primitivas para otros algoritmos[17].

Tipos de Características

Como se mencionó anteriormente, existen diferentes tipos de características en una imagen, las más comunes en los sistemas de visión de computador y procesamiento de imágenes digitales están los bordes, las esquinas y las regiones.

Los **bordes**, son puntos donde hay una frontera (o arista) entre dos regiones de una imagen. En general, un borde puede ser de cualquier forma arbitraria y puede incluir uniones.

Las **esquinas o puntos de interés**, son las características de una imagen que son parecidas a un punto, que tienen una estructura local bidimensional.

Las **regiones**, son estructuras de una imagen que son parecidas a una región, contrario a las esquinas que son parecidas a un punto. Los descriptores de regiones generalmente pueden contener un punto preferido (un punto máximo local o centro de gravedad), por lo que también pueden ser identificados como operadores de puntos de interés.

(a) (b) (c)

Figura 2.18: Tipos de características, (a)bordes, (b)esquinas y (c)regiones.

Detección de Características

La detección de características en los sistemas de visión por computadora, se refiere a los algoritmos que tienen como objetivo, calcular abstracciones de la información de una imagen y realizar decisiones locales en cada píxel de si existe o no una característica relevante de cierto tipo. El conjunto de características resultante sera un subconjunto del dominio de la imagen, generalmente en forma de puntos aislados, curvas continuas o regiones conectadas[17].

Los algoritmos para detección de características son operaciones de procesamiento de imágenes de bajo nivel. Eso significa, que usualmente es la primera operación que se ejecuta sobre una imagen, y examina cada píxel para ver si esta presente una característica en ese píxel.

La fase de detección de características esta ligada al desempeño general del sistema, el cual, ge-neralmente será tan bueno como su detector de características. En consecuencia, la propiedad más deseada en un detector de características es su repetibilidad, si una misma característica será detectada en dos o más imágenes de la misma escena o no.

Descripción de Características

Una vez que las características han sido detectadas, es necesario extraerlas. El resultado de esta operación se le conoce con el nombre de vector de característico o descriptor característico.[17]

Un **descriptor** x, es un punto en un espacio n-dimensional R^n, llamado espacio descriptor. Este espacio representa los valores de las propiedades medidas o calculadas en un orden dado. Este descriptor una vez que ha sido extraído de una imagen de entrenamiento, puede ser usado para identificar el objeto cuando se intente localizar el objeto en una imagen que contiene otros objetos. Es importante que las características extraídas de la imagen de entrenamiento pueden ser detectados aún cuando ocurran cambios en la imagen, tales como en la iluminación, ruido, pose, etc.

Existen una gran gamma de métodos para extraer y describir las características de una imagen, cada uno con distintas ventajas y desventajas. Uno de los descriptores más usado en la actualidad es el SIFT (*scale invariant feature transform*), pero también existen otros como el SURF (*speedup robust features*), el BRIEF (*Binary robust independent elementary features*), el ORB (*oriented FAST and rotated BRIEF*), entre otros[17, 20, 14, 18, 1, 2].

Los descriptores se pueden agrupar según la forma como se calculan y representan, los dos grupos mas representativos son los descriptores basados en HOG (Histogramas de Gradientes Orientados) y los descriptores binarios.

Los **descriptores HOG**, son aquellos que comparan la vecindad de un punto de interés con otra, a través de vectores, estos vectores se generan a partir de la concatenación de varios histogramas de vectores orientados en direcciones predefinidas. Entre los descriptores que pertenecen a este grupo estan el SIFT y el SURF[20, 14, 18].

Los **descriptores binarios**, son aquellos que compararan la vecindad de un punto de interés con otra, a través de string binarios y usando generalmente la distancia Hamming entre ambos strings. Estos descriptores están compuestos de tres partes, áreas de muestreo (representan las secciones de donde se extraerá la información), calculo de la orientación (fase para asegurar que el descriptor sea invariante a la rotación) y un conjunto de parejas de muestreo (el conjunto de parejas a formar

con los puntos extraídos de las áreas de muestreo). Entre los descriptores que pertenecen a este grupo están el BRIEF y el ORB[1, 2].

En este trabajo el enfoque será en el descriptor ORB, pero antes es necesario hablar sobre el descriptor BRIEF que fue el primer descriptor binario publicado, en el año 2010 y es la base del descriptor ORB.

El **descriptor BRIEF**, es un string binario de longitud N, que se calcula sobre un parche P de tamaño SxS, primero se requiere aplicar un filtro de suavizado sobre el parche, luego determinar N pares únicos de píxeles (Xi, Yi), donde Xi y Yi pertenecen al parche P. Se evalúa cada pareja y si la intensidad del píxel en la posición Xi es menor que la de Yi, se coloca un 1 sino 0. Y el string final sera la concatenación de todos los resultados. La longitud recomendada por los autores es de 256 bits [1].

Existen varias estrategias para generar el conjunto de parejas de píxeles requeridos definidas por los autores de BRIEF en su trabajo, pero la que ellos recomiendan es obtener los píxeles de manera aleatoria usando una distribución gaussiana. Para realizar el matching entre dos descriptores solo es necesario calcular su distancia Hamming con la operación XOR, con lo que lo hace que sea un descriptor bastante rápido de calcular en comparación con los descriptores HOG[1].

El **descriptor ORB**, es un descriptor basado en BRIEF de longitud 256 y que además es invariante a la rotación y al escalamiento, publicado en el año 2011[2]. Para obtener invarianza a la rotación, se calcula la orientación del parche usando como medida la intensidad del centroide [2]. Una vez que se obtiene en ángulo de orientación, se puede rotar el parche a unos ángulos predefinidos, con lo que se logra la invarianza a la rotación [2].

Otra de las diferencias entre ORB y BRIEF, es la estrategia usada para generar el conjunto de pares de píxeles. En BRIEF, la generación del conjunto se hace de manera aleatoria usando una distribución gaussiana, en cambio, los autores de ORB identificaron cual es el conjunto que mejor representa la información. Ellos calcularon que existen aproximadamente 250000 posibles parejas para ser consideradas. De todas las posibilidades se requiere seleccionar solamente 256 de ellas, de tal manera que cada pareja aporte nueva información al descriptor. Para la selección de las 256 parejas, se uso un conjunto de entrenamiento tomados del dataset PASCAL 2006 y posteriormente se aplico un algoritmo voraz, obteniendo como resultado el conjunto de parejas que se deben evaluar[2].

2.3. Realidad Aumentada

La Realidad Aumentada (RA) es una vista de un entorno del mundo real, a través de un dispositivo tecnológico de forma directa o indirecta, cuyos elementos se combinan con estímulos virtuales generados por un computador, tales como sonido, vídeo o gráficos, para la creación de una realidad mixta en tiempo real.[3]

Las dos definiciones de Realidad Aumentada más aceptadas hoy en día son la de Azuma[4] y la de Paul Milgram y Fumio Kishino[**3**]. La definición de Azuma establece que un sistema de realidad aumentada, es aquel que cumple con las siguientes 3 características:

1. Combina elementos virtuales y reales.
2. Es interactiva en tiempo real.
3. Registra (alinea) objetos reales con los virtuales y viceversa.

Y la de Paul Milgran y Fumio Kishino, que definen la realidad de Milgram-Virtuality Continuum, que es la relación existente entre un entorno real puro y un entorno virtual puro, y en el medio se encuentra un entorno de Realidad Aumentada (más cerca del entorno real) y Virtualidad Aumentada (está más cerca del entorno virtual) [**3**].

Figura 2.19: Milgram-Virtuality Continuum [**3**]

2.3.1. Historia

Entre algunos eventos importantes en el mundo de la realidad aumentada podemos nombrar los siguientes[3]:

1962: Morton Heilig, un director de fotografía, crea un simulador de moto llamado Sensorama con imágenes, sonido, vibración y olfato.

1973: Ivan Sutherland inventa la display de cabeza (HMD) lo que sugiere una ventana a un mundo virtual.

1985: Myron Krueger crea Videoplace que permite a los usuarios interactuar con objetos virtuales por primera vez.

1992: Tom Caudell crea el término Realidad Aumentada.

1994: Steven Feiner, Blair MacIntyre y Doree Seligmann primera utilización importante de un sistema de Realidad Aumentada en un prototipo, KARMA, presentado en la conferencia de la interfaz gráfica. Ampliamente citada en la publicación Communications of the ACM al siguiente año.

1999: Hirokazu Kato desarrolla ARToolKit en el HitLab y se presenta en SIGGRAPH ese año.

2000: Bruce H. Thomas desarrolla , el primero juego al aire libre con dispositivos móviles de Realidad Aumentada, y se presenta en el International Symposium on Wearable Computers.

2009: AR Toolkit es portado a Adobe Flash (FLARToolkit) por Saqoosha, con lo que la realidad aumentada llega al navegador Web.

2.3.2. Arquitectura

Los sistemas de realidad aumentada requieren de componentes de hardware como de software, para proveer una experiencia de realidad aumentada convincente. El software le envía al sistema las acciones que debe realizar y el hardware las ejecuta.

Todos los sistemas de realidad aumentada contienen al menos tres componentes básicos de hardware, sensores, procesadores y dispositivos de salida.[8]

Sensores

Los sensores son dispositivos que adquieren información acerca del mundo real. Entre sus funciones tienen, proveer información de la posición y orientación del usuario (o sustituto del usuario como los dispositivos móviles), temperatura, pH, luminosidad/oscuridad o cualquier otra información del ambiente para posteriormente ser enviada a la aplicación [8].

Una de las funciones principales de los sensores, es proveer información acerca del mundo real, que permite a la aplicación determinar la posición y orientación de los elementos que se encuentran en el mundo real, a este funcionalidad se le llama rastreo [8].

Procesadores

El procesador es un componente esencial en un sistema de realidad aumentada. El término procesador se puede referir a una sola unidad de procesamiento o a múltiples componentes trabajando en conjunto. El procesador es el encargado de recibir las señales de los sensores, ejecutar las instrucciones de la aplicación basadas en la información del sensor y crear las señales que controlan a la(s) pantalla(s) del sistema [8].

En general, el sistema de procesamiento en una aplicación de realidad aumentada consiste primordialmente de uno o más microprocesadores de propósito general (CPU) y posiblemente una o más unidades gráficas de procesamiento (GPU).

Un número de especificaciones indican aproximadamente como un sistema de procesamiento se comportará para una aplicación dada. Algunas de las especificaciones más importantes son:

1. **Número de procesadores**, la cantidad de CPUs disponibles en el sistema.
2. **Velocidad del procesador**, es la cantidad de operaciones por segundo que puede realizar un procesador.
3. **Memoria disponible**, cantidad de memoria principal disponible.
4. **Almacenamiento disponible**, cantidad de memoria secundaria disponible para almacenar los datos, es mas lenta que la memoria principal.
5. **Tarjeta Gráfica**, es hardware especializado en realizar operaciones relacionadas con computación gráfica.
6. **Ancho de banda de la red**, tasa a la cual los datos pueden pasar en una red, es expresada en bits por segundo.
7. **Retardo de la red**, es el tiempo que tarda un mensaje en llegar desde su lugar de origen a su lugar de destino.

Dispositivos de salida

Los dispositivos de salida, son aquellos que proveen las señales que nuestros sentidos perciben, tales como señales visuales, auditivas, táctiles, olfativas y quizás gustativas. Entre los dispositivos de salida más utilizados en los sistemas de realidad aumentada tenemos, los visuales (pantallas, lentes, proyectores, HMD) , auditivos (cornetas, audífonos), táctiles (transductores) y estéreos (lentes, HMD) [8].

Existen un número de especificaciones que proveen información acerca de los diferentes tipos de dispositivos de salida y como será su desempeño para las tareas que se les asigno. Entre algunas de las comunes para todos los dispositivos son:

Fidelidad, es una medida de calidad y/o que tan bien la representación replica al mundo real equivalente.

Resolución aparente, es como tus sentidos perciben el sistema, en lugar de exactamente lo que esta mostrando el dispositivo. Por ejemplo, una pantalla grande puede verse muy detallada pero en realidad tiene un resolución baja.

Brillo, es la amplitud de la señal.

Campo visual, es el área total que puede ser capturado por un sensor estacionario..

Field of Regard, es el área total que puede ser capturado por un sensor móvil.

Propiedades logísticas, están relacionados con los requisitos de peso, tamaño, durabilidad, energía, etc de los dispositivos. Estos deben ser considerados de acuerdo a la aplicación de realidad aumentada

Sin importar la plataforma de hardware disponible, es el software, el que permite que el hardware realice lo que se necesita/desea hacer. Existen componentes de software que son parte de la infraestructura de realidad aumentada sin importar la aplicación que se utilice, hay software que son específicos de la aplicación y hay software que son usados para crear contenido para la aplicación.

El software involucrado con la creación y utilización de una aplicación de realidad aumentada se puede dividir aproximadamente en cuatro categorías:

- El software involucrado directamente con la aplicación de realidad aumentada.
- El software usado para crear la aplicación de realidad aumentada.
- El software usado para crear el contenido para la aplicación de realidad aumentada.
- Otros software relacionados con realidad aumentada.

Otra forma de conceptualizar los componentes de software para los sistemas de realidad aumentada es:

- Librerías de programación de bajo nivel (ej. software de rastreo).
- Librerías para el renderizado y construcción de la aplicación.
- Plugins para aplicaciones existentes.
- Aplicaciones independientes (creación de contenidos).
- Software para crear contenidos para la aplicación de realidad aumentada.

La relación entre los componentes de un sistema de realidad aumentada explicados anteriormente se puede ver en la Figura 2.20.

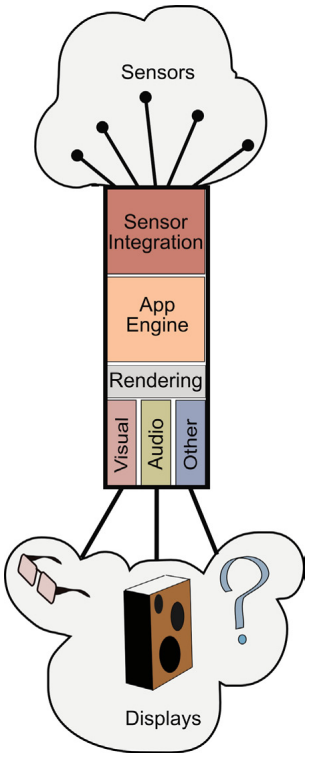


Figura 2.20: Arquitectura de un sistema de realidad aumentada [8]

2.3.3. Problemas principales

Toda aplicación de realidad aumentada tienes dos problemas que deben ser resueltos por todas las aplicaciones de realidad aumentada, estos son el rastreo y el registro[4]. En los sistemas de realidad aumentada el ambiente real ofrece un marco de referencia para el contenido. Por lo tanto, los usuarios se percatan inmediatamente de alguna discrepancia, esta discrepancia esta directamente relacionada con el rastreo y el registro. La solución a estos dos problemas deben estar alineados con los requerimientos de la aplicación, una aplicación puede requerir un rastreo preciso o robusto y/o un registro rápido o preciso.

Registro

El registro utiliza los resultados del rastreo para alinear el contenido virtual y el real el uno al otro. La alineación se debe hacer con respecto a los objetos reales y la posición de la cámara. El registro es un ciclo-abierto o un ciclo-cerrado. En un ciclo-abierto el registro se basa solamente en la información del sistema de rastreo, no utiliza ningún mecanismo de retroalimentación. En un ciclo-cerrado el registro proporciona una retroalimentación sobre la precisión de la operación, la cual es tomada en cuenta en posteriores operaciones.

Conseguir un registro correcto puede ser complicado porque el usuario ve lo que esta ocurriendo en el mundo real. Los usuarios siempre reconocen discrepancias entre el contenido virtual y los

objetos reales[4]. Para el contexto de algunas aplicaciones la precisión puede no ser una prioridad. Por ejemplo en el caso de una aplicación que permite anotaciones en las edificaciones de una cuidad, es aceptable no posicionar la anotación justamente en el centro del edificio. Sin embargo, la información no debe ser puesta en un edificio diferente.

Azuma[4] declara que un registro que sea preciso es difícil de conseguir debido a que existen diversos y muy diferentes tipos de fuentes de error. Azuma divide los errores de registro en dos amplias categorías: estática y dinámica. Los errores estáticos ocurren mientras ni la cámara ni los objetos se mueven. Los errores dinámicos aparecen cuando hay movimiento debido a los retrasos del sistema.

Rastreo

El rastreo determina la posición y orientación (pose) de la cámara dentro del entorno. El rastreo es una parte fundamental de cada aplicación de realidad aumentada. Sin la pose, el contenido virtual no puede ser posicionado en ninguna relación a los elementos reales o a la perspectiva de la cámara.[8]

Los sistemas de rastreo determinan los tres grados de libertad para la posición como para la orientación. Estos valores de posición y orientación pueden ser relativos o absolutos a su ambiente. El rastreo debe ser tan preciso, exacto y robusto como sea posible con el fin de crear la ilusión de que el contenido virtual es parte del mundo real[4]. Idealmente, un proceso de rastro calcula los resultados en tiempo real sin ser afectado por las distintas circunstancias y alrededores, por ejemplo cambio en las condiciones de iluminación.

Los sistemas de rastreo se pueden diferenciar como de ciclo-abierto o ciclo-cerrado. Lo sistemas de ciclo-cerrado son sistemas que corrigen sus errores dinámicamente a través de una retroalimentación acerca de sus resultados y en cambio, los sistemas de ciclo-abierto no poseen una retroalimentación.

Existen distintos métodos para realizar el rastreo, entre ellos tenemos, el rastreo óptico, el acústico, el electromagnético, el mecánico, etc.

El **rastreo óptico**, es una técnica que utiliza la visión por computadora para localizar objetos en un espacio , el sensor que se usa es la cámara. La cámara recoge la luz a por una lente y provee una señal que representa, lo que la cámara "ve". La imagen es luego analizada para identificar la posición y orientación de los elementos de interés.[8]

Así como existen un gran variedad de cámaras para tomar fotografías y grabar vídeos, también existen una gran gama de cámaras para el rastreo óptico en los sistemas de realidad aumentada. Las cámaras mas comunes que se usan en los sistemas de realidad aumentada son las cámaras web, la de los teléfonos inteligentes, la de las tabletas y las de propósito especial.

Las ventajas que tiene el rastreo óptico son:

- Se puede realizar sin requerir cables u otra cosa unida al objeto a rastrear.
- Se puede rastrear varios objetos simultáneamente.
- En la mayoría de los casos, no requiere alterar el mundo real, nada más en algunos casos donde se agregan algunos marcadores para que el sistema los rastree.

- Muchos dispositivos utilizados en sistemas de realidad aumentada ya poseen una cámara, como los teléfonos inteligentes.

Las desventajas que tiene el rastreo óptico son:

- Requiere de suficiente luz ambiental para poder "ver.[el] mundo real con suficiente detalle, para así poder proveer imágenes adecuadas al software de visión por computadora.
- Necesita una línea de visión libre entre la cámara y las entidades que están siendo rastreadas.
- El entorno en el que es utilizado debe proporcionar información apta para el rastreo, es decir, si el entorno es del mismo color, sin ningún otra característica disponible, el sistema no podrá determinar algo sobre la posición y orientación de los elementos en el entorno.
- Introduce retardo al sistema, ya que toma tiempo el adquirir una imagen y analizarla para extraer la información deseada.

El **rastreo acústico**, es aquel donde se usan micrófonos como sensores. Los micrófonos se adjuntan al objeto que se desea rastrear o se coloca en el entorno. Con este sistema, se necesita que exista una fuente de información acústica a ser detectada por el micrófono. Generalmente, se usa ultrasonido. El ultrasonido, es el sonido que su frecuencia es superior a las que puede percibir el oído. Los sistemas de rastreo ultrasónicos funcionan haciendo que el objeto rastreado emita un sonido, y un conjunto de micrófonos en el entorno para capturar ese sonido. Basados en el tiempo y la amplitud del sonido detectado en cada micrófono, se puede calcular la posición de origen del sonido.[8]

Las ventajas que tiene el rastreo acústico son:

- No se ven afectados por las condiciones de iluminación.

Las desventajas que tiene el rastreo acústico son:

- No puede ser usado en entornos que son ruidosos con señales de sonido con el mismo rango de frecuencia del sistema.
- Cada objeto a ser rastreado debe tener una fuente de sonido adjunta, es decir, que solo puede ser usada en ambientes donde se conoce a priori la cantidad de objetos a rastrear y que se puedan equipar con una fuente de sonido.
- Tienen un rango limitado.

El **rastreo electromagnético**, mide los campos magnéticos generados por una corriente eléctrica que pasa secuencialmente a través de tres bobinas organizados de forma perpendicular uno al otro. Cada bobina se convierte en un electroimán, y el sistema de sensores miden como su campo magnético atrae a las demás. Esta medición le dice al sistema la dirección y orientación del emisor. Un buen sistema de rastreo electromagnético es altamente sensible, con bajos niveles de retardo.[8]

Las ventajas del rastreo electromagnético son:

- No son dependientes de la luz ambiental.

Las desventajas del rastreo electromagnético son:

- Son sensibles a los metales en el entorno donde se encuentran. En consecuencia, deben ser calibrados para eliminar la interferencia de los métales en el entorno.
- Tienen un rango limitado.

- Son costosos.

El **rastreo mecánico** funciona anexando vínculos a los objetos que desea rastrear. Esos vínculos tienen sensores en cada una de las uniones que reportan el ángulo entre los vínculos. A menudo esto es hecho colocando un potenciómetro en las uniones y leyendo el nivel de voltaje. A medida que el ángulo del vínculo cambia, el valor de la resistencia del potenciómetro varía y el correspondiente cambio en voltaje ocurre. El voltaje puede luego ser usado para determinar el ángulo entro los vínculos. Esta información, en conjunto con los ángulos del resto de los vínculos en el sistema, permite calcular la posición y pose del objeto.Otros sensores que son muy utilizados en los sistemas de realidad aumentada son el acelerómetro, que mide aceleraciones y el giroscopio que mide la orientación [8].

Las ventajas del rastreo mecánico son:

- Es rápido y preciso.
- Puede integrar una fuerza de respuesta con la aplicación de realidad aumentada de acuerdo a una acción realizada.

Las desventajas del rastreo mecánico son:

- En la mayoría de los casos no es posible usarlo debido a que no se pueden usar las conexiones mecánicas requeridas, por ejemplo, si estas usando tu teléfono en un espacio abierto.
- Es visualmente molesto.
- Sólo funcionará correctamente en aplicaciones donde el componente mecánico pueda ser escondido de la vista.
- Puede ser costosa en comparación con las otras tecnologías.
- Debido a la necesidad de conectar físicamente los objetos con el sistema de rastreo mecánico, generalmente es de rango limitado.

2.3.4. Principales Tipos

Existen dos tipos principales de sistemas de realidad aumentada. Los basados en la inserción de marcadores físicos dentro de la escena para registrar los elementos virtuales y los que usan como base las características naturales del entorno. Acá nos centraremos principalmente en los sistemas que usan el rastreo óptico.[8, 29]

Basados en marcadores

En el rastreo óptico, el sistema deduce la pose de la cámara de acuerdo a las observaciones de lo que ve. En un entorno desconocido, es complicado, requiere de algo de tiempo para reunir suficientes datos para poder deducir la pose y la pose calculada empieza a desviarse con el tiempo debido a los errores acumulados.[8, 29]

Una solución para superar estos inconvenientes es insertar un elemento en el ambiente que sea fácilmente detectable usando técnicas de visión por computadora. Este elemento se llama marcador.

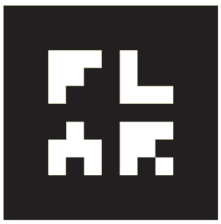


Figura 2.21: Ejemplo de un marcador

Un marcador es un símbolo o imagen que un sistema computarizado puede detectar a partir una imagen de vídeo usando técnicas de procesamiento de imágenes, reconocimiento de patrones y visión por computadora. Una vez detectado, entonces define la escala y pose correcta de la cámara.[29]

Una de sus grandes ventajas, es que los marcadores pueden ser diseñados de forma tal que garanticen ser detectados correctamente bajo cambios considerables de pose e iluminación. Su obvia desventaja es que requieren una modificación física de la escena durante el proceso de captura.

Los sistemas basados en marcadores son bastante populares debido a que son simples de implementar y también a que existen librerías buenas y reconocidas. Además, los marcadores proveen la escala correcta y marcos de referencias convenientes, pueden codificar información o al menos tener una identidad. Esto permite al sistema adjuntar ciertos objetos o interacciones a los marcadores.

Para implementar este tipo de sistema, es necesario realizar de forma general las siguientes operaciones:

1. Detectar el marcador.
2. Identificarlo.
3. Calcular su pose.

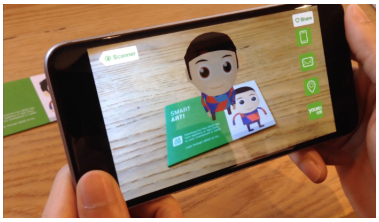


Figura 2.22: Ejemplo de realidad aumentada basado en marcadores

Basados en características

La alternativa a los basados en marcadores son los basados en características. En vez de extraer la posición de la cámara analizando la pose de un objeto conocido en el entorno, la pose de la cámara puede ser extraída de las características naturales que pertenecen al entorno. Idealmente,

un objeto virtual puede ser insertado en una posición relativa a objetos dentro del entorno sin la necesidad de insertar un marcador como punto de referencia.[8, 29]

Una de sus ventajas es que al contrario de los basados en marcadores, no requieren una modificación física del entorno. Su desventaja es que requieren una mayor capacidad computacional, son más sensibles a los cambios de iluminación y no funcionan adecuadamente en ambientes donde sus objetos cambien de pose frecuentemente.

Para implementar este tipo de sistema, es necesario realizas de forma general las siguientes operaciones:

1. Extraer las características de una imagen o un cuadro de vídeo.
2. Comparar las características extraídas con las almacenadas previamente.
3. Calcular su pose.

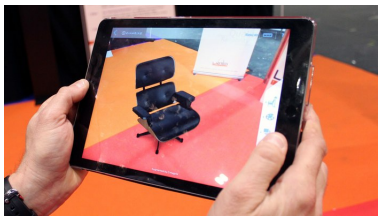


Figura 2.23: Ejemplo de realidad aumentada basada en características

2.4. Geometría Multivista

La geometría multivista es el campo que estudia las relaciones entre las cámaras y las características cuando hay correspondencias entre las múltiples imágenes que fueron tomadas desde distintos puntos de vista. La más importante de ellas es la geometría desde dos puntos de vista.

Con dos vistas de una escena y los puntos con correspondencia en estas vistas, existen restricciones geométricas en los puntos de la imagen como resultado de la orientación relativa de la cámara, las propiedades de la cámara y la posición de los puntos 3D.

2.4.1. Homografía

Una homografía es una transformación proyectiva 2D que establece una correspondencia entre los puntos de un plano con otro. Las homografías tienen muchos usos prácticos, tales como, alinear imágenes, rectificar imágenes, deformación de texturas y en la creación de panoramas.[30] La definición matemática de una homografía es:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ ó } x' = Hx$$

Las homografías pueden ser calculadas directamente a partir de los puntos que se corresponden en las dos imágenes (o planos). Cada punto en la correspondencia proporciona dos ecuaciones, una para la coordenada x y otra para y, y como una transformación proyectiva tiene 8 grados de libertad, por lo tanto se necesitan al menos cuatro correspondencias para calcular H. Si hay exactamente 4 correspondencias, entonces se puede calcular una solución exacta para H. Si hay más, entonces es posible que estas correspondencias no sean compatibles con alguna transformación proyectiva, y sea necesario determinar la "mejor"transformación H, que minimice alguna función de costo.[12]

2.4.2. Transformada Lineal Directa

La **transformada lineal directa**, es un algoritmo para calcular H dado cuatro o mas correspondencias entre dos imágenes. Reordenando la ecuación $x' = Hx'$, se obtiene que

$$Ah = 0$$

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & x_1 \\ 0 & 0 & 0 & -x_1 & y_1 & -1 & x_1x'_1 & y_1y'_1 & y'_1 \\ -x_2 & y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 & x_2 \\ 0 & 0 & 0 & -x_2 & y_2 & -1 & x_2x'_2 & y_2y'_2 & y'_2 \\ & \vdots & & \vdots & & \vdots & & \vdots & \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = 0$$

Donde A es una matriz con el doble de filas que el número de correspondencias, se puede calcular una solución de mínimos cuadrados usando el método de descomposición en valores simples (SVD).[30, 12]

2.4.3. Modelo de Cámara estenopeica

El modelo de cámara estenopeica o modelo de cámara proyectiva, es un modelo simple y suficientemente preciso para la mayoría de las aplicaciones, es muy utilizado en la visión por computador. El nombre viene del tipo de cámara, como una cámara oscura, la luz atraviesa una pequeño agujero al interior de una caja o cuarto oscuro. En este modelo, la luz pasa a través de un solo punto, el centro de la cámara, C, antes de ser proyectado a un plano de imagen.

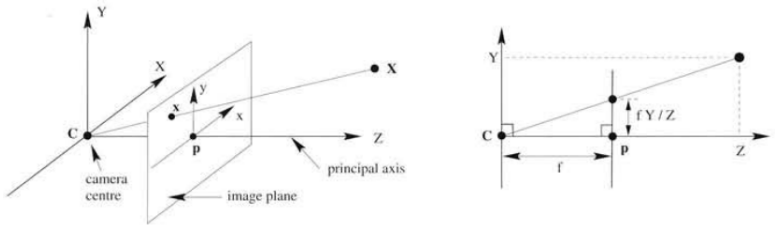


Figura 2.24: Modelo de cámara estenopeica

Las propiedades proyectivas de una cámara estenopeica pueden ser derivadas de la figura 2.24 y asumiendo que el eje de la imagen esta alineado con los ejes x e y del sistema de coordenadas 3D, por ende, el eje óptico de la cámara coincide con el eje z y la proyección se obtiene a partir de triángulos similares. Y si se toma en cuenta la rotación y traslación para insertar un punto 3D en este sistema de coordenadas antes de ser proyectado, se obtiene la transformación de proyección completa.

Con la cámara estenopeica, un punto 3D X es proyectado a un punto x de la imagen (ambos en coordenadas homogéneas) como

$$\lambda x = PX$$

Donde la matriz de 3x4 P se llama la matriz de la cámara (o matriz de proyección). El escalar λ es el inverso de la profundidad de un punto 3D y es necesario si queremos todas las coordenadas sean homogéneas con el último valor normalizado a uno.

La matriz de la cámara

La matriz de la cámara puede ser descompuesta como:

$$P = K \begin{bmatrix} R & | & t \end{bmatrix}$$

Donde R es la matriz de rotación que describe la orientación de la cámara, t un vector de traslación 3D que describe la posición del centro de la cámara, y K la matriz de calibración intrínseca que describe las propiedades proyectivas de la cámara.

La matriz de calibración, depende únicamente de las propiedades de la cámara y su forma general se escribe de la siguiente manera:

$$K = \begin{bmatrix} \alpha f & s & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

La distancia focal f, es la distancia entre el plano de imagen y el centro de la cámara. La oblicuidad s, solamente se usa si el arreglo de píxeles en el sensor están oblicuos y en la mayoría de los casos se puede asignar el valor de 0. La relación de aspecto α, es la relación de aspecto para los píxeles en el sensor, generalmente se asume como 1. Con lo que la matriz anterior queda de la siguiente forma:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Los únicos parámetros restantes son la distancia focal f y las coordenadas del centro óptico, el punto de la imagen $c = \begin{bmatrix} c_x & c_y \end{bmatrix}$, donde el eje óptico intersecta al plano de la imagen. El centro óptico se puede aproximar como la mitad del ancho y altura de la imagen, con lo que la única variable desconocida es la distancia focal f.

2.4.4. Calibración de la Cámara

La calibración de una cámara especifica los parámetros intrínsecos y extrínsecos dada una configuración de una o más cámaras. Los parámetros intrínsecos son la distancia focal, la dimensión de la matriz de los sensores, el tamaño de la celda de los sensores o relación de aspecto, los parámetros de distorsión radial, las coordenadas del punto principal o el factor de escalamiento. Los parámetros extrínsecos son las transformaciones afines que se aplican para identificar las poses de la cámara en un un sistema de coordenadas global.[30, 12]

Existen distintos tipos de enfoques para calcular los parámetros intrínsecos y extrínsecos para una configuración específica de cámaras. Los más comunes son la transformada directa lineal y por multiplanos usando el método de Zhang.

Calibración por DLT

La calibración por el método de la transformada lineal directa (DLT) usa las correspondencias entre los puntos del mundo y los puntos de la imagen de la cámara, para estimar los parámetros de la cámara. En particular, la calibración mediante DLT explota el hecho de que el modelo de cámara estenopeica, define un conjunto de relaciones de similaridad que pueden ser resueltas mediante el algoritmo de transformada lineal directa. Para emplear este enfoque, se requiere coordenadas precisas de un conjunto no degenerado de puntos 3D. Una manera típica de alcanzar esto, es construir una plataforma de calibración a partir de tres tableros de ajedrez mutuamente perpendiculares. Debido a que las esquinas de cada recuadro son equidistantes, calcular las coordenadas 3D de cada esquina dado el ancho y el alto de cada recuadro es directa. La ventaja de la calibración DLT es su simplicidad, un conjunto de cámaras arbitrarias puede ser calibradas resolviendo un único sistema

lineal homogéneo de ecuaciones. Sin embargo, la practicidad de la calibración DLT esta limitada por la necesidad de una plataforma de calibración 3D y por el hecho de necesitar coordenadas 3D extremadamente precisas para evitar inestabilidad numérica.[12]

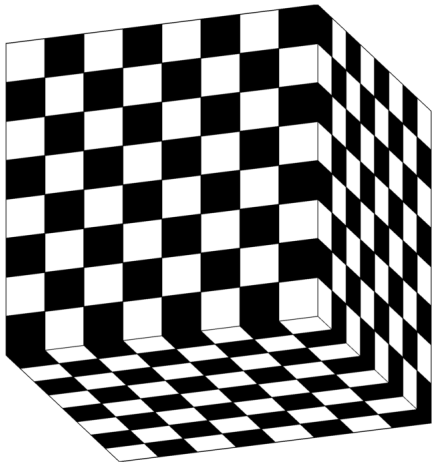


Figura 2.25: Plataforma de calibración 3D

Calibración por multiplanos

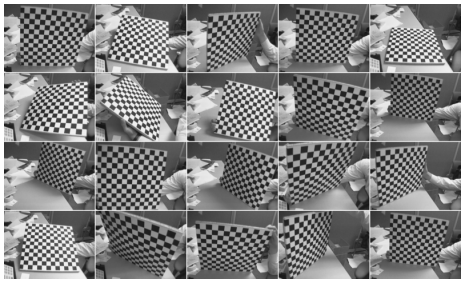
La calibración por multiplanos es una variante de la auto-calibración de cámaras, que permite calcular los parámetros de una cámara a partir de dos o más vistas de una superficie planar. El trabajo pionero en esta área es el de Zhang[33]. El método de Zhang calibra las cámaras resolviendo un sistema lineal homogéneo particular que captura las relaciones homográficas entre múltiples perspectivas de un mismo plano. Este enfoque multivista es popular, ya que es mas natural captar múltiples vistas de una sola superficie planar, como por ejemplo un tablero de ajedrez, que construir una plataforma de calibración, como en la calibración DLT.

2.5. SLAM

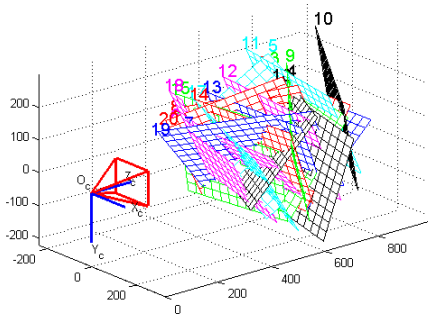
Los sistemas de localización y generación de mapas simultanea (SLAM - *Simultaneous Localization And Mapping*), son aquellos que en un ambiente desconocido y a partir de marcas naturales existentes, estiman su ubicación y orientación de manera simultanea y la de las marcas naturales [6, 16, 5, 13].

Existen muchas soluciones a este problema, utilizando diferentes enfoques y sensores. Los cuales se pueden agrupar en dos períodos, el período clásico (1986 - 2004). En esta época se introdujo una formulación probabilística para SLAM, incluidos ideas basadas en los Filtros Extendidos de Kalman, Filtros de Particulas de Rao-Blackwellised. Además delineó los desafíos básicos relacionados a la eficiencia y a la asociación robusta de datos [6, 16, 5, 13].

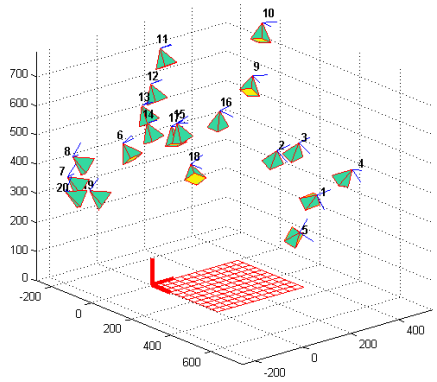
El segundo período, es el que se denomina el período de análisis algorítmico (2004 - 2015). En este período se estudiaron las propiedades fundamentales del SLAM, incluyendo la observabilidad,



(a) Vistas Multiples de un tablero de ajedrez para calibración por multiplanos



(b) Orientaciones reconstruidas (coordenadas de cámara)



(c) Orientaciones reconstruidas (coordenadas de mundo)

convergencia y consistencia. También se entendió el papel clave de de aprovechar las matrices dispersas que salen de la formulación del problema, para la implementación de soluciones más eficientes al problema del SLAM [6, 16, 5, 13].

La arquitectura de los sistemas SLAM incluyen dos componentes principales: el frontend y el backend. El frontend convierte los datos de los sensores en modelos que son amigables para ser estimados, mientras que el backend realiza inferencias sobre el modelo producido por el frontend [6, 16, 5, 13].

Los sistemas SLAM resuelven muchas tareas dependiendo de la aplicación, aunque existen algunas tareas comunes entre todos ellos que representan el corazón de estos sistemas, rastreo, generación de mapas, reubicación y cierre de ciclo. El rastreo es la que se encarga de hacerle seguimiento a las marcas naturales del ambiente. La de generación de mapa, con la información obtenida de los sensores ir creando y expandiendo un mapa del entorno a medida que se va explorando. La reubicación, tener la capacidad de volverse a orientar. La tarea de cierre de ciclo poder identificar la verdadera topología del entorno reconociendo cuando se ha visitado algún sitio más de una vez, sin esta capacidad el sistema asumiría que el entorno es un corredor infinito [6, 16, 5, 13].

Algunos de los sistemas SLAM que se han desarrollado hasta la actualidad son, EKF-SLAM, FastSLAM, L-SLAM, PTAM, LSD-SLAM, ORB-SLAM, GraphSLAM y LinearSLAM.

2.6. ORB-SLAM

ORB-SLAM es un sistema SLAM monocular basado en características, opera en tiempo real, en ambientes interiores y exteriores, grandes y pequeños. Fue creado por Raúl Mur-Artal en el año 2015 [25].

Este sistema fue diseñado a partir de las ideas expuestas en el trabajo de Klein y Murray sobre SLAM, Parallel Tracking and Mapping (PTAM) [16], para la selección de fotogramas claves, pareo de características, triangulación de puntos, ubicación de la cámara para cada fotograma y reubicación después de ocurrir una falla en el rastreo. También se basa en los trabajos sobre reconocimiento de lugares de Gálvez-López y Tardós [9], cierre de ciclos de Strasdat et. al [31] y el uso de la información sobre la covisibilidad para operaciones a gran escala [32, 23].

2.6.1. Arquitectura

El sistema esta dividido en tres módulos: rastreo, manejo del mapa local y el del cierre de ciclo (ver Figura 2.26), donde cada modulo se ejecuta en un hilo independiente. Para detectar e identificar las características se usa el descriptor ORB, debido a que es rápido de calcular y aparear, manteniendo buena invarianza a los cambios de puntos de vista, rotación y escalamiento [25, 24]. Para los cálculos de optimización se usan los algoritmos implementados en la librería g2o [19].

Módulo de Rastreo

El módulo de rastreo, es el encargado de ubicar la cámara en cada fotograma y decidir cuando insertar un fotograma nuevo. El primer paso que realiza este módulo es apareamiento entre el fotograma actual y el anterior y optimizar la pose usando un ajuste solo tomando en cuenta el movimiento. Si el rastreo se pierde (debido a movimientos bruscos o oclusión), el módulo de reconocimiento de lugares es utilizado para ejecutar una reubicación global [25].

Una vez que se calcula un estimado inicial de la pose de la cámara y de las características apareadas, se recupera el mapa local dentro del campo de visión de la cámara usando el grafo de covisibilidad de fotogramas que mantiene el sistema. Luego se buscan las parejas junto con los puntos correspondientes al mapa local y se vuelve a calcular la pose óptima de la cámara. Finalmente, decide si el fotograma debe ser insertado [25].

Módulo de Mapa Local

El módulo de mapa local, procesa los nuevos fotogramas y realiza un ajuste local para conseguir una reconstrucción óptima de los alrededores a la cámara. Las nuevas correspondencias para los ORB apareados en el nuevo fotograma, se buscan en los fotogramas adyacentes en el grafo de covisiblidad para triangular nuevos puntos. También dependiendo de la información recolectada, se aplica una estricta política de filtrado para mantener solos los puntos mejor representados y de los fotogramas redundantes.

Módulo de Cierre de Ciclo

El módulo de cierre de ciclo, busca ciclos cada nuevo fotograma. Si se detecta un ciclo, se calcula una transformada de similitud que informa acerca del error acumulado en el ciclo. Luego ambos lados del ciclos se alinean y los puntos duplicados se fusionan. Finalmente se aplica una optimización sobre un grafo de poses sobre restricciones de similaridad [31] para alcanzar una consistencia global.

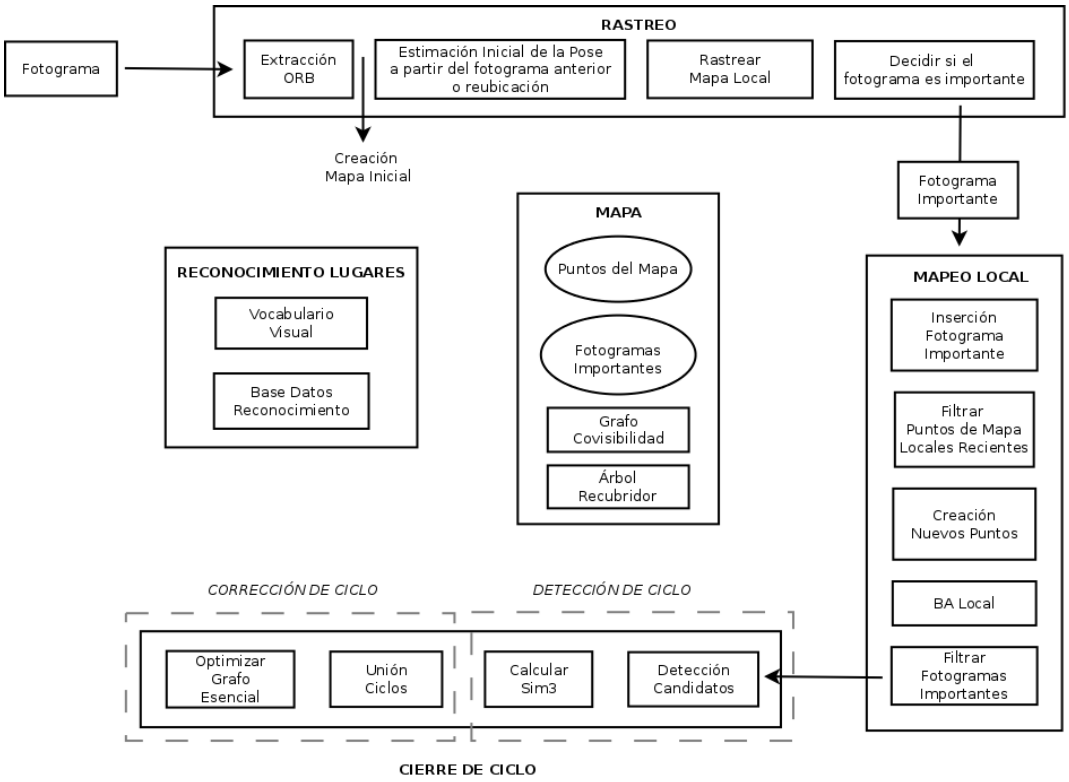


Figura 2.26: Arquitectura del ORB-SLAM

Las ventajas que ofrece este sistema son:

- El uso de las mismas características (ORB) para todas las tareas, rastreo, trazado de mapas, reubicación y cierre de ciclo. Con lo que se obtiene un sistema mas eficiente, simple y confiable.
- El uso del descriptor ORB para la identificación, detección y apareamiento de características, con lo que se consigue un desempeño en tiempo real sin el uso del GPU, además provee una buena invarianza a los cambios de punto de vista, iluminación, rotación y escalamiento.
- Manejo de ambientes grandes en tiempo real.
- Cierre de ciclos en tiempo real.
- Reubicación de la orientación de la cámara en tiempo real, con una alta invarianza a los cambios de punto de vista e iluminación. Con lo que permite al sistema recuperarse de una falla en el rastreo e incrementa la reutilización del mapa.

- Un proceso robusto de inicialización para la creación de un mapa inicial en cualquier tipo de escena.
- Un manejo eficiente de los fotogramas y puntos del mapa, a través de una política flexible para la incorporación pero bastante restrictiva para la eliminación, con lo que se mejora la robustez del rastreo y las operaciones a largo plazo, porque la información redundante es descartada.

Las desventajas que posee este sistema son:

- Debe existir buena iluminación.
- No funciona con área donde existan pocas características.
- No funciona adecuadamente cuando existen patrones repetidos frecuentemente.
- El mapa generado no necesariamente esta en la misma escala del mundo real.

Capítulo 3

Marco Aplicativo

En este capítulo se documenta el proceso de desarrollo de las actividades que se siguieron para la implementación de la aplicación móvil para el sistema operativo Android, objeto del presente Trabajo Especial de Grado, tomando como base, la metodología de desarrollo iterativo e incremental, la cual permite de una forma flexible y efectiva la implantación de dicha aplicación.

Con esta aplicación se busca ofrecer una herramienta de realidad aumentada, que facilite la decoración de un espacio interior, de una manera rápida y sencilla, haciendo uso de los últimos avances en el área de SLAM, como el sistema ORB-SLAM, para la generación de mapas virtuales usando cámaras monoculares y adaptándola para ser usada en el sistema Android.

3.1. Descripción General de la aplicación

Una aplicación es un sistema complejo, por lo cual para entender su funcionamiento, es indispensable tener una visión general de las características, requerimientos y funcionalidades del mismo. En este caso, la descripción general de la aplicación contiene una lista de los requerimientos funcionales y no funcionales, la plataformas soportadas, la guía de estilo para los elementos de la interfaz, los casos de uso del sistema, un listado de las interfaces principales y una vista general de como es la arquitectura de la aplicación. Los cuales se describen a continuación.

3.1.1. Requerimientos Funcionales

- Calibrar la cámara del dispositivo y calcular su matriz intrínseca(K).
- Escanear un espacio interior y generar un mapa virtual, para ser usado como guía en el proceso de decoración.
- Estimar la pose del dispositivo a partir de la imagen captada por la cámara.
- Mostrar un catálogo de modelos dependiendo del tipo de espacio interior a decorar.
- Agregar y registrar modelos del catálogo como objetos virtuales dentro del mundo real.
- Trasladar, escalar y rotar objetos virtuales.

- Visualizar los objetos virtuales que se encuentran dentro del frustrum de la cámara.

3.1.2. Requerimientos No Funcionales

- Un rendimiento adecuado que permita usar la aplicación de manera fluida.
- Soportar la mayor cantidad de dispositivos posibles con diferentes especificaciones.
- Una arquitectura modular y fácilmente extensible.
- Una interfaz simple pero visualmente agradable.
- La interfaz de decoración debe obstruir lo menos posible la vista de la cámara.

3.1.3. Plataforma

La aplicación corre bajo el sistema operativo Android, soporta las versiones de Android 4.2.1 en adelante, funciona tanto teléfonos inteligentes como en tabletas. La aplicación requiere al menos 200 MB de espacio disponible de memoria secundaria (para almacenar los modelos y el archivo de vocabulario que usa ORB-SLAM), un mínimo de 500 MB de memoria RAM libre y un procesador quadcore o mejor para que funcione adecuadamente.

3.1.4. Estilo

Una vez definidos los requerimientos funcionales y no funcionales de la aplicación, es necesario establecer las pautas que se deben seguir para el diseño de las interfaces de la aplicación. Ya que la aplicación funciona en el sistema operativo Android, se decidió usar las pautas de Material Design[21] creadas por Google. De acuerdo a las especificaciones del Material Design, es necesario definir una paleta de colores, que se lista en la Tabla 3.1. Además se definió el logo de la aplicación y el conjunto de iconos que se muestran en las Figura 3.1 y Figura 3.2.

Tipo de Color	RGB	Color
Primario	#2196F3	
Primario Oscuro	#1976D2	
Primario Claro	#BBDEFB	
Énfasis	#FF5722	
Texto Primario	#212121	
Texto Secundario	#757575	
Iconos	#FFFFFF	
Divisor	#BDBDBD	

Tabla 3.1: Paleta de colores usada por la aplicación basadas en Material Design

3.1.5. Modelo de Casos de Uso

Un modelo de casos de uso describe la secuencia y el comportamiento del sistema en las interacciones que se desarrollarán durante su comunicación con los actores, en respuesta a un evento

Figura 3.1: Logo de la aplicación

Figura 3.2: Listado de los iconos de la aplicación

iniciado por un actor y/u otro sistema. Además, con este modelo se llega a un lenguaje estándar que es entendido tanto por los expertos como por los usuarios y que permite el análisis del dominio de la aplicación. El modelo de casos de uso de la aplicación se puede ver en la Figura 3.3. Y la descripción de los casos de uso en las tablas que se muestran a continuación.

Caso de Uso	1. Calibrar Cámara
Actor	Usuario
Tipo	Asociación
Descripción	Permite al usuario calibrar la cámara del dispositivo
Precondición	Ninguna
Postcondición	La cámara del dispositivo ha sido calibrada

Tabla 3.2: Caso de Uso - Calibrar Cámara

Caso de Uso	2. Decorar Espacio
Actor	Usuario
Tipo	Asociación
Descripción	Permite al usuario decorar un espacio con modelos virtuales
Precondición	La cámara ha sido calibrada
Postcondición	El usuario a decorado un espacio con modelos virtuales

Tabla 3.3: Caso de Uso - Decorar Espacio

Caso de Uso	1.1 Ver Tablero Calibración
Actor	Usuario
Tipo	Extend
Descripción	Abrir el archivo que contiene el tablero de calibración para ser impreso
Precondición	Ninguna
Postcondición	Visualizar el archivo con la imagen del tablero de calibración

Tabla 3.4: Caso de Uso - Ver Tablero Calibración

Caso de Uso	1.2 Tomar Foto del Tablero
Actor	Usuario
Tipo	Extend
Descripción	Tomar fotos del tablero de calibración y calcular la matriz intrínseca de la cámara
Precondición	Tener impreso el tablero de calibración
Postcondición	La cámara del dispositivo esta calibrada

Tabla 3.5: Caso de Uso - Tomar Foto del Tablero

Caso de Uso	2.1 Seleccionar Tipo Espacio
Actor	Usuario
Tipo	Extend
Descripción	Permite seleccionar el tipo de espacio a decorar y catalogo de modelos a utilizar
Precondición	La cámara del dispositivo debe estar calibrada
Postcondición	Se almacena la selección del usuario

Tabla 3.6: Caso de Uso - Seleccionar Tipo Espacio

Caso de Uso	2.1.1 Cambiar de Modo
Actor	Usuario
Tipo	Extend
Descripción	Permite al usuario cambiar entre la modalidad de escanear y decorar
Precondición	Se debe haber seleccionado un tipo de espacio a decorar
Postcondición	Pasar de la modalidad escanear a decorar o viceversa

Tabla 3.7: Caso de Uso - Cambiar de Modo

Caso de Uso	2.1.2 Cambiar Estado Vista
Actor	Usuario
Tipo	Extend
Descripción	Permite al usuario cambiar entre una vista bloqueada o desbloqueada. La vista bloqueada congela la imagen de fondo y la desbloqueada actualiza constantemente la imagen de fondo con la capturada por la cámara.
Precondición	La aplicación se encuentra en modalidad de decorar
Postcondición	Cambiar el estado de la vista, pasar de la bloqueada a la desbloqueada y viceversa

Tabla 3.8: Caso de Uso - Cambiar Estado Vista

Caso de Uso	2.2.3 Agregar Objeto
Actor	Usuario
Tipo	Extend
Descripción	Permite al usuario agregar un nuevo objeto del catalogo al espacio
Precondición	La vista debe estar bloqueada
Postcondición	El modelo es agregado y registrado al espacio

Tabla 3.9: Caso de Uso - Agregar Objeto

Caso de Uso	Seleccionar Objeto
Actor	Usuario
Tipo	Extend
Descripción	Permite seleccionar un objeto para ser trasladado, escalado o rotado
Precondición	La vista debe estar bloqueada
Postcondición	Se almacena cual objeto fue seleccionado para su posterior transformación

Tabla 3.10: Caso de Uso - Seleccionar Objeto

Caso de Uso	2.1.5 Trasladar Objeto
Actor	Usuario
Tipo	Extend
Descripción	Permite al usuario trasladar un objeto seleccionado previamente de una posición a otra
Precondición	Existe un objeto seleccionado
Postcondición	El objeto cambio su posición

Tabla 3.11: Caso de Uso - Trasladar Objeto

Caso de Uso	2.1.6 Rotar Objeto
Actor	Usuario
Tipo	Extend
Descripción	Permite al usuario rotar un objeto seleccionado previamente
Precondición	Existe un objeto seleccionado
Postcondición	El objeto cambio su orientación

Tabla 3.12: Caso de Uso - Rotar Objeto

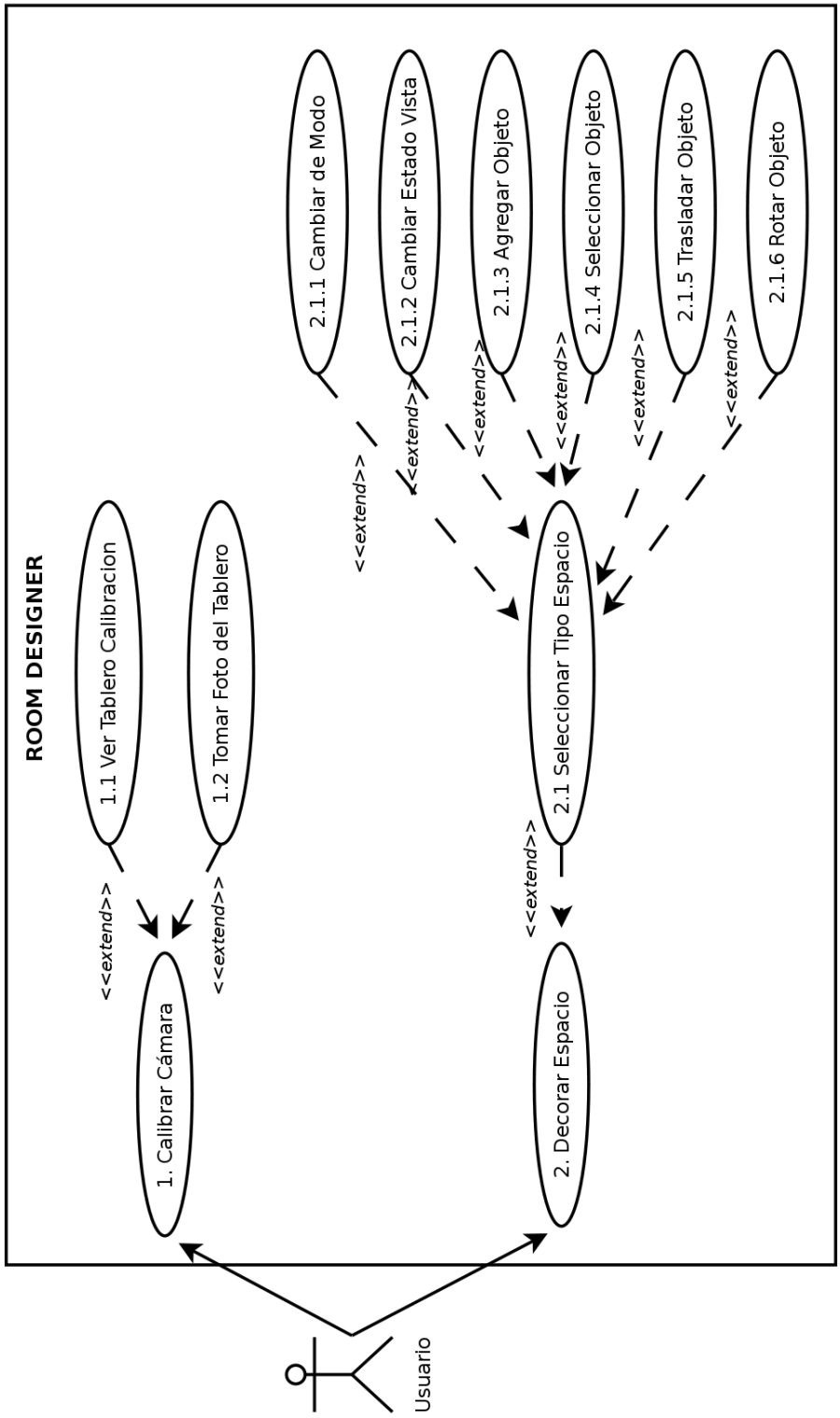


Figura 3.3: Casos de Uso

3.1.6. Interfaces

La aplicación se puede dividir en tres partes de acuerdo a la funcionalidad que está asociada. La primera es la interfaz principal que es donde el usuario puede elegir entre las dos funcionalidades posibles, calibrar la cámara del dispositivo y decorar un espacio. La segunda la conforman el conjunto de interfaces que se encargan de manejar todo el proceso de calibración del dispositivo. Y la tercera parte las interfaces relacionadas con el proceso de decoración.

Interfaz Principal

Esta es la interfaz inicial de la aplicación, acá es donde el usuario puede elegir entre las opciones de calibrar la cámara del dispositivo y decorar un espacio. Es una interfaz sencilla, donde se muestra el logo de la aplicación en la parte superior y dos botones con sus respectivos iconos para cada opción. La interfaz principal se puede apreciar en la Figura 3.4.

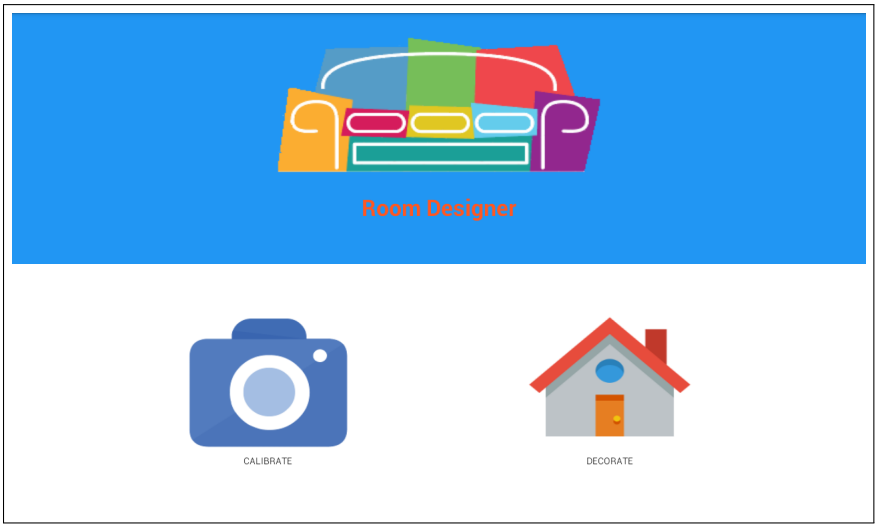
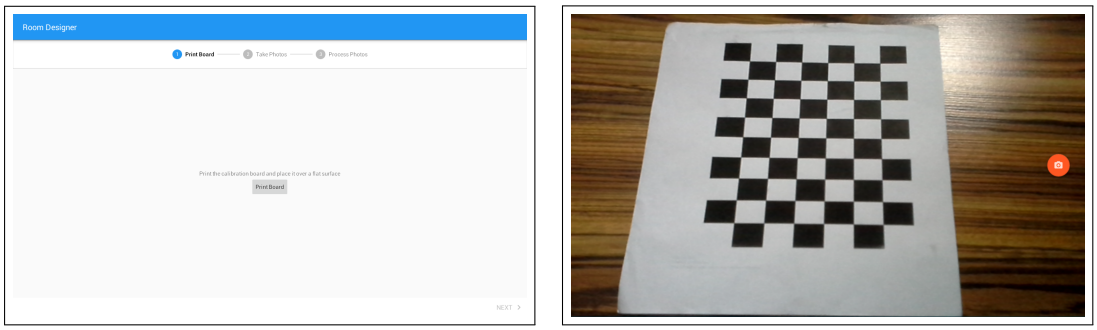


Figura 3.4: Interfaz Principal

Interfaces de Calibración

Las interfaces de calibración son el conjunto de interfaces que se encargan de calibrar el dispositivo. El proceso de calibración esta dividido en tres pasos, abrir el archivo que contiene el tablero de calibración para que sea impreso por el usuario, la captura de 10 fotos del tablero de calibración desde distintas posiciones y por último el calculo de la matriz intrínseca de la cámara, que es necesaria en la sección de decoración. Las interfaces mas representativas se pueden ver en la Figura 3.5.



(a) Abrir archivo con el Tablero de Calibración (b) Capturar el Tablero de Calibración

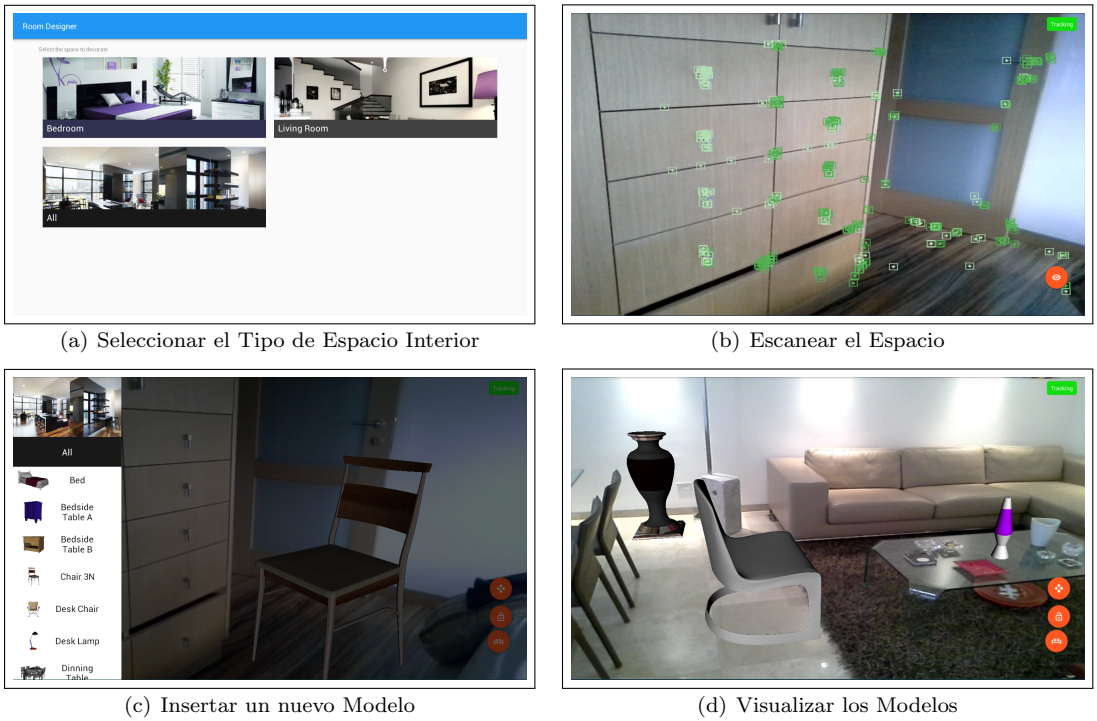
Figura 3.5: Interfaces de Calibración

Interfaces de Decoración

Las interfaces de decoración son el conjunto de interfaces que se encargan del proceso de decoración de un espacio. El proceso de decoración esta divido en tres pasos, seleccionar el tipo de espacio a decorar, escanear el espacio a ser decorado y generar un mapa virtual y finalmente decorar usando modelos 3D.

La interfaz para escanear un espacio posee dos elementos importantes, una etiqueta en la parte superior derecha, que indica el estado en que se encuentra. Cada estado viene representado por un color, el rojo (iniciando) para identificar que el sistema esta intentando generar un mapa válido a partir de 2 fotogramas con cierto paralelaje entre ellas, el verde (rastreando) indica que actualmente se esta escaneando el ambiente y azul (desorientado) que indica que el sistema ha perdido su orientación y se debe enfocar un área escaneada previamente para volver a orientarse. El otro elemento importante, son los cuadros verdes con un punto en el medio que aparecen sobre la imagen, estos representan los puntos que actualmente están siendo rastreados por el sistema, vienen en 3 diferentes grados de verde, desde un verde claro hasta un verde brillante. El color de punto viene fuertemente relacionado con la calidad del rastreo del punto en el mapa virtual, entre mas brillante mejor calidad. Una vez que se considera que se ha escaneado el espacio adecuadamente se cambia a la modalidad de decoración usando el botón que se encuentra en la parte inferior derecha, con lo que el mapa virtual deja de ser actualizado.

En la modalidad de decoración se puede ir navegando el espacio escaneado anteriormente y se visualizaran los objetos insertados por el usuario que se encuentren dentro de esa área. Para insertar nuevos modelos se debe bloquear la vista actual con el botón del candado cerrado, se procede a abrir el catálogo que se encuentra oculto en el lado izquierdo de la pantalla, haciendo swipe del lado izquierdo de la pantalla hacia el derecho, se selecciona el modelo, se posiciona el objeto en la ubicación y orientación deseada y por último desbloqueamos la vista con el botón del candado abierto. Los objetos insertados al ambiente se registran con ayuda del mapa virtual y solo se muestran aquellos que se encuentran dentro del frustum de la cámara. La interfaces mas importantes se muestran en la Figura 3.6.



(a) Seleccionar el Tipo de Espacio Interior

(b) Escanear el Espacio

(c) Insertar un nuevo Modelo

(d) Visualizar los Modelos

Figura 3.6: Interfaces de Decoración

3.1.7. Arquitectura

La arquitectura de la aplicación esta dividida en dos parts, el *frontend* representado por *AndroidActivity* y el *backend* representado por *AREngine*, tal como se ve en la Figura 3.7. Cada parte está desarrollada en un lenguaje de programación diferente, el *frontend* en Java y el *backend* en C++.

Frontend

El *frontend* esta compuesto por tres hilos, el *UIThread* que se encarga de manejar todo lo relacionado con la interfaz, el *SLAMThread* a cargo del manejo de la cámara, del procesamiento de los fotogramas y de la comunicación entre la interfaz y el backend y por último el *RenderThread* que controla la visualización de los resultados obtenidos por AREngine. Toda comunicación entre hilos se realiza usando pase de mensajes.

Backend

El *backend* esta compuesto por dos módulos, el ORB-SLAM y el RenderEngine. El módulo de ORB-SLAM, que a partir de la imagen capturada por la cámara calcula su pose y crea un mapa virtual del entorno, luego envía la pose y la imagen capturada por la cámara al módulo de AREngine. El módulo de RenderEngine usa la pose de la cámara para dibujar los modelos insertados por el

usuario que estén dentro del campo de visión de la cámara y que estén orientados correctamente en conjunto con la imagen capturada por la cámara como imagen de fondo.

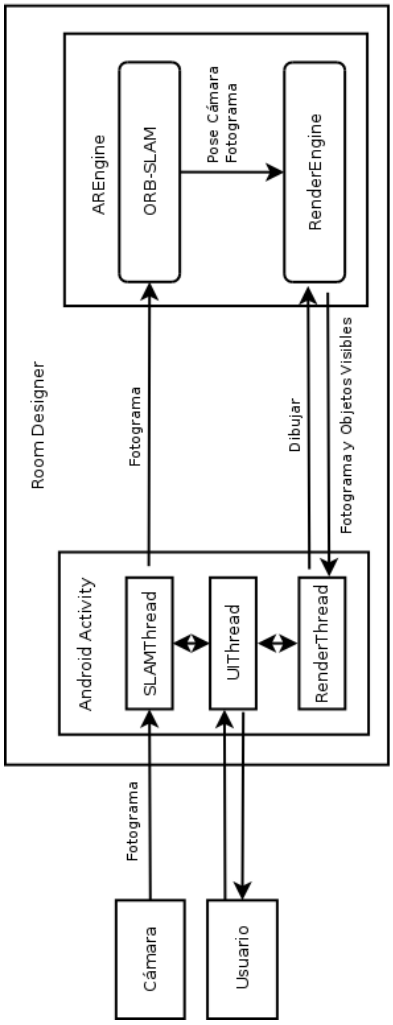


Figura 3.7: Arquitectura Room Designer

3.2. Proceso de Desarrollo

El proceso de desarollo a usar será un desarrollo de forma iterativa e incremental. El Desarrollo iterativo e incremental es un proceso de desarrollo de software, donde se agrupan un conjunto de tareas en pequeñas etapas repetitivas (iteraciones). El modelo consta de diversas etapas de desarrollo en cada incremento, las cuales inician con el análisis, luego con la etapa de iteración y finalizan con la instauración y aprobación del sistema. La idea principal de este proceso es desarrollar un sistema de manera incremental, permitiéndole al desarrollador sacar ventaja de lo que se ha aprendido a lo largo del desarrollo anterior, incrementando, versiones entregables del sistema.

Las iteraciones realizadas en este desarrollo se definen en la Tabla 3.13, las cuales se detallan a continuación.

Iteración	**Actividad**
1	Desarrollo de las funcionalidades de visualización
2	Calibración de la cámara
3	Integración del ORB-SLAM
4	Mejora de las interfaces de la aplicación
5	Implementación del Catálogo de Modelos
6	Manejo de los objetos a través de gestos
7	Implementación de un sistema de ayuda

Tabla 3.13: Listado de las iteraciones en el desarrollo de la aplicación

3.2.1. Iteración 1: Desarrollo de las funcionalidades de visualización

El objetivo de esta iteración consistió en el desarrollo del módulo de visualización usando la librería gráfica OpenGL ES, específicamente la versión 2.0 que es la soportada por la versión 16 del API de Android. Para el uso de OpenGL en Android se requiere utilizar el componente de interfaz *GLSurfaceView*, que tiene como responsabilidad el manejo de la superficie donde OpenGL dibujara los elementos, hacer el despliegue en un hilo aparte para no afectar el rendimiento de hilo principal (*UIThread*) en donde se maneja el resto de la interfaz, a este hilo lo denominamos *RenderThread*.

Uno de los aspectos importantes del *RenderThread* es que es el único hilo donde se pueden realizar llamadas a las funciones provistas por OpenGL ES, lo que implica que, si se requiere usar alguna funcionalidad provista por esta API, se debe pasar un mensaje al *RenderThread* para que ejecute la tarea usando las funciones provista por Android para la comunicación entre hilos.

Para el despliegue de los objetos usando OpenGL ES es necesario crear una clase que implemente la interfaz *GLSurfaceView.Renderable* y asignarsela al *GLSurfaceView*. La interfaz *Renderable* define tres métodos que deben ser implementados por la clase, el método *onSurfaceCreated*, que se llama una vez que se crea o recrea la superficie donde sera desplegada, el método *onSurfaceChanged* que se llama cuando las dimensiones de la superficie cambian y el método *onDrawFrame* que ejecuta cuando se desea dibujar.

En el caso de la aplicación, la clase que implementa la interfaz *GLSurfaceView.Renderable* es *NativeRenderer* y las funcionalidades que tiene cada método a ser implementado son, el método *onSurfaceCreated* para inicializar los recursos necesarios, cargar los shaders para dibujar la información capturada por la cámara, la creación de la textura a donde sera enviada esta imagen y posteriormente para iniciar el sistema SLAM. El método *onSurfaceChanged* para alinear la matriz de proyección del dispositivo con la que se utiliza para dibujar los objetos con OpenGL ES usando las dimensiones de la superficie y la matriz intrínseca de la cámara del dispositivo. El método *onDrawFrame* para dibujar los modelos que se encuentran dentro del campo de visión de la cámara y la imagen capturada por la cámara.

El módulo de visualización esta desarrollado con el lenguaje de programación C++ y el NDK de Android, de manera que se facilite la integración entre el sistema SLAM y el resto de la aplicación. Debido a que tenemos que comunicar la sección de la aplicación manejada por la VM (máquina

virtual) de Java (definida en la clase *NativeRenderer*) con la sección nativa usando JNI (Java native interface), se creó una clase intermedia que abstrajera del frontend los detalles internos del backend. La clase encargada de este trabajo se denomino *AREngine*, que encapsulara a los módulos de visualización y SLAM.

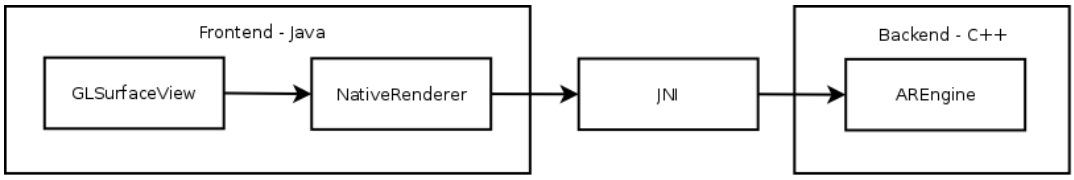


Figura 3.8: Proceso de comunicación entre el frontend y el backend por medio de JNI

Una vez definida como es la comunicación entre el backend y el frontend usando el JNI y la clase *AREngine*, es necesario implementar el módulo de visualización en la sección nativa de la aplicación. El módulo está compuesto por varias clases que se encargarán de administrar los recursos de OpenGL ES y los modelos 3D, tales como las texturas, los shaders, la memoria para los vértices, normales, coordenadas de texturas que usan los mallados, etc. Para ello se definieron varias clases, *TextureManager* (administra las texturas), *Program* (controla un programa de GPU), *Material* (definición del material a usar, texturas y programa de GPU), *Mesh* (datos del mallado, vértices, normales, coordenadas de texturas y el material), *ProgramManager* (administra las instancias de Program), *OBJLoader* (cargar los modelos 3D en formato OBJ), *Model* (define la lista de mallados que usa un modelo), *ModelManager* (gestiona las instancias de Model) y *Entity* (instancia de un Modelo dentro del espacio virtual con su respectiva posición, orientación y escala). La relación entre las clases se puede ver en el diagrama de clase en la Figura 3.9.

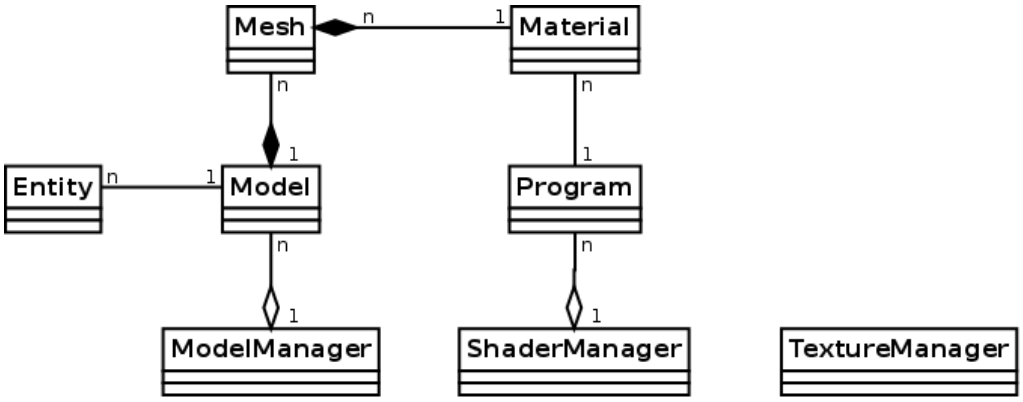


Figura 3.9: Diagrama de clase de las clases relacionadas con la visualización

El objetivo principal de las clases administradoras de recursos es reusar la información lo más posible y así minimizar los tiempos de carga o inicialización, como por ejemplo, si un usuario desea insertar dos veces un mismo modelo, solo se llamaría al procedimiento de carga una vez y la segunda se identifica que el modelo ya ha sido cargado anteriormente y se ahorra realizar ese paso. Además de minimizar los tiempos de carga, también se minimiza el uso de memoria al no existir datos repetidos.

A partir de la versión 2.0 de OpenGL ES, el flujo de trabajo incorpora dos componentes pro-

gramables que son el *Vertex Shader* y el *Fragment Shader*. El primero trabaja a nivel de vértices y el segundo a nivel de fragmentos, tal como se muestra en la Figura 3.10. Para cumplir con este requisito en la aplicación se idearon dos programas sencillo, uno para el *Vertex Shader* y otro para el *Fragment Shader* que proyectan los modelos sobre la pantalla según las matrices de proyección, vista y modelado suministradas y colorearlos de acuerdo a la textura seleccionada.

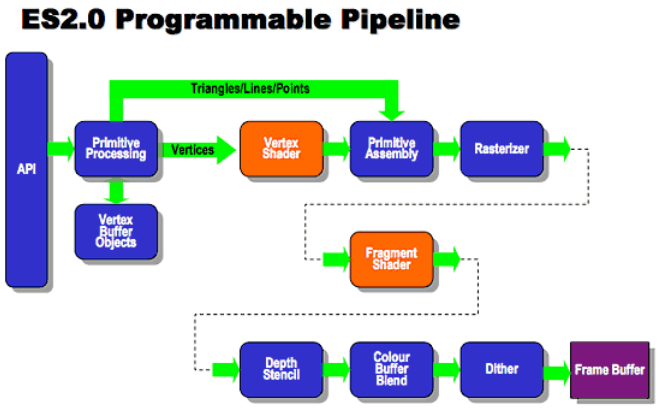


Figura 3.10: Flujo de trabajo de OpenGL ES 2.0 [15]

El proceso de visualización de los modelos y de la imagen capturada por la cámara es un proceso relativamente sencillo que usa una técnica de doble buffer para la imagen de fondo. La técnica de doble buffer permite usar de manera eficiente el GPU porque evita que espere a que los datos sean transferidos a su memoria con la desventaja de que se muestra es la imagen capturada anteriormente. Cada vez que *GLSurfaceView* llama al método *onDrawFrame* se realiza el procedimiento que se enumera de forma general a continuación.

1. Enviar los datos al GPU de la imagen capturada por la cámara a la textura que se usa como fondo disponible.
2. Indicar al GPU el programa a utilizar para renderizar los elementos.
3. Deshabilitar el test de profundidad para dibujar la imagen de fondo.
4. Dibujar un rectángulo que ocupe toda la superficie de despliegue y que use como textura la definida como fondo actualizada en la llamada anterior.
5. Marcar la textura de fondo anterior como disponible y la actual como la anterior.
6. Habilitar el test de profundidad para dibujar los modelos encima de la imagen de fondo.
7. Enviar al programa GPU las matrices de proyección y vista.
8. Para cada *Mesh* de las instancias de *Entity* dentro del frustum, se indica la textura a utilizar, la sección de memoria donde se encuentran los datos de los vértices, coordenadas de texturas y normales y la matriz de modelo. Y finalmente se dibuja el mallado correspondiente.

Al final del proceso, el resultado esperado es la imagen de fondo capturada por la cámara y por encima los modelos 3D dibujados correctamente. El resultado final se puede observar en la Figura 3.11.

Figura 3.11: Visualización de un modelo 3D

3.2.2. Iteración 2: Calibración de la cámara

En esta fase se desarrollo la funcionalidad para la calibración de la cámara del dispositivo. Este paso es necesario para el uso de la librería ORB-SLAM en un futuro y para el registro de los objetos virtuales cuando se aumenta el entorno. La calibración de la cámara, que es equivalente a la estimación de los parámetros internos de la cámara, permite al ORB-SLAM calcular la posición de uno o más puntos de interés que encuentran en dos imágenes tomadas desde dos puntos de vista distintos.

Para realizar la calibración se usa un patrón de tablero de Ajedrez que provee la librería de OpenCV de tamaño 10x7, que se muestra en la Figura 3.12. El tablero de calibración debe ser impreso y puesto sobre una superficie plana, asegurándose que no se mueva de su posición mientras se realice el proceso de calibración.

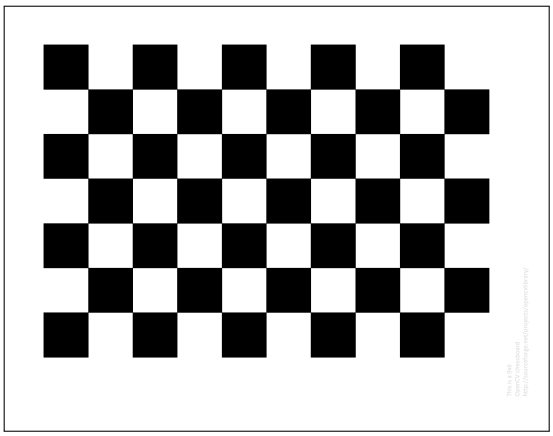


Figura 3.12: Tablero de Calibración - Patrón tipo tablero de Ajedrez de 10x7

El proceso de calibración es bastante directo y simple, una vez que se coloca el tablero de calibración en una superficie plana, el usuario debe tomarle fotos desde distintas posiciones para garantizar una mejor estimación de los parámetros. Para la detección del tablero de calibración se

usa la función de OpenCV *findChessboardCorners* y para la estimación de los parámetros internos se usa la función de OpenCV *calibrateCamera*. En el caso de la aplicación se estableció que se deben tomar 10 fotos, así se garantiza un buen balance entre la cantidad de datos y el tiempo que le tomaría al usuario realizar la calibración. Una vez que se capturan todas las fotos, la aplicación procede a calcular los parámetros y finalmente se guardan en un archivo de texto para posterior uso. Algunos ejemplos de como tomar las fotos se pueden ver en la Figura 3.13. Este paso solo es necesario realizarlo una vez, aunque se deja la opción de repetirlo si el usuario considera que no lo realizo correctamente.

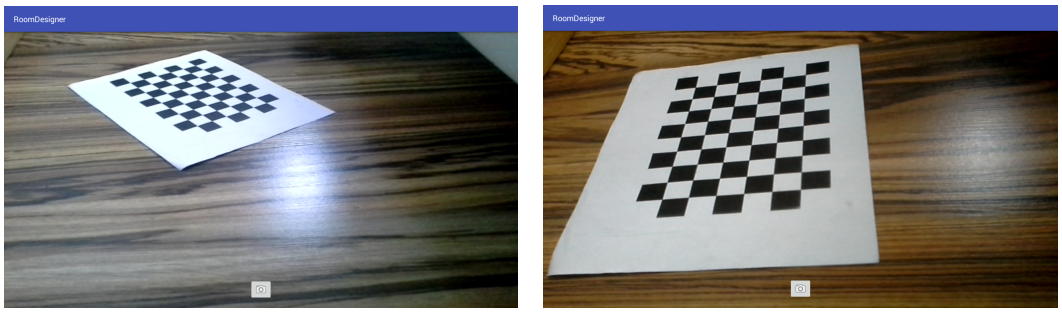


Figura 3.13: Ejemplo de como se deben tomar las fotos del tablero para calibrar la cámara

Para garantizar una mejor experiencia del usuario, la detección del tablero y el calculo de la matriz intrínseca se realiza en otro hilo independiente para no bloquear el *UIThread*. Además de ejecutarse en otro hilo, se le indica al usuario que la foto esta siendo procesada para que el usuario este consciente de que la aplicación esta trabajando y la cantidad de fotos que se han procesado hasta ese momento para que sepa en que paso se encuentra actualmente, esto se puede ver en la Figura 3.14. Una vez finalizada la detección de la foto se le indica con un mensaje si la detección fue exitosa o no.

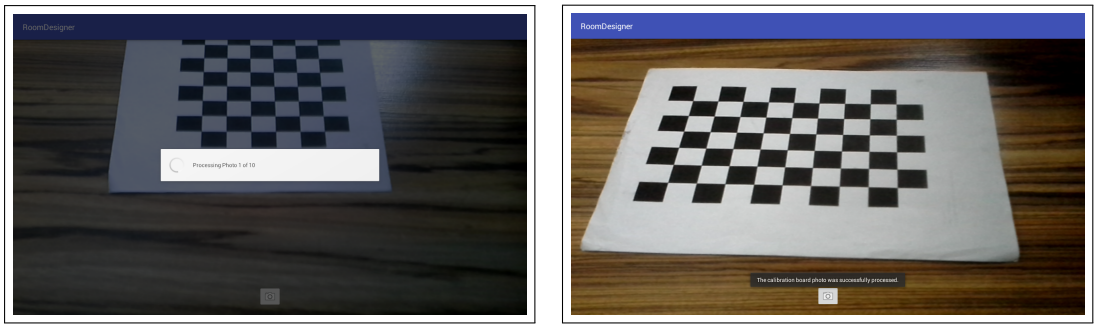


Figura 3.14: Notificación del estado de la calibración

Dado que se implemento las funcionalidades de calibración y visualización se creó una vista que será la interfaz principal de la aplicación, en donde se permitirá al usuario elegir la acción que desea realizar, calibrar o decorar. Esta interfaz inicialmente solo contiene dos botones y solo se puede decorar una vez que se haya calibrado la cámara anteriormente. En la Figura 3.15 se muestra como es la interfaz principal en esta iteración.

Figura 3.15: Interfaz principal

3.2.3. Iteración 3: Integración del ORB-SLAM

El objetivo de esta iteración, es la integración de ORB-SLAM a la aplicación. El código de este algoritmo fue hecho con un enfoque hacia el área de la robótica y cámaras monoculares RGB, RGB-D y las estéreo y para los sistemas basados en GNU/Linux, por lo que es imprescindible portar el código y sus dependencias al sistema operativo Android.

El código disponible es una aplicación a enfocada a evaluar el algoritmo y no para ser incorporada a códigos de terceros, esto implica que es necesario remover todas las funcionalidades que no sean indispensables para el cálculo de la pose de la cámara ni de la generación del mapa, asegurándose en no alterar la lógica del algoritmo. Las funcionalidades que requieren ser eliminadas son la relacionada con la visualización, la lectura de los datos de entrada y la relacionada con las cámaras que no sean monoculares RGB.

El primer paso para portar el código es identificar las dependencias y revisar que funcionen en el sistema operativo Android, que en este caso todas las dependencias funcionan correctamente en Android. Las dependencias de ORB-SLAM son Eigen para los cálculos de álgebra lineal, g2o para la optimización de funciones de error no lineales basadas en grafos, OpenCV para la manipulación de imágenes y algoritmos de visión por computador, pagolin para la visualización e interfaz y DBoW2 para el reconocimiento de lugares usando imágenes. De todas las dependencias, la única que no es necesaria para el proyecto es la librería pangolin, ya que el proceso de visualización se maneja directamente con el SDK de Android.

En cuanto al código de ORB-SLAM el proceso de acomodar el código para que funcione en Android se realizo por partes, el primer paso fue eliminar la dependencia de la librería pangolin para la visualización y dibujar los puntos que están siendo rastreados por el sistema SLAM sobre la imagen capturada para verificar su correcto funcionamiento. El segundo paso acomodar las funciones que no están soportadas por el NDK de Android y algunas funciones que dan errores de compilación. El tercer paso, cargar el archivo de vocabulario necesario para el módulo de reconocimiento. Y el último paso, cargar los datos de calibración generados por el módulo de calibración de la aplicación. El resultado se puede apreciar en la Figura 3.16

Ya con el código portado al sistema Android, se evidenció que la carga del archivo vocabulario es demasiado lenta, es un archivo de aproximadamente 145 MB y tarda varios minutos en leer toda

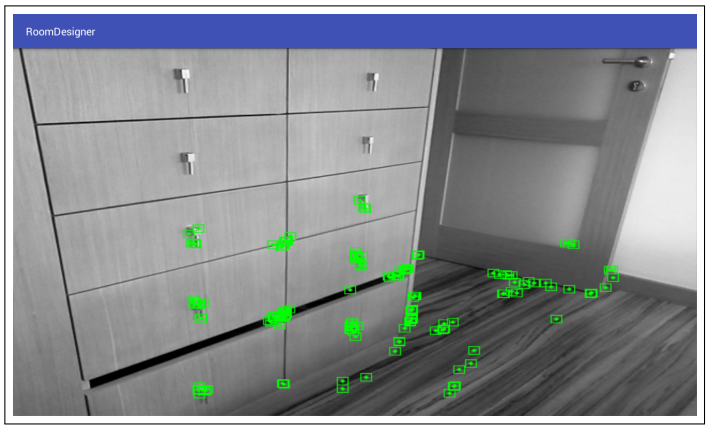


Figura 3.16: Visualización de los resultados de ORB-SLAM

la información. Con lo que se decidió transformar el archivo de texto a un archivo binario, donde en vez de leer un campo a la vez, se leen varios campos a la vez. Además que al ser un archivo binario los datos ocupan menos espacio de disco secundario, el archivo pasó de 145 MB a aproximadamente 44 MB y el tiempo de carga paso de minutos a unos cuantos segundos. Para una mejor experiencia del usuario, se creó una pantalla de carga para que el usuario este al tanto de que la aplicación esta trabajando. La interfaz de carga se puede apreciar en la Figura 3.17.

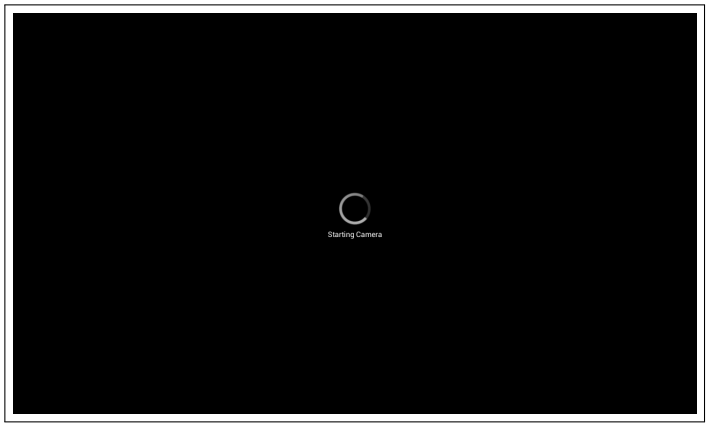


Figura 3.17: Interfaz de carga para la inicialización del proceso de decoración

Otro punto importante en el que se trabajo fue en el funcionamiento adecuado del algoritmo, a pesar de que es un código que ya se ha trabajado durante un tiempo considerable y en la mayoría de los casos funciona correctamente, aún tiene errores en su implementación. Los errores que pudieron ser encontrados e identificado la razón de su falla fueron solucionados, entre algunos de ellos están bloqueo por abrazo mortal a la hora de acceder a un recurso crítico, código para muestrear los puntos de interés no era aleatorio, el uso de variables incorrectas para iterar un arreglo de elementos, errores calculando la matriz intrínseca. Otro aspecto negativo del código, es el manejo de la memoria, que no siempre se libera toda la memoria que se reserva durante la ejecución, con lo que a largo plazo el dispositivo puede disminuir su desempeño, por empezar a usar la memoria secundaria durante el proceso de paginación al usar toda la memoria principal disponible. Se hicieron algunas

mejoras en este punto pero para la gran mayoría del código no se realizaron cambios, ya que afectaban considerablemente el rendimiento de la aplicación o requería reescribir casi todo el código nuevamente.

Y finalmente con la integración del ORB-SLAM a la aplicación se puede conocer la orientación de la cámara, que a su vez será usada como la matriz vista para el renderizar los modelos. Además de la orientación, también es necesario conocer el estado en que se encuentra el sistema SLAM a medida que procesa los fotogramas, para que el usuario pueda saber que acción realizar. Los estados disponibles son, iniciando, rastreando y desorientado. Para el estado de iniciando, el sistema aún esta en proceso de crear el mapa virtual, en este estado el usuario debe enfocar la cámara en un punto e ir moviendo el dispositivo alrededor de ese punto de manera horizontal, hasta que el sistema encuentre dos imágenes con cierto paralelaje para poder iniciar el sistema SLAM. El estado de rastreo, el usuario puede seguir moviéndose dentro del espacio interior y el estado desorientado el usuario debe enfocar la cámara en un punto que el sistema rastreo previamente para que se vuelva a orientar dentro del espacio.

3.2.4. Iteración 4: Mejora de las interfaces de la aplicación

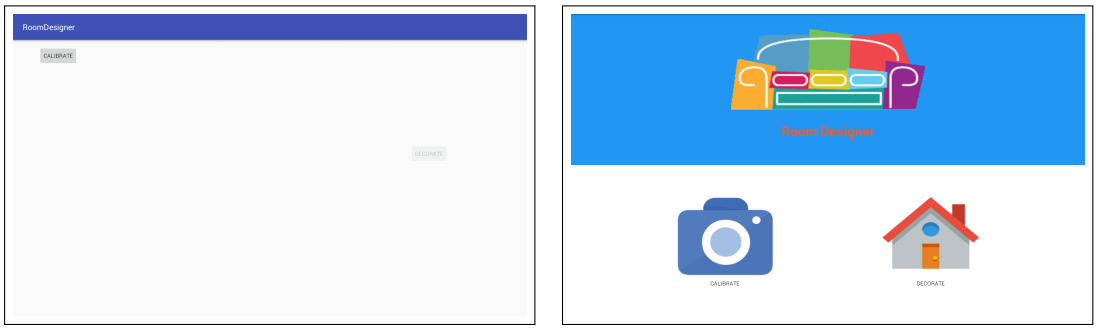
La intención de esta fase, es la mejora de todas la interfaces de la aplicación. La idea es construir las interfaces que sean visualmente amenas y que la aplicación sea lo más usable posible. Para cumplir con ambos puntos, se siguió las especificaciones de *Material Design*[22] creadas por Google para el diseño de las interfaces. Además se definió una paleta de colores para cada uno de los tipos de color necesarios en el *Material Design*, los cuales se lista en la Tabla 3.14. Las interfaces que se mejoraron son, la principal, la de calibración y la de decoración.

Tipo de Color	RGB	Color
Primario	#2196F3	
Primario Oscuro	#1976D2	
Primario Claro	#BBDEFB	
Énfasis	#FF5722	
Texto Primario	#212121	
Texto Secundario	#757575	
Iconos	#FFFFFF	
Divisor	#BDBDBD	

Tabla 3.14: Paleta de colores usada por la aplicación basadas en Material Design

Interfaz Principal

La interfaz principal actual, solo contiene dos botones para acceder a las funcionalidades de calibrar y decorar (Ver Figura 3.18. En el nuevo diseño se dividió la pantalla en dos secciones, la superior con el logo y el nombre de la aplicación y fondo de color primario, y la parte inferior los botones para acceder a las secciones de calibración y decoración, con un fondo blanco. Los botones se cambiaron por imágenes representativas de las acciones, pero también tienen el nombre de la acción debajo de la imagen. El resultado final se puede apreciar en la Figura 3.18.



(a) Actual (b) Nueva

Figura 3.18: Interfaz principal

Interfaz Calibración

La interfaz actual para la calibración de la cámara, es simplemente una ventana donde se deben tomar 10 fotos al tablero de calibración, pero no se le explica al usuario, que debe hacer exactamente y no existe una forma de obtener directamente el tablero de calibración en la aplicación (Ver Figura 3.19). Por lo tanto, se decidió crear un asistente que guié al usuario en cada uno de los pasos, abrir e imprimir el tablero de calibración, tomar las fotos correspondientes y luego calibrar la cámara del dispositivo con las imágenes. Otra consideración para el diseño, es permitir que el usuario pueda rehacer algún paso, en caso de algún error.

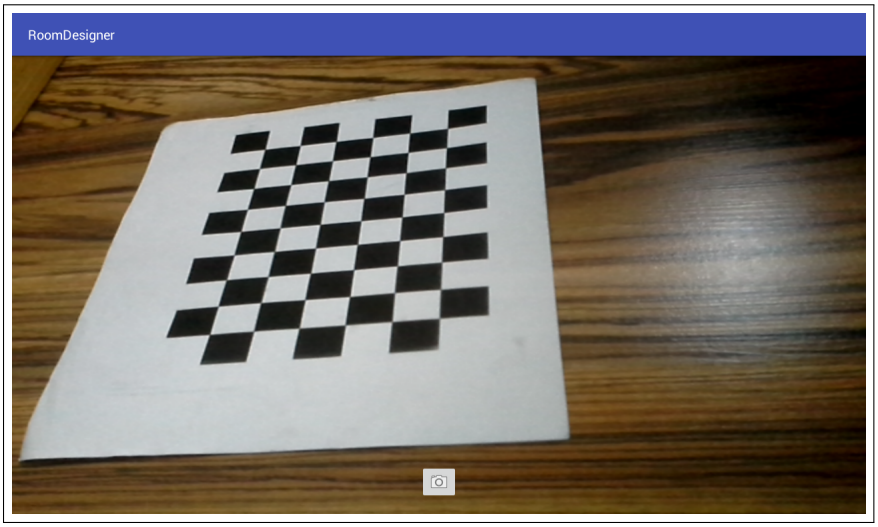
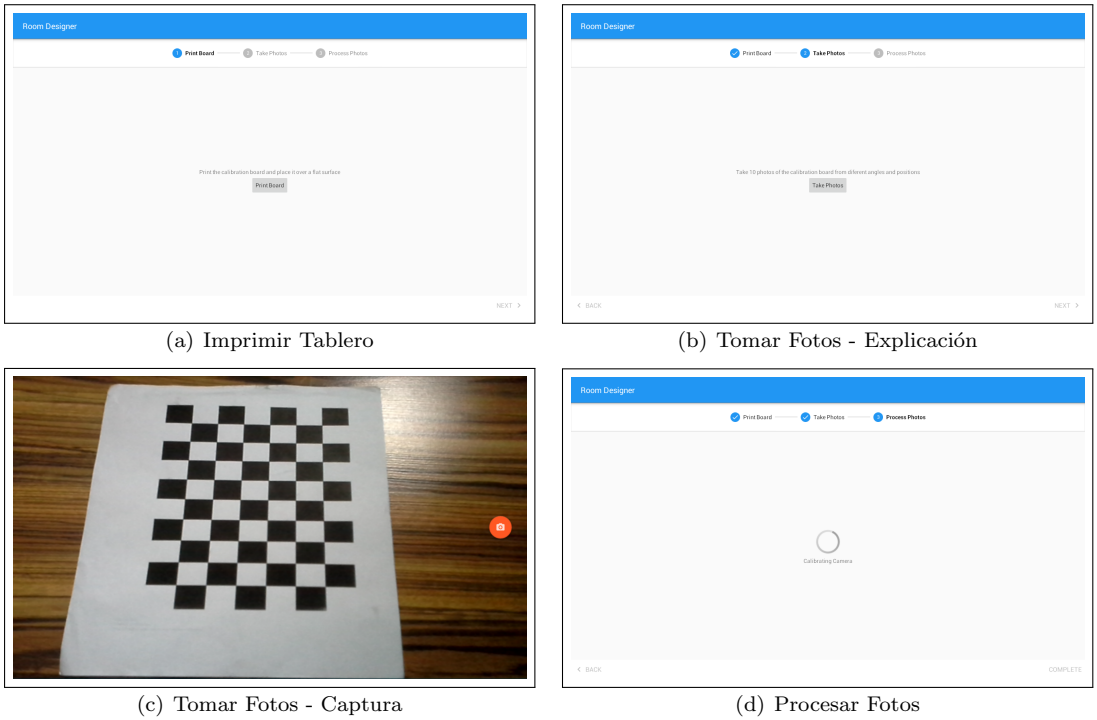


Figura 3.19: Interfaz de calibración actual

La interfaz nueva, esta dividida en tres secciones, la superior, cuenta con una barra donde se enumeran los pasos a completar para calibrar el dispositivo, indicando además el paso en que el usuario se encuentra actualmente, con un circulo de color azul, los pasos completados con un círculo azul y una marca de visto bueno, y los pasos pendientes con un círculo de color gris. La sección del medio es donde se explica que se debe hacer y un botón para ejecutar esa acción. Y La sección

inferior, una barra de navegación donde se puede ir al siguiente paso (sí se ha completado) o al anterior (sí se desea repetir algún paso). Las nuevas interfaces se pueden ver en la Figura 3.20.

Otra modificación importante, es el cambio de posición del botón para tomar la foto del tablero de calibración, inicialmente el botón se encontraba en la parte inferior pero se movió a la sección intermedia y se alineo hacia la derecha. La razón de esta modificación, es debido a que mientras se use la aplicación, es necesario tener el dispositivo de forma horizontal, por lo que es más fácil alcanzar con los dedos los bordes del dispositivo, que la mitad del mismo.



(a) Imprimir Tablero

(b) Tomar Fotos - Explicación

(c) Tomar Fotos - Captura

(d) Procesar Fotos

Figura 3.20: Interfaces del proceso de calibración

Al completar el proceso de calibración dependiendo de la opción que eligió el usuario en la pantalla principal, la aplicación llevará al usuario de nuevo a la pantalla principal, sí eligió calibrar, e irá a la pantalla de selección del tipo de espacio a decorar, sí eligió la opción de decorar.

Interfaz de Decoración

Por ahora, en esta interfaz solo se pueden ver los resultados de las características que rastrea el ORB-SLAM. Esta información para el usuario es insuficiente para saber que acción debe tomar a lo largo de corrida, ya que sí por ejemplo, el ORB-SLAM cambia al estado desorientado, el usuario no lo sabrá. Por lo cual, es indispensable mostrar en la interfaz el estado actual del ORB-SLAM.

Otra elemento importante a considerar, es el proceso de escaneo de un espacio interior. Una vez que el ORB-SLAM ha generado un mapa inicial entre dos imágenes con cierto paralelaje entre ellas, pasa al estado de rastreo y empieza a hacer seguimiento de las características de la imagen

capturada por la cámara del dispositivo. En la interfaz actual, las características rastreadas se muestran en la pantalla como unos recuadros verdes, tal como se ve en la Figura 3.21.

Las características a medida que se van procesando, las mismas pueden ser incorporadas al mapa virtual, descartadas o aumentar/disminuir la confianza de que pertenezcan al mapa y ser rastreadas. La información de que tan confiable o no es una característica, puede ser relevante a la hora de guiar al usuario en el proceso de escaneo, por ejemplo, sí el área que esta escaneando actualmente, hay muchas características confiables, el usuario podría pasar a escanear otra sección sin ningún inconveniente ya que el sistema pudo generar un mapa bastante robusto de esa área. En cambio si hay pocas o ninguna característica confiable, es posible que el sistema tenga que volver al estado de inicialización al no tener suficientes características que rastrear. Por lo que, contar con alguna manera de conocer que tan confiable o no es una característica seria ideal.

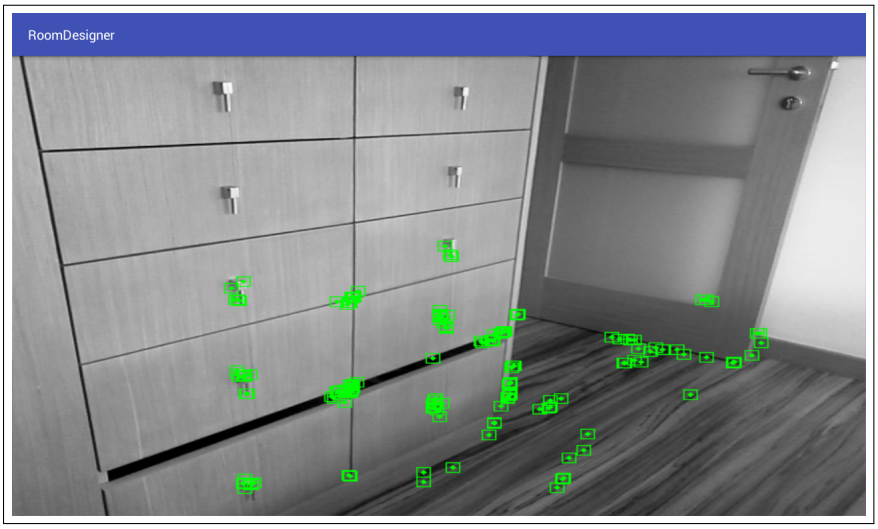


Figura 3.21: Interfaz de decoración actual con las características que estan siendo rastreadas por el ORB-SLAM

Por otra parte, debido a las características inherentes de los sistemas SLAM monoculares, los mapas que van generando, son mapas a cierta escala del mundo real, y a medida, que el sistema se alimenta con mas datos, el factor de escalamiento puede variar. Esta variación en el factor de escalamiento, trae como consecuencia, que al insertar un objeto virtual dentro del mundo y al variar el factor de escalamiento, puede que el objeto ya no se encuentre en la misma área que se coloco. Para solucionar este problema, hay 2 posibles soluciones, la primera es reajustar la posición de los objetos cuando cambie el factor de escalabilidad y la segunda es dividir el proceso de decoración en dos partes, una para escanear el ambiente y luego otra para la decoración, en donde se insertan y transforman los objetos, pero no se actualiza el mapa del ORB-SLAM con nuevos datos. La opción elegida fue la segunda, por ser más simple y no requiere calcular y actualizar las posiciones de los objetos.

Tomando en cuenta los puntos anteriores, se diseñó la nueva interfaz para la decoración. Ahora la interfaz tendrá dos posibles estados, el de escanear y el de decorar, el icono del botón indica en que estado se encuentra actualmente, un ojo para el estado de escanear y un mueble para el de decorar, tal como se puede apreciar en la Figura 3.22. Los elementos que ahora se muestren en la interfaz dependerá del estado en que se encuentre. Para cambiar entre los estados, se usa un

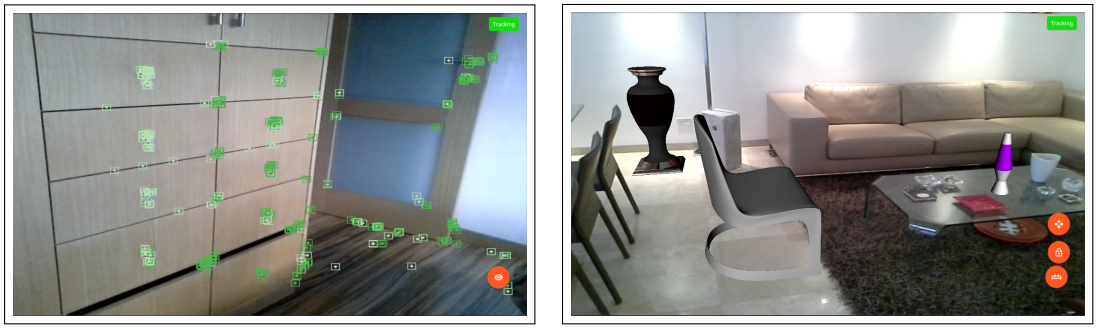
botón que se encuentra en la esquina inferior derecha, que solo esta activado cuando el sistema de ORB-SLAM se encuentra en estado de rastreo, ya que no tiene sentido intentar insertar objetos cuando no se ha generado un mapa, en el caso del estado de inicialización, o cuando está en el estado de desorientado y lo que se busca, es que se vuelva a orientar el sistema SLAM.

Para el estado de escanear, los elementos que se muestran en la interfaz son, un indicador del estado, en la parte superior derecha. cada estado está representado con un color diferente, rojo para el estado inicializar, verde para el estado de rastreo y azul para el de desorientación. La otra modificación, es la identificación la calidad de las características rastreadas a través de colores, para ello se definió unos niveles de confianza según cuantos frames "ven"la característica en el grafo de covisibilidad del ORB-SLAM, los cuales se listan en la Tabla 3.15, a mayor nivel más confiable es la característica.

Nivel de Confianza	RGB	Color
1	#B4DDB4	
2	#52B152	
3	#00FF00	

Tabla 3.15: Niveles de confianza para las características rastreadas por ORB-SLAM

Para el estado de decorar, los elementos que se muestran en la interfaz son dos botones, un botón para bloquear/desbloquear la vista y un botón para cambiar entre la acción de trasladar y rotar un objeto. El botón de bloquear/desbloquear la vista, congela la imagen de fondo que proviene de la cámara. La razón de este botón, es para facilitar el proceso de inserción y/o edición de los objetos, ya que es posible que a medida que el usuario realice este proceso, mueva el dispositivo, y por ende, el objeto no quede en la posición u orientación deseada. Al tener la imagen congelada se evitara este inconveniente. Luego cuando el usuario decida que ya terminó de editar los objetos presentes en la vista actual, procedería a desbloquear la vista. El diseño final de ambos estados, se pueden ver en la Figura 3.22.



(a) Modalidad Escanear (b) Modalidad Decorar

Figura 3.22: Interfaces del proceso de decoración

3.2.5. Iteración 5: Implementación del Catálogo de Modelos

El objetivo de esta iteración, es la implementación del catálogo de los modelos a ser usados por los usuarios al momento de decorar un espacio. Un catálogo es una lista ordenada o clasificada

de elementos que pertenecen a un conjunto dado, en el caso de la aplicación, el conjunto son los modelos 3D de objetos disponibles para decorar un espacio interior.

Para poder incluir un modelo dentro del catálogo, el mismo debe cumplir con ciertos requisitos, deben estar en el formato *wavefront*, los materiales solo deben ser aquellos que usen una textura o color para definir el color del mallado, el mallado solo puede estar compuesto por triángulos, debe tener calculadas las normales de sus caras, el vector de dirección hacia arriba debe ser el z-negativo para estar alineado con el eje de coordenadas que usa la aplicación, el modelo debe estar centrado de acuerdo a su centro geométrico y deben tener una licencia que permita su uso no comercial.

Para definir la manera de clasificar los modelos, hay que pensar en como es el proceso de decoración. La persona elige un área dentro del espacio interior, esta área cumple con algún fin en particular y de acuerdo al propósito designado, se usaran ciertos objetos para su decoración. Por ejemplo, si el área es una cocina, los objetos que usualmente están presentes son aquellos relacionados con la actividad de cocinar y/o lavar como neveras, hornos, lavaplatos, microondas, lavadoras, secadoras, gabinetes, etc. Por lo tanto una buena manera de clasificar los modelos dentro del catálogo, es según el área a decorar y con la opción de poder decorar una área con cualquier modelo disponible sin importar el tipo de área. Cabe acotar que es posible que un objeto pueda ser usado en varias áreas.

Dentro de un espacio interior existen varios tipos de áreas pero para la aplicación solo existirán tres alternativas, habitación, sala de estar y la opción libre. La razón por la cuál solo habrán tres áreas, es porque adecuar los modelos para que cumplan con todos los requisitos anteriormente descritos, toma tiempo.

La mayoría de los modelos no cumplen con los requisitos para ser incluidos dentro del catálogo pero en casi todos los casos se pueden transformar hasta que cumplan con todos los requisitos siguiendo la información en la Tabla 3.16. La cantidad de modelos disponibles para ser incluidos en el catálogo eran 50 y luego de ser procesados se redujo la cantidad a 20.

Problema	**Solución**
Formato distinto a *wavefront*	Si es un formato soportado por *3ds Max* o *Blender* importar el modelo y exportar a wavefront sino descartar
Materiales que usan otra opción distinta a texturas o colores	Descartar modelo
Mallado con polígonos distintos de triángulos	Triangularizar mallado usando *Blender*
Algunas o todas las caras del mallado no tienen sus normales calculadas	Calcular las normales usando *Blender*
El vector de dirección hacia arriba no es el eje z-negativo	Cambiar el vector dirección hacia arriba usando *Blender*
El modelo no se encuentra centrado de acuerdo a su centro geométrico	Centrar el modelo a su centro geométrico usando *Blender*

Tabla 3.16: Tabla de como solucionar los problemas en los modelos

Una vez definido los requerimientos que deben tener los modelos del catálogo y como se clasificaran, es indispensable establecer de que manera se almacena esta información dentro del dispositivo. La idea es tener un formato sencillo de leer, flexible a la hora de agregar o eliminar atributos, que sea fácil de editar y legible para los humanos. Un formato que cumple con estas características es el formato de archivo JSON (*JavaScript Object Notación*. La descripción del catálogo en formato

JSON es un arreglo de modelos y la descripción de los modelos es la que se define en la Tabla 3.17.

Atributo	Tipo de dato	Descripción
name	String	El nombre del modelo
thumbnail	String	La imagen representativa del modelo para mostrar al usuario una vista previa de como se ve el modelo
model	String	El archivo del modelo 3D asociado
areas	Arreglo de String	Conjunto de áreas a las que pertenece el modelo. Las áreas disponibles son "Bedroomz "Living Room"

Tabla 3.17: Descripción de los atributos de los modelos del catálogo

Según lo mencionado anteriormente el primer paso para realizar la decoración de un espacio, es seleccionar que área se desea decorar, para cumplir con esto, una vez que el usuario selecciona la opción de decorar en la interfaz principal de la aplicación, se muestra al usuario la interfaz para que seleccione el área que desea decorar. La interfaz de selección esta compuesta por una lista de las áreas soportadas por la aplicación y la alternativa de decorar sin restricciones. Cada alternativa se presenta con una imagen representativa del espacio junto con su nombre del área, tal como se puede apreciar en la Figura 3.23. Las opciones disponibles como se menciono anteriormente son, el área de la habitación ("Bedroom"), sala ("Living Roomz libre (.ªll"). Finalmente, después de ser seleccionada el área se procede a comenzar con el proceso de decoración.

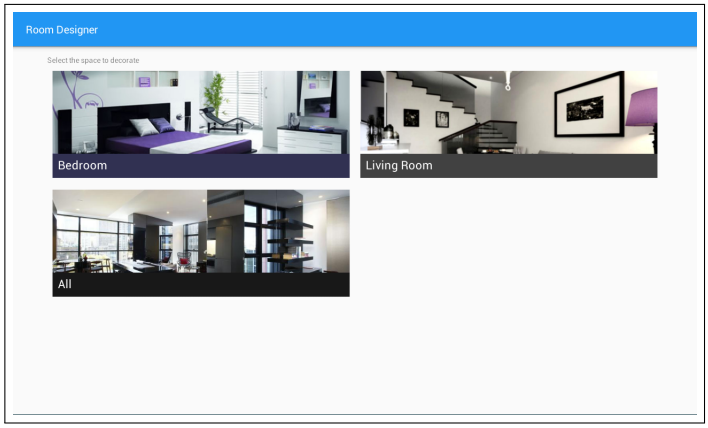


Figura 3.23: Interfaz para la sección del área a decorar

Para la visualización e inserción de los modelos del catálogo para el área seleccionada, se incorporo un componente en forma de barra en la interfaz de decoración. La barra contiene en la parte superior el área seleccionada para decorar y en el resto la lista de los modelos disponibles para decorar. Cada modelo se muestra con una imagen y el nombre del modelo, como se muestra en la Figura 3.24. La barra, inicialmente se encuentra oculta y cuando el usuario necesite agregar un objeto, se muestra en pantalla. Para ver el catálogo, el usuario debe colocar el dedo en el lado izquierdo de la pantalla y lo arrastra hacia el lado derecho, hasta que el catálogo esté completamente visible. Para ocultar el catálogo se presiona en la zona de la pantalla que no esta ocupada por el catálogo.

Para insertar un modelo, primero se debe abrir el catálogo con los pasos anteriormente descritos, luego buscar el modelo que se desea agregar y por último presionar con el dedo sobre el nombre

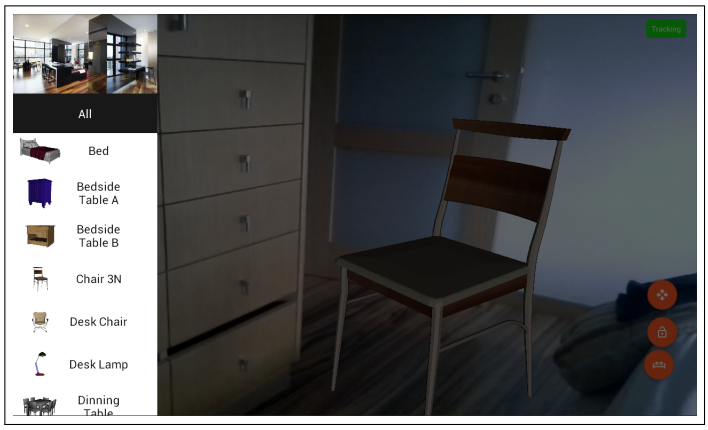


Figura 3.24: Interfaz del catálogo de modelos

o la imagen del modelo. La aplicación se encargará de posicionar el nuevo modelo en el centro de la pantalla y a una profundidad equivalente a la mitad del frustum, para calcular la posición del modelo, se transforma el punto central de la pantalla en píxeles, a coordenadas normalizadas de dispositivo (cada coordenada esta entre los valores -1 y 1) y luego se aplica la transformación para ir de las coordenadas normalizadas de dispositivo a coordenadas de mundo, usando la función provista por la librería GLM *unProject*. Además, para que el objeto inicialmente este contenido dentro de la pantalla, la aplicación escalará el objeto de tal forma que su altura sea 1/2 de dimensión vertical de la pantalla. Tal como se puede apreciar en la Figura 3.25.

Figura 3.25: Ejemplo de la posición inicial de un modelo cuando se acaba de agregar a la escena

3.2.6. Iteración 6: Manejo de los objetos a través de gestos

El alcance de esta iteración, es la implementación de la detección de gestos para la interacción con los objetos para la modificación de la posición, orientación y tamaño de los objetos. Cada acción esta asociada a un gesto y a un estado de la aplicación, como se define en la Tabla 3.18. Esta funcionalidad no esta activa siempre en el modo de decoración, sino que activa o desactiva

solamente cuando sea necesario, para evitar alterar los objetos por error. La manera de activar esta opción, es bloquear la vista, editar los objetos deseados y desbloquear la vista.

Acción	**Gesto**	**Precondición**
Seleccionar	Presionar con el dedo dos veces seguidas sobre el modelo (doble tap)	Ninguna
Trasladar	Movimiento con un solo dedo sobre la pantalla	Tener la opción de trasladar activada y un objeto seleccionado
Rotar	Movimiento con un solo dedo sobre la pantalla	Tener la opción de rotar activada y un objeto seleccionado
Escalar	Acercar o alejar dos dedos, para disminuir o aumentar el tamaño del modelo respectivamente	Tener un objeto seleccionado

Tabla 3.18: Gestos asociados a las acciones permitidas sobre los modelos

El proceso de detección de gestos se ejecuta en dos fases, recolectar la información e interpretar la información para revisar si cumple con los requisitos de algún gesto soportado por la aplicación. La fase de recolectar información ocurre con la captura del evento del tipo *OnTouchEvent*, este evento se genera por cada secuencia de acciones realizadas por el usuario al tocar la pantalla con uno o más dedos, tales como cambios de posición, presión, usar un dedo nuevo, levantar un dedo, etc. A una secuencia de estos cambios se le denomina gesto. Un gesto comienza cuando se toca por primera vez la pantalla, continua a medida que el sistema rastrea la posición de los dedos y termina cuando los dedos del usuario dejan de tocar la pantalla.

El SDK de Android tiene dos clases para la detección de los gestos más comunes, el *GestureDetector* y el *ScaleGestureDetector*. El primero soporta los gestos de presionar por un tiempo breve, presionar por un tiempo largo, presionar dos veces rápidamente, desplazamiento y el segundo el gesto de separar o acercar dos dedos (relacionado con cambiar el tamaño de las cosas), los cuales cubren las necesidades de la aplicación.

Inicialmente, se implementó la detección de gestos usando las clases mencionadas anteriormente, pero al usar ambas a la vez, no funcionaba bien, debido a que a veces se detectaba un evento como otro por llevar el control de los eventos por separado. Por lo tanto, se decidió usar el *GestureDetector* para el gesto de doble tap y para el resto una implementación propia.

Para la detección de los gestos restantes, el gesto de movimiento de un solo dedo y el gesto acercar o alejar dos dedos, se hace de la siguiente manera, primero se almacena la posición inicial de cada dedo, después, cuando algún dedo cambie de posición, dependiendo de cuantos dedos estén tocando la pantalla, se verifica si el movimiento realizado se puede clasificar como el gesto respectivo.

En el caso de un solo dedo tocando la pantalla (Ver Figura 3.26), la condición para detectar el gesto es que, la suma de las diferencias absolutas de las coordenadas x e y entre la posición actual del dedo y la posición anterior, sea mayor a una tolerancia de movimiento (para obtener este valor se usa el método *ViewConfiguration.getScaledTouchSlop*). Esta condición lo que permite es descartar movimientos aparentes del dedo, como por ejemplo cambiar la presión con que se toca la pantalla.

En el caso de que dos dedos estén tocando la pantalla (Ver Figura 3.27), las condiciones para detectar el gesto son que, la diferencia absoluta de la distancia euclidiana entre ambos dedos en el estado anterior y actual, sea mayor a la tolerancia de movimiento, y que los dedos se estén moviendo en sentidos opuestos (se alejan o se acercan entre sí).

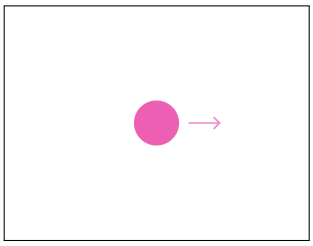


Figura 3.26: Gesto de mover un solo dedo para realizar la acción de trasladar o rotar un objeto

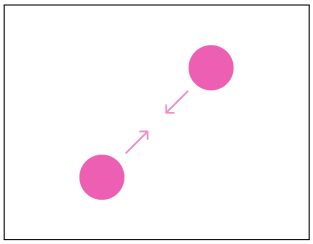


Figura 3.27: Gesto de acercar o alejar dos dedos para realizar la acción de escalar un objeto

Una vez que se detecta algún gesto soportado por la aplicación, la misma debe realizar la acción correspondiente sobre el objeto, ya sea seleccionar, trasladar, rotar o escalar. La acción de seleccionar, permite al usuario indicar a la aplicación el objeto que se desea editar en un futuro. La forma de seleccionar es mediante el gesto de doble tap sobre el objeto, y sí se desea desmarcar la selección el usuario debe hacer doble tap en algún área de la pantalla que no contenga a un objeto. Los pasos que realiza la aplicación para saber que objeto seleccionar son:

1. Renderizar los objetos usando como color de relleno, un numero identificador único para cada objeto y 0 para el fondo.
2. Leer el color del píxel de la pantalla en la posición donde presiono el usuario.
3. Extraer el identificador del color del píxel.
4. Si el identificador es distinto de 0, entonces guardar el identificador como el objeto a ser editado en un futuro.

La acción de trasladar, permite al usuario cambiar la posición de un objeto seleccionado. La manera de hacer esta acción es mediante el gesto de mover un dedo a la posición donde se desea poner el objeto dentro de la vista actual. El trasladar el objeto solo se hace en el plano XY del frustum y la profundidad a donde se coloca el objeto siempre esta a la mitad del frustrum. Para lograr colocar objetos en diferentes profundidades, es necesario que el usuario se mueva de posición para cambiar el valor de la profundidad. Los pasos que realiza la aplicación para mover un objeto de posición son:

1. Convertir la posición de destino de píxeles a coordenadas normalizadas de dispositivo y asignar como valor de profundidad 0.5.
2. Transformar las coordenadas normalizadas de dispositivo a coordenadas de mundo usando la función provista por la librería GLM *unProject*.
3. Asignar la nueva posición al objeto seleccionado.

La acción de rotar, permite al usuario cambiar la orientación de un objeto seleccionado, La forma de realizar esta acción es por medio del gesto de mover un dedo. Idealmente, la rotación seria solamente alrededor de la normal del plano en donde se encuentra en el objeto, pero como esta información no esta disponible, se implemento el rotar como una bola de seguimiento (*trackball*) virtual, la cual permite una rotación con 3 grados de libertad, para que así, el usuario pueda orientar el objeto según el plano donde se encuentre. Los pasaos que realiza la aplicación para rotar un objeto son:

1. Proyectar el punto sobre el trackball virtual.
2. Calcular el ángulo y eje de rotación a partir de las posición inicial y final del dedo proyectadas.
3. Representar la rotación como un quaternion.
4. Aplicar la transformación al objeto seleccionado.

La acción de escalar, permite al usuario modificar el tamaño de un objeto seleccionado. La manera de hacer esta tarea es mediante el gesto de alejar o acercar dos dedos entre sí. Los pasos que realiza la aplicación para cambiar el tamaño de un objeto son:

1. Calcular el factor de escalamiento como el cociente distancia final y la distancia inicial de los dedos.
2. Aplicar la transformación al objeto seleccionado.

3.2.7. Iteración 7: Implementación de un sistema de ayuda

En esta fase se implemento un sistema de ayuda al usuario, que le va sugiriendo que acciones puede hacer o explicándole el significado de los elementos de la interfaz, a medida que va usando la aplicación. La idea es que esta ayuda sea lo menos molesta posible, se muestre cuando el usuario quiera y solo aparezca la primera vez que haya algún elemento o acción desconocida para el usuario. La razón por la cual se implemento esta funcionalidad, es debido a que la curva de aprendizaje de la aplicación es relativamente alta, especialmente el proceso de escanear un espacio, porque requiere que se sigan los pasos al pie de la letra.

El sistema de ayuda esta basado en la especificación de *Material Design* creada por *Google*, específicamente la sección de crecimiento y comunicación (Growth & Communications), que contiene las mejores practicas y componentes para ayudar a los usuarios a entender que pueden hacer con la aplicación, como ayudar a los usuarios a empezar a usar la aplicación, de que forma se pueden introducir nuevas funcionalidades al usuario y como interactuar con los elementos de la interfaz.

El primer paso consiste en enseñarle al usuario como acceder a la ayuda. Cuando el usuario accede por primera vez a la vista de decoración, la aplicación le muestra un botón azul, le explica para que sirve y le sugiere que lo presione con una animación, como se puede apreciar en la Figura 3.28. Cuando el usuario presiona el botón de ayuda, se le muestra una muy breve descripción de que debe hacer o una corta y sencilla explicación sobre el estado actual de la aplicación (Ver Figura 3.29. Una vez que el usuario accede a la ayuda de alguna acción, el sistema guarda el registro para que no se muestre de nuevo en un futuro, incluyendo nuevas actualizaciones de la aplicación.

El sistema de ayuda también funciona como una especie de tutorial, le va sugiriendo las siguientes acciones poco a poco, hasta que el usuario haya explorado todas las funcionalidades de la aplicación. Por ejemplo, si el usuario agrega un objeto por primera vez, la aplicación le dirá como se debe

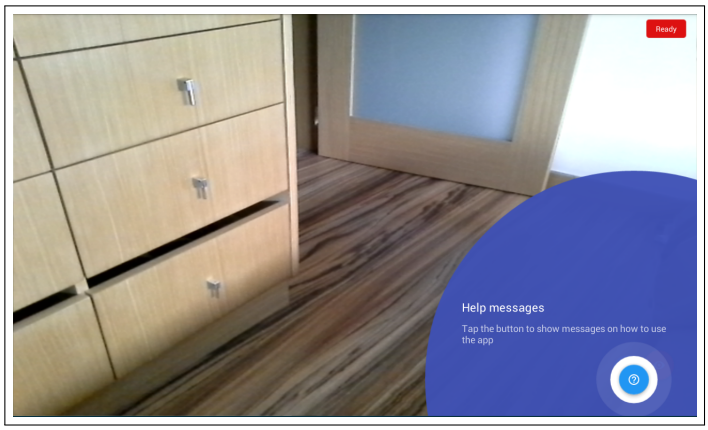


Figura 3.28: Ayuda inicial cuando se entra por primera vez a iniciar el proceso de decoración

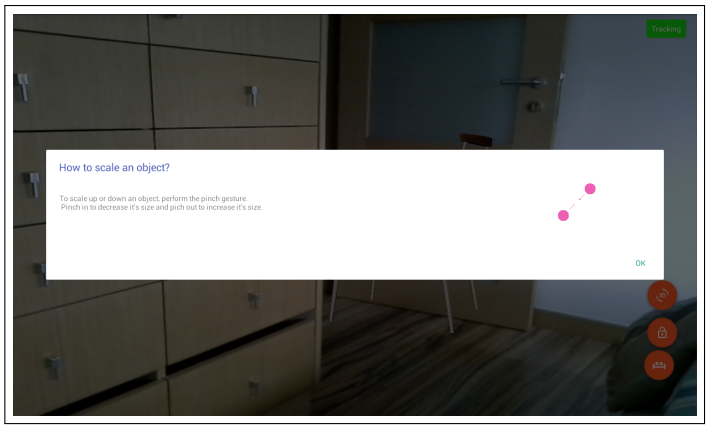


Figura 3.29: Diálogo de ayuda indicando al usuario que gesto debe realizar para escalar un objeto

seleccionar, luego de que lo seleccione, le dirá como se hace para trasladar, rotar y escalar el objeto paso a paso. Un ejemplo gráfico de como sería esto se encuentra en la Figura 3.30.

Figura 3.30: Ejemplo de sugerencia de una acción

Capítulo 4

Pruebas y Resultados

Este capitulo trata sobre las evaluaciones de rendimiento de la aplicación en diferentes dispositivos. Para ello se contó con 8 dispositivos, con características variadas. La mayoría de los dispositivos son teléfonos inteligentes y solo el Galaxy Note Tab 10.1 es una tableta. La lista de los dispositivos junto con sus propiedades esta definida en la Tabla 4.1.

Dispositivo	**Versión Android**	**CPU**	**GPU**	**RAM**
Blu Vivo 5 Mini	6.1	1.3 GHz Quadcore	Mali 400	1 GB
Nexus 6	7.0	2.7 GHz Quadcore	Adreno 420	3 GB
Xperia Z3	6.0	2.5 GHz Quadcore	Adreno 330	3 GB
Nexus 5	6.0	2.3 GHz Quadcore	Adreno 330	2 GB
Galaxy Note Tab 10.1	4.1	1.4 GHz Quadcore	Mali-400 MP4	2 GB
LG Optimus G E970	4.1	1.5 GHz Quadcore	Adreno 320	2 GB
HTC One X	4.2	1.5 GHz Quadcore	ULP GeForce	1 GB
LG G2	4.4	2.26 GHz Quadcore	Adreno 330	2 GB

Tabla 4.1: Lista de dispositivos usados para la evaluación

La evaluación consiste primero calibrar la cámara del dispositivo y luego medir el tiempo promedio en segundos que tarda la aplicación en procesar una imagen para cada estado del ORB-SLAM, durante la modalidad de escanear, variando además la cantidad de características que se extraen de la imagen. El ambiente a escanear es el Centro de Computación Gráfica, para mantener ciertas condiciones constantes durante la prueba.

Al realizar la evaluación, todos los dispositivos se pudieron calibrar satisfactoriamente. Por otra parte, al intentar medir los tiempos promedios, la aplicación no funcionó correctamente en todos los dispositivos. Los teléfonos LG Optimus G E970, el HTC One X y el LG G2, presentaron fallas, los dos primeros solo muestran una pantalla negra cuando se intenta escanear y el último no se pudo lograr que inicializara el sistema SLAM. El resto de los resultados se recopilaron en la Tabla 4.2 y además se elaboraron tres gráficos comparativos (Ver Figuras 4.1, 4.2 y 4.3) con los tiempos de cada dispositivo cuando se extraen 500, 700 y 900 características.

Al observar los tiempos de la Tabla 4.2, lo primero que resalta, es que la aplicación tarda mucho más tiempo de lo ideal para ser una aplicación de tiempo real, que debería estar alrededor de 0.033 segundos, aunque un tiempo de 0.066 segundos se puede considerar bastante bueno, por la

Dispositivo	# Características	T. Inicialización	T. Rastreo	T. Desorientado
Blu Vivo 5 Mini	500	-	-	-
	700	0.723s	0.433s	0.370s
	900	-	-	-
Nexus 6	500	0.532s	-	-
	700	-	-	-
	900	0.610s	0.676s	1.021s
Xperia Z3	500	0.310s	0.211s	0.209s
	700	0.282s	0.283s	0.298s
	900	0.369s	0.292s	0.306s
Nexus 5	500	0.366s	0.282s	0.300s
	700	0.659s	0.398s	0.418s
	900	0.514s	0.339s	0.322s
Galaxy Tab	500	0.339s	0.225s	0.193s
	700	0.339s	0.233s	0.200s
	900	0.426s	0.265s	0.247s

Tabla 4.2: Tiempos promedios para cada estado del ORB-SLAM para diferentes dispositivos y nro de características extraídas

gran cantidad de cálculos que realiza un sistema SLAM. Este resultado es importante a la hora de evaluar la experiencia del usuario, especialmente en la etapa de inicialización del sistema SLAM, donde el usuario debe mover el dispositivo de forma horizontal alrededor de cierto punto, hasta que el sistema obtenga dos imágenes que tengan un paralelaje adecuado. Pero debido a que el tiempo de procesamiento para esta etapa es el más lento de todos, la aplicación va mas lento que los movimientos del usuario, por lo que en muchos casos, inicializar el sistema SLAM es díficil y tedioso. Y al ser este la experiencia inicial para los usuarios, lo más probable es que no sigan usando la aplicación al no poder realizar con éxito este paso.

A pesar de que el tiempo de procesamiento es algo lento, la forma en que esta implementada la aplicación, donde el procesamiento de la imagen se realiza en un hilo aparte, hace que cuando el usuario este decorando una escena estática no sea perceptible. Pero por otra parte, para el caso de los espacios con elementos dinámicos, es evidente que existe un cierto retraso al procesar la imagen, que destruye la sensación de inmersión entre lo virtual y lo real.

Una comparación interesante es entre Xperia Z3 y el Nexus 6, que si se observan los resultados obtenidos para ambos dispositivos, el Nexus 6, tiene un tiempo de procesamiento mayor a pesar de ser un dispositivo con mejor hardware y software. Se podría decir que en cuanto al hardware del Nexus 6, es la siguiente iteración o versión del hardware del Xperia Z3. Este resultado no es lo esperado, sino que debería tener un tiempo menor o igual. Se presume que este resultado puede ser debido a nuevas funcionalidades incorporadas en la versión 7.0 de Android, para aumentar el ahorro de la batería, disminuyendo así el rendimiento de las aplicaciones.

Otro resultado interesante es el del Galaxy Tab comparado con el resto de los dispositivos, los tiempos obtenidos en este dispositivo son similares o mejores que el resto de los dispositivos. Este resultado no es lo esperado, ya que al ser un dispositivo con un hardware de menor desempeño en comparación con el resto, los tiempos obtenidos deberían ser mayores que el resto de los dispositivos, lo que es contrario a los resultados obtenidos. Con lo que se puede concluir, que lo más seguro, es que no se están aprovechando al máximo las capacidades de los dispositivos modernos, como por ejemplo el uso de las instrucciones NEON. También es posible, como se mencionó anteriormente,

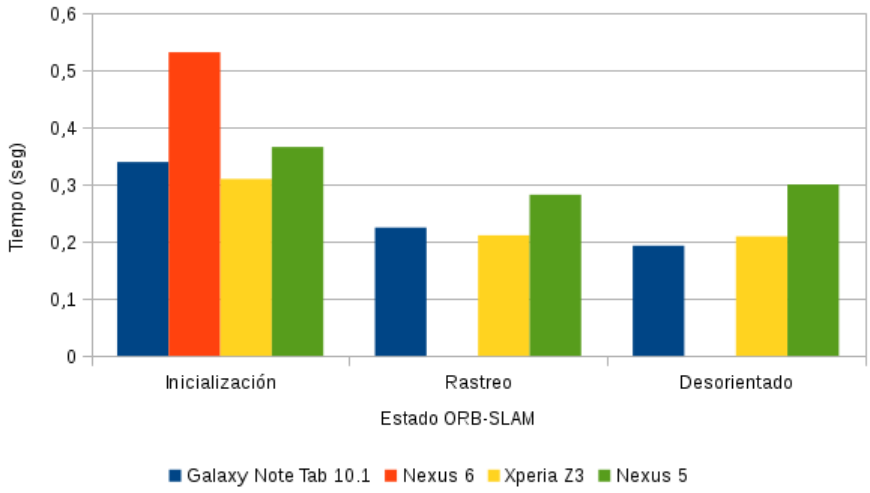


Figura 4.1: Tiempo Promedio en cada estado del ORB-SLAM cuando se extraen 500 características de la imagen

que las últimas versiones de Android contengan funciones para aumentar el ahorro de la batería que disminuyan el rendimiento de la aplicacion.

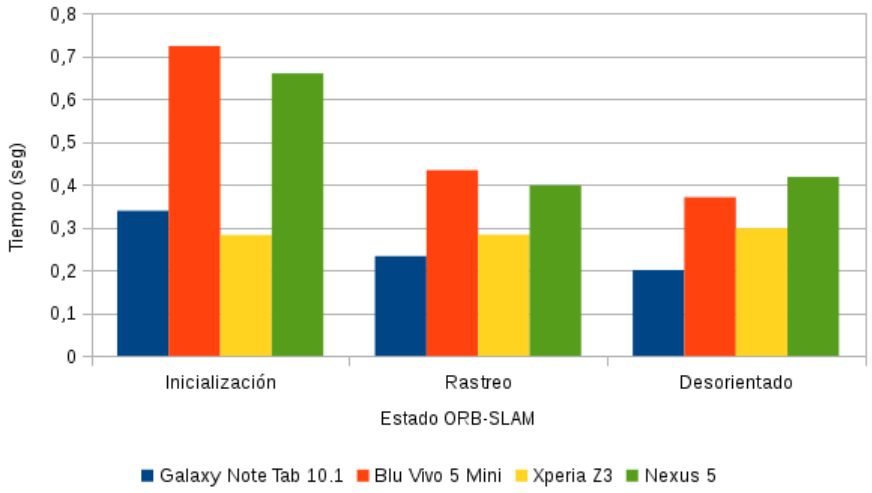


Figura 4.2: Tiempo Promedio en cada estado del ORB-SLAM cuando se extraen 700 características de la imagen

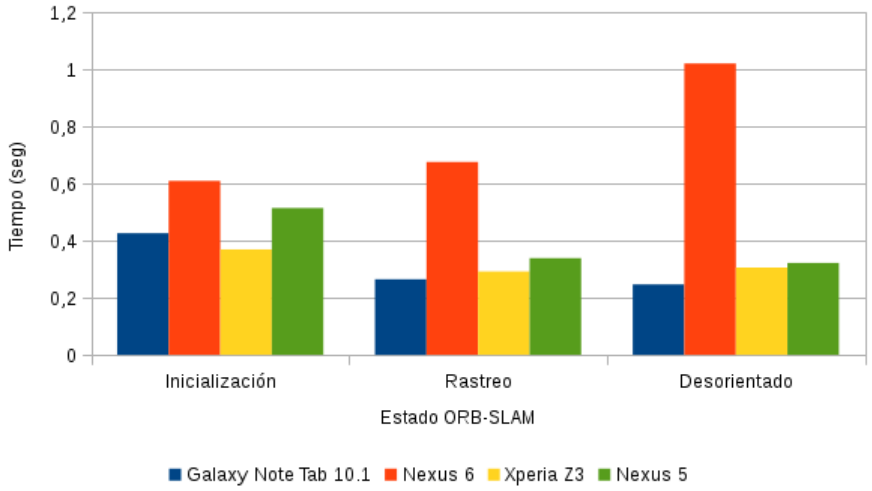


Figura 4.3: Tiempo Promedio en cada estado del ORB-SLAM cuando se extraen 900 características de la imagen

Capítulo 5

Conclusiones y Trabajos Futuros

5.1. Conclusiones

Se cumplió satisfactoriamente los objetivos planteados en el presente Trabajo Especial de Grado, el cual de forma general, consistió en el desarrollo de una aplicación de realidad aumentada para los sistemas móviles Android usando como base ORB-SLAM, para la decoración de espacios interiores.

Al insertar los modelos a la escena y al estar orientados correctamente por el usuario, se alcanza una alta sensación de inmersión. Para mejorar la experiencia y facilitar el trabajo del usuario, la aplicación debería estimar el plano en donde se está colocando el objeto y orientarlo el objeto adecuadamente.

La aplicación funciona en varios dispositivos correctamente, aunque en algunos de ellos, la aplicación aún es demasiado lenta o tiene ciertas fallas que afecta la usabilidad y la experiencia del usuario.

La arquitectura modular de la aplicación y separación de responsabilidades permite incorporar mejoras de una manera sencilla sin afectar gran parte del sistema.

El tiempo que tarda en procesar una imagen por el sistema ORB-SLAM aún es demasiado lento, pero se considera que aún queda espacio para optimizar este proceso, ya sea, haciendo un mejor uso de la memoria, evaluando como es el acceso a la cache, paralelizando las los cálculos con el conjunto de instrucciones NEON (SIMD) o con la tecnología de Android, RenderScript.

5.2. Trabajos Futuros

Como recomendaciones para trabajos futuros con la finalidad de extender y mejorar el trabajo realizado en esta aplicación, se proponen los siguientes trabajos:

- Optimizar la librería ORB-SLAM para hacer un mejor uso de los recursos de los dispositivos móviles. Como por ejemplo el uso de las instrucciones NEON, el uso de la tecnología RenderScript, disminuir el uso de memoria principal o evaluando el patrón de acceso a la cache del CPU.

- Hacer una reconstrucción 3D del espacio a partir del mapa generado por ORB-SLAM.
- Mejorar el módulo de ORB-SLAM para la estimación más precisa de la escala del espacio usando elementos con tamaños conocidos.
- Diseñar y evaluar estrategias para mejorar el proceso de escanear un espacio y hacerlo más robusto y fácil para los usuarios.
- Expandir el ORB-SLAM para que use los puntos de fuga y/o líneas.

Bibliografía

[1] Michael Calonder et al. "BRIEF: Binary robust independent elementary features". En: (2010).

[2] Rublee Ethan et al. "ORB: an efficient alternative to SIFT or SURF". En: (2011).

[3] *Augmented Reality*. Feb. de 2016. URL: `https://en.wikipedia.org/wiki/Augmented_reality`.

[4] R. T. Azuma. "A survey of Augmented Reality". En: *Teleoperators and Virtual Environments* (1997).

[5] Tim Bailey y Hugh Durrant-Whyte. "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art". En: ().

[6] C. Cadena y col. "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age". En: *IEEE Transactions on Robotics* 32.6 (2016), págs. 1309-1332.

[7] *Computer Vision*. Feb. de 2016. URL: `https://en.wikipedia.org/wiki/Computer_vision`.

[8] Alan B. Craig. *Understanding Augmented Reality Concepts and Applications*. Elsevier, 2013.

[9] D. Gálvez-López y J. D. Tardós. "Bags of binary words for fast place recognition in image sequences". En: (2012).

[10] Iryna Gordon y David G. Lowe. "Scene Modelling, Recognition and Tracking with Invariant Image Features". En: (2004).

[11] Chris Harris y Mike Stephens. "A combined corner and edge detector". En: (1988).

[12] Zisserman A." "Hartley R. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[13] Andrew J. Davison Hauke Strasdat J.M.M. Montiel. "Visual SLAM: Why Filter?" En: ().

[14] Luc Van Gool Herbert Bay Tinne Tuytelaars. "SURF: Speeded Up Robust Features". En: (2006).

[15] *Khronos OpenGL ES 2.X*. Mar. de 2017. URL: `https://www.khronos.org/opengles/2_X/`.

[16] G. Klein y D. Murray. "Parallel tracking and mapping for small AR workspaces". En: (2007).

[17] "Reinhard Klette". *Concise Computer Vision An Introduccion into Theory and Algorithms*. Springer, 2014.

[18] Cordelia Schmid Krystian Mikolajczyk. "A performance evaluation of local descriptors". En: (2005).

[19] R. Kuemmerle y col. "g2o: A general framework for graph optimization". En: (2011).

[20] David G. Lowe. "Object Recognition from Local Scale-Invariant Features". En: (1999).

[21] *Material Design*. Sep. de 2016. URL: `https://material.io/`.

[22] *Material Design*. Oct. de 2016. URL: `https://material.google.com`.

[23] C. Mei, G. Sibley y P. Newman. "Closing loops without places". En: (2010).

[24] R. Mur-Artal y J. D. Tardós. "Fast relocalisation and loop closing in keyframe-based SLAM". En: (2014).

[25] Raúl Mur-Artal. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". En: (2015).
[26] "Richard E. Woods" "Rafael C. Gonzalez". *Digital Image Processing*. Prentice Hall, 2007.
[27] E. Rosten y T. Drummond. "Fusing points and lines for high performance tracking". En: (2005).
[28] E. Rosten y T. Drummond. "Machine learning for high-speed corner detection". En: (2006).
[29] Sanni Siltanen. "Theory and applications of marker-based augmented reality". En: *VTT Science* (2012).
[30] "J. Solem". *Programming Computer Vision with Python*. O'Reilly, 2012.
[31] H. Strasdat, J. M. M. Montiel y A. J. Daviso. "Scale drift aware large scale monocular SLAM". En: (2010).
[32] H. Strasdat y col. "Double window optimisation for constant time visual SLAM". En: (2011).
[33] Zhengyou Zhang. "A Flexible New Technique for Camera Calibration". En: (1998).