Escuela Politécnica Superior de Jaén
Máster Universitario en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Trabajo Fin de Máster

GUÍA DE PERSONAS INVIDENTES MEDIANTE CONSTRUCCIÓN DE MAPAS SÓNICOS DEL ENTORNO

Alumno/a: **Luque Luque, Adrián**

Tutor: Rueda Ruiz, Antonio Jesús
Dpto.: Departamento de Informática

Tutor: Fuertes García, José Manuel
Dpto.: Departamento de Informática

Julio, 2018

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rueda Ruiz y Don José Manuel Fuertes García, tutores del Trabajo Fin de Master titulado: *Guía de personas invidentes mediante construcción de mapas sónicos del entorno*, que presenta Adrián Luque Luque, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, 4 de Julio de 2018

El alumno:

LUQUE LUQUE ADRIAN - 77324298T

Firmado digitalmente por LUQUE LUQUE ADRIAN - 77324298T
Fecha: 2018.07.03 20:56:13 +02'00'

Adrián Luque Luque

Los tutores:

Firmado digitalmente por RUEDA RUIZ ANTONIO JESUS - 33395227D
Fecha: 2018.07.04 00:31:43 +02'00'

Antonio Jesús Rueda Ruiz

Firmado digitalmente por FUERTES GARCIA JOSE MANUEL - 26478036E
Fecha: 2018.07.04 09:59:36 +02'00'

José Manuel Fuertes García

AGRADECIMIENTOS

Este documento ha sido el fruto de un gran esfuerzo y del que estoy realmente orgulloso por la solución alcanzada.

Por ello, me gustaría dar las gracias a mis compañeros (conejillos de indias), profesores y familia que me han apoyado durante estos meses. En especial a mis tutores Antonio y José Manuel por su ayuda.

Me gustaría también agradecer a mi pareja por soportarme y apoyarme durante los días de trabajo.

GRACIAS A TODOS

Índice

Índice de ilustraciones

Índice de tablas

1. INTRODUCCIÓN

Según la Organización Mundial de la Salud [1], la cifra mundial estimada de personas con una discapacidad visual severa es de 253 millones de personas, 36 millones con ceguera total y 217 millones con una discapacidad de moderada a grave.

En los últimos años, han aparecido una gran variedad de dispositivos destinados a la ayuda de estas personas [2] [3] [4]. Muchos iniciados por grupos de investigación conscientes de sus necesidades. Muy pocos han sido plasmados en productos comerciales y los que lo han conseguido, son excesivamente caros para llegar al público general o a países en vías de desarrollo [5].

El objetivo que persigue este proyecto es, construir un prototipo de ayuda, que permita una mayor autonomía a las personas con ceguera, para desenvolverse en entornos desconocidos o cambiantes, de bajo coste. Este accesorio no sustituirá otros cuyo uso esta estandarizado, como puede ser el bastón o perros-guía, si no que los complementará en ciertas situaciones.

En ciertas situaciones, el uso de bastones o perros-guía podría no ser posible o suficiente. Por ejemplo, el bastón no es capaz de detectar obstáculos por encima del nivel de la cintura. En España, por ley es posible entrar en cualquier establecimiento o zona, a la que normalmente no se permite la entrada a animales, como playas, servicios públicos de transporte u hoteles.

Pero hay situaciones, en las que el desconocimiento o malas intenciones, pueden ocasionar conflictos, como el caso reciente ocurrido a un atleta ciego [6], al que le prohibieron la entrada a dos taxis, por ir acompañado con su perro lazarillo.

Dada la ambición del proyecto y el límite temporal de 300 horas de un proyecto de fin de máster, no se pretende que el dispositivo desarrollado sea una versión final. Pero si se podrán validar los componentes y algoritmos desarrollados, durante el transcurso del mismo.

1.1. Objetivos

Este TFM pretende investigar en profundidad y desarrollar una idea que se basa en la generación de mapas sónicos a partir del entorno utilizando la información proporcionada por una cámara de profundidad, con aplicaciones en la guía de personas invidentes durante su desplazamiento. Para generar este mapa sónico se utilizarán sonidos procedurales con distintas intensidades, tonos, texturas y orientación espacial utilizando las posibilidades del sonido estéreo.

Los objetivos específicos para este trabajo de fin de máster son:

- Diseñar un algoritmo que permita construir un mapa sónico del entorno a partir de la información obtenida desde una cámara de profundidad.
- Desarrollar un prototipo que implemente el algoritmo diseñado en un dispositivo móvil.
- Evaluar las posibilidades del sistema desarrollado con usuarios reales.
- Realizar una memoria de investigación describiendo los fundamentos teóricos de la técnica desarrollada, el diseño e implementación del prototipo y los resultados de las pruebas de evaluación.

1.2. Requisitos

Para la realización de estos objetivos se han definido una serie de requisitos que se deben de cumplir, se distinguirán entre requisitos funcionales y no funcionales.

1.2.1. Requisitos funcionales

Los requisitos funcionales, son las capacidades que debe de tener el sistema.

- El procesamiento de imágenes, tiene que procesar las imágenes lo suficientemente rápido para adaptarse a la velocidad de movimiento de una persona.
- El dispositivo debe de ser lo más discreto posible, en el caso de necesitar cables, se intentarán ocultar en la medida de lo posible.
- El sistema tiene que poder usarse de forma natural, sin que moleste o interfiera en otros sentidos.
- La complejidad del estímulo auditivo deberá de ser controlable, para permitir una curva de aprendizaje correcta.

1.2.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos criterios sobre la calidad del producto, usabilidad, rendimientos, estética, etc.

- La batería del sistema debe de soportar un día normal de uso, ya sea por la propia capacidad de la batería, porque soporte el cambio de baterías, o porque permita la carga con baterías externas, mientras se esté usando.
- Los algoritmos desarrollados deberán de poder portarse a otras plataformas, con los mínimos cambios posibles. Esto permitirá adaptarse a nuevos y mejores componentes que aparezcan en el mercado.

1.3. Estructura y planificación del proyecto

En este apartado, se discutirá sobre la división de tareas del proyecto, la asignación de tiempos a cada tarea y por último, se mostrará un diagrama de Gantt [7], para facilitar la comprensión del reparto de tiempo designados a cada tarea.

1.3.1. División de tareas

En la siguiente página se puede ver, Ilustración 1.1, la separación de tareas. Esta división permite, desgranar el trabajo en cargas de trabajo más

simples, pero lo suficientemente complejas para ser tenidas en cuenta como hitos o milestones.

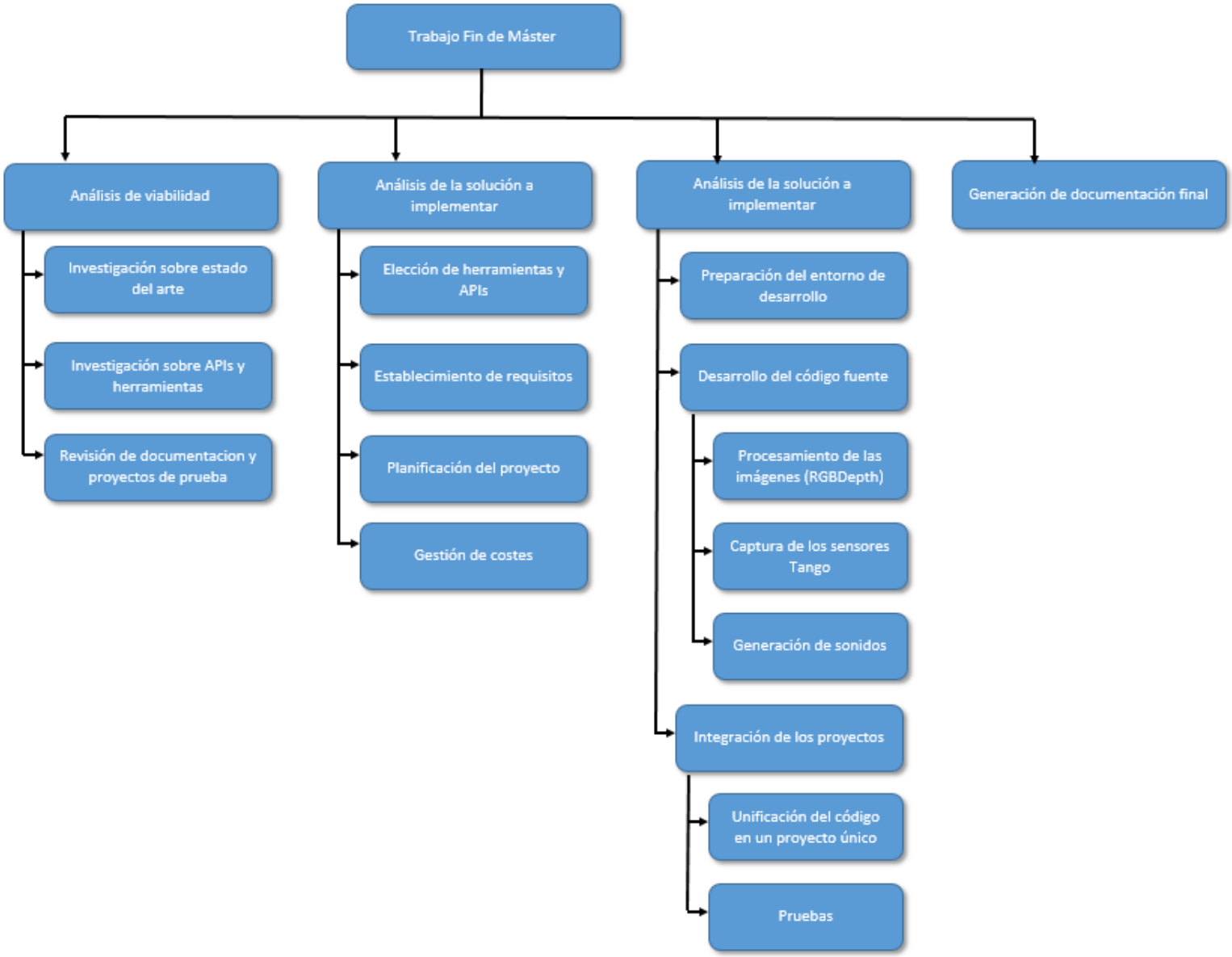


Ilustración 1.1 – División de tareas

1.3.2. Calendario

Según la carga de créditos del Trabajo de Fin de Grado, el tiempo de dedicación es de 300 horas. Dado que el alumno está actualmente trabajando, se han definido días de trabajo de 3 horas, lo que resulta en un total de 100 días, o 20 semanas, de trabajo.

Se ha elaborado un calendario, Tabla 1.1, teniendo en cuenta los conocimientos previos del alumno, relativos a las áreas desarrolladas en el proyecto, gestión de proyectos, procesamiento de la información visual, algoritmos geométricos y desarrollo de aplicaciones móviles.

Nombre de la tarea	Días (3 horas)	Horas
Análisis de viabilidad		
Investigación sobre estado del arte	10	30
Investigación sobre APIs y herramientas	5	15
Revisión de documentacion y proyectos de prueba	10	30
Análisis de la solución a implementar		
Elección de herramientas y APIs	2	6
Establecimiento de requisitos	2	6
Planificación del proyecto	3	9
Gestión de costes	2	6
Desarrollo		
Preparación del entorno de desarrollo	1	3
Desarrollo del código fuente		
Procesamiento de las imágenes (RGBDepth)	25	75
Captura de los sensores Tango	10	30
Generación de sonidos	10	30
Integración de los proyectos		
Unificación del código en un proyecto único	5	15
Pruebas	5	15
Generación de documentación final	10	30
Total	**100**	**300**

Tabla 1.1 – Duración tareas

1.3.3. Diagrama de Gantt

Como parte final de la planificación temporal del proyecto, se ha creado un diagrama de Gantt [7], el cual consiste en gráfica, donde el tiempo se representa en el eje horizontal y las tareas en el eje vertical. A pesar de su aparente sencillez, se pueden añadir componentes más complejos, como dependencias entre tareas, posibilidad de tareas en paralelo, o restricciones en base a los recursos necesarios de cada tarea. Estos recursos pueden ser trabajadores, elementos software o hardware, o monetarios.

En la Ilustración 1.2, podemos ver el diagrama de Gantt del proyecto. El proyecto se está llevando a cabo por un solo alumno, por lo que las tareas que

podrían paralelizarse, como, “Captura de los sensores Tango” y “Generación de sonidos”, han tenido que dibujarse de forma secuencial.

Así mismo, la generación de documentación se ha programado para el final, aunque se ha desarrollado durante la duración del proyecto.

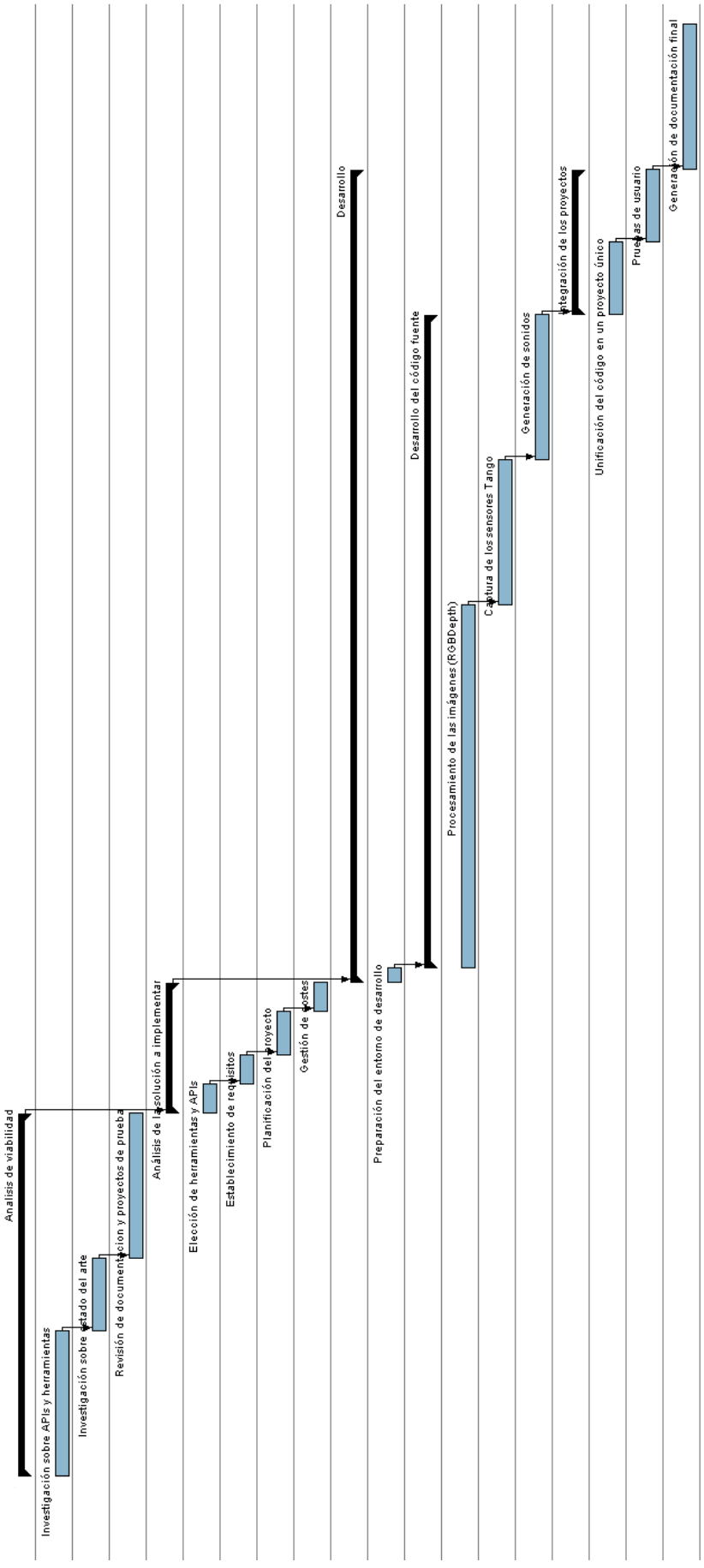


Ilustración 1.2 – Diagrama de Gantt

1.4. Coste del proyecto

A continuación, se desglosarán los costes del proyecto, en contratación de personal y gastos en hardware y software.

1.4.1. Recursos humanos

El proyecto se desarrollará por un alumno, que desempeñara los diferentes roles que serían necesarios en un proyecto real. En concreto:

- Gestor de proyectos, su tarea será la de análisis previo al desarrollo, la generación de la documentación asociada al proyecto y la redacción de la memoria final.
- Analista, su trabajo se centrará en la tarea de "Análisis de la solución a implementar" y participará, en menor medida en la fase de desarrollo y pruebas.
- Programador, se encargará principalmente del desarrollo del prototipo y de la revisión del prototipo, una vez terminadas las pruebas.

En la Tabla 1.2, se desglosa la asignación temporal para cada rol en cada una de las tareas principales y en la , se detallan los salarios de cada rol participante en el proyecto.

Tareas	Investigación	Análisis	Desarrollo	Documentacion	Total
Gestor	75	10	0	30	115
Analista	0	17	38	0	55
Programador	0	0	130	0	130

Tabla 1.2 - Distribución de tareas según rol

Rol	**Salario Anual**	**Salario por hora**	**Ajuste SS**	**Horas**	**Coste final**
Gestor	30000€	14.20€	18.89€	115	2172.35€
Analista	24000€	11.36€	15.11€	55	831.05€
Programador	20000€	9.47€	12.60€	130	1638.00€
				Total	**4641.40€**

Tabla 1.3 - Salarios desglosados

Los salarios anteriores han sido calculados mediante búsquedas de ofertas de trabajo, en portales web, para puestos con responsabilidad similar a la

descrita en los roles. En el salario por hora se han tenido en cuenta los siguientes datos:

- 12 pagas anuales.
- 22 días laborales al mes.
- 8 horas diarias de trabajo.
- 33% de los impuestos de la seguridad social.

1.4.2. Coste hardware y software

A continuación, se desglosará el coste hardware y software del proyecto, empezando por los dispositivos usados en el proyecto.

Para calcular su coste relativo, hemos tenido en cuenta una vida útil de 5 años y 20 semanas de uso específico para el proyecto, por lo que el valor amortizado corresponde a un 8% del valor inicial.

Los dispositivos finalmente usados durante el desarrollo del prototipo y listados en la Tabla 1.4, son, un ordenador de sobremesa, un kit de desarrollo Tango y unos auriculares de conducción ósea. Las principales características del ordenador son, procesador i7-7600, 16GB de ram y una gráfica dedicada nVidia 1070, con 8GB de vram.

Dispositivo	Coste	Coste asociado
Ordenador de desarrollo	1350€	108.00€
Tablet Tango	500€	40.00€
Auriculares de conducción ósea	85€	6.80€
	Total	**154.8€**

Tabla 1.4 - Coste hardware

Teniendo en cuenta que el proyecto se ha realizado en un entorno académico, todo el software usado para llevar a cabo el proyecto ha sido gratuito. El único software que tiene una licencia de pago es el SDK de audio [8], y solo para aplicaciones que superen el medio millón de instalaciones. Dado el reducido público del prototipo, consideramos un coste cero.

Listado de software usado:

- Windows 10. [9]

- Visual Studio 2017. [10]
- Android Studio. [11]
- Dataset rgbd (freiburg1). [12]
- Superpowered Audio SDK. [8]
- OpenOffice. [13]
- GanttProject. [14]
- Audacity. [15]

1.4.3. Coste total

Una vez que hemos definido los gastos destinados a cada componente del proyecto, podemos calcular el coste total, Tabla 1.5.

Componente	Coste
Recursos humanos	**4641.40€**
Hardware	**154.80€**
Software	**0.00€**
Total	**4881.20€**

Tabla 1.5 - Coste total

1.5. Estructura de la memoria

A continuación, se muestra un resumen de los siguientes capítulos de la memoria:

En el capítulo 2, realizaremos un estudio extensivo de dispositivos de ayuda a las personas con ceguera, algoritmos existentes para la segmentación de imágenes rgbd y sistemas de generación de audio 3D procedural. También se detallarán y elegirán, los distintos componentes hardware y software disponibles para la realización del prototipo.

Una vez elegidos los algoritmos, librerías y dispositivos, en el capítulo 3 se detallarán las dos versiones desarrolladas, una versión para escritorio, y otra para dispositivos Tango [16]. Al final del mismo, probaremos el dispositivo con usuarios para demostrar la validez del mismo.

Terminando con el capítulo 4, donde se expondrán las conclusiones y trabajo futuro.

Por último, se incluye la bibliografía consultada, el manual de usuario y desarrollador, y una descripción de los anexos que acompañan a esta memoria.

2. Análisis de viabilidad

En este capítulo se expondrán los resultados de la investigación llevada a cabo sobre el estado del arte y, herramientas software y hardware disponibles para llevar a cabo el prototipo.

2.1. Estado del arte

La investigación del estado del arte se ha dividido en dos secciones, ayudas de electrónicas de viaje o electronics travel aids, ETA, segmentación de imágenes rgbd y generación de audio 3D procedural.

2.1.1. Electronics travel aids

En la literatura aparecen varias reviews sobre dispositivos de ayuda a personas con ceguera, una de las más completas es la escrita por Dimitrios Dakopoulos y Nikolaos G. Bourbakis [17]. En ella aparecen características deseables [18], prototipos de diferentes grupos de investigación y métricas de evaluación.

Características que deberían incorporar los dispositivos de ayuda:

- Deberán reconocer los obstáculos presentes desde el nivel del suelo hasta la altura de la cabeza del usuario.
- Deberán incluir información sobre las texturas o discontinuidades de las superficies.
- Detección de los objetos que bordean la trayectoria del usuario para facilitar su avance.
- Reconocimiento de obstáculos, incluyendo dirección y distancia.
- Identificación y localización de símbolos o señales viales.
- La información debería poder ayudar al usuario a crear un mapa mental del entorno.
- Adicionalmente se pueden tener en cuenta otros aspectos:
    - Ergonómico y compactos
    - Interfaz mínima y natural.
    - Resultados repetibles y confiables.

De los 22 prototipos mencionados en la review, hay algunos similares al prototipo que pretendemos desarrollar, ya sea en la forma en la que se usan los sensores, o en la que se manda la información espacial al usuario. En concreto:

2.1.1.1. vOIce

El proyecto vOIce [2] nació con el argumento de que el sistema auditivo, podía interpretar sonidos extremadamente complejos y cambiantes. El sistema, Ilustración 2.1, compuesto por una webcam integrada en unas gafas, un portátil y unos auriculares, transformaban la señal visual de forma reversible y sin filtros.

Ilustración 2.1 - vOIce

Los usuarios que lo probaron indicaron que el prototipo era prometedor, pero que era necesaria gran entrenamiento previo. Más adelante, implementaron una solución para dispositivos móviles, que permitía usar la cámara del móvil y unos auriculares.

2.1.1.2. EPFL Project

En la École Polytechnique Fédérale de Lausanne [3], desarrollaron un sistema consistente en 4 sensores de ultrasonidos, 8 motores de vibración, una placa controladora y dispositivo de calibración, una PDA, Ilustración 2.2. Tanto los sensores, como los motores, están colocados a la altura de los hombros, e integrados en la ropa del usuario.

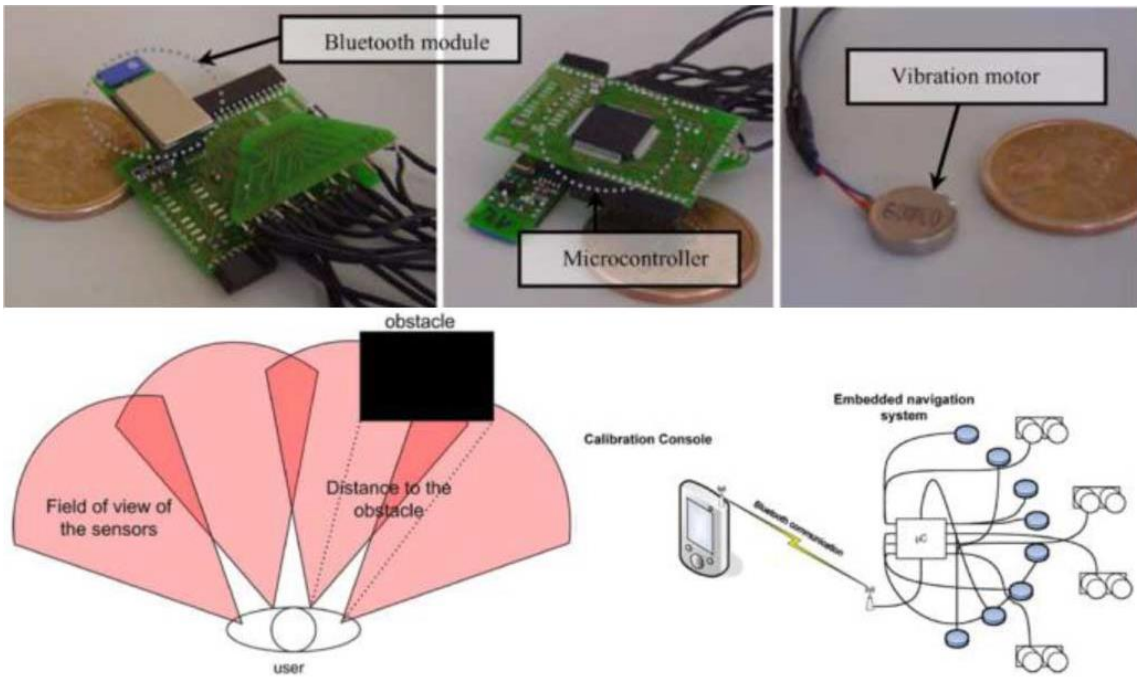


Ilustración 2.2 - Dispositivo EPFL

Podemos considerar que el dispositivo es barato, compacto y portable, ya que solo es necesaria una prenda de ropa modificada y una pequeña placa controladora. La PDA, se conecta por bluetooth y solo es necesaria en las etapas de calibración inicial y entrenamiento.

Los contras de este proyecto son, que solo es posible detectar obstáculos a la altura de los hombros, por lo que no es una representación adecuada del espacio 3D y la falta pruebas en personas ciegas.

2.1.1.3. Electron-Neural Vision System

Desarrollado en la universidad de Wollongong, el sistema ENVS [4], integra las funciones de un dispositivo de ayuda al viaje, complementado con un sistema de navegación GPS. Lo que permite evitar obstáculos cercanos, usando un sistema estéreo de cámaras y seguir una ruta, mediante el GPS y una brújula digital.

La información de obstáculos y rutas, se envía a un sistema de electro estimulación nervioso (TENS), consistente en una unidad controladora y un par de guantes. Las pruebas realizadas en personas ciegas fueron satisfactorias, con un entrenamiento mínimo de una hora, eran capaces de reconocer obstáculos y seguir una ruta GPS predefinida.

Hay dos inconvenientes en usar este dispositivo, el primero, es portable, pero no compacto, es necesario un ordenador portátil y como se puede en la Ilustración 2.3, el sistema ENVS al completo, es muy voluminoso. Y el segundo, no es capaz de detectar el suelo, o pequeños escalones.

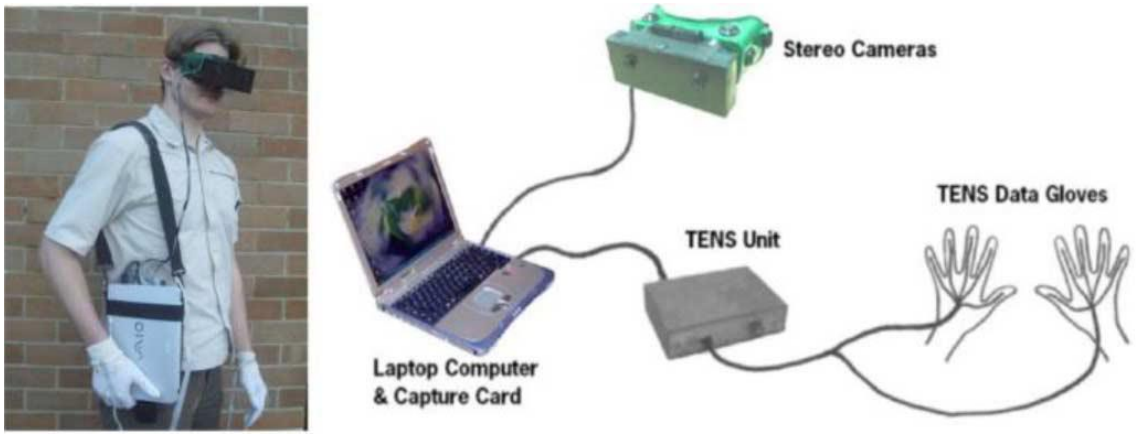


Ilustración 2.3 - Sistema ENVS

Los sistemas mencionados anteriormente eran prototipos, desarrollos de investigación académica y que nunca se formalizaron en un producto final comercial. El siguiente dispositivo es un producto final.

2.1.1.4. Eyesynth

Eyesynth [5] es un dispositivo de origen español, creado por un equipo de 13 personas, entre desarrolladores, técnicos de sonido, tester, etc. Actualmente se encuentra en fase de reserva, por 499€, con un precio final de 2420€, muy superior al que perseguimos con nuestro prototipo.

Al comienzo del desarrollo del TFM, Eyesynth se encontraba en desarrollo, contaba con dos cámaras y unos auriculares de conducción ósea, integrados en las gafas y un ordenador portátil en una mochila, en la Ilustración 2.4 se puede ver este prototipo.

Ilustración 2.4 - Prototipo Eyesynth

A fecha de junio de 2018, han avanzado lo suficiente como para abrir la reserva del producto, y aunque no hay fotos reales, sí que muestran, renders del supuesto producto final, Ilustración 2.5, características técnicas y la inclusión de dos modos de funcionamiento.

Ilustración 2.5 - Render Eyesynth

Los modos de funcionamiento anunciados, son el modo rastreo, que genera un patrón de sonido usando una línea vertical en el centro de visión, y un modo panorámico, que genera sonidos a partir del campo de visión completo, Ilustración 2.6. En principio el modo panorámico, podría parecerse en funcionalidad a nuestro prototipo, filtrando los elementos más relevantes de la

escena, o ser similar al dispositivo vOIce, usando la información visual al completo.

Ilustración 2.6 - Modos de Eyesynth

Después de revisar toda la información disponible de Eyesynth, hay dos cosas que creemos que pueden llevar a confusión. Mencionan el audio coclear, en vez, del término auriculares de conducción ósea, usado en la literatura, la palabra coclear puede llevar a confusión, por su relación a los implantes cocleares realizados en personas con problemas auditivos. En la foto que acompaña la definición del modo panorámico, se puede ver un ángulo de visión de 180º, cuando en las especificaciones técnicas indican un FOV de 76º.

Las dos cuestiones anteriores no deben ensombrecer el gran trabajo del grupo responsable de Eyesynth, ya que a nuestro juicio han conseguido un producto final muy completo, compact, y confiable. El mayor inconveniente que podemos achacarle es el alto precio.

2.1.2. Segmentación de imágenes RGBDepth

El principal componente, y sobre el que se apoya el resto del proyecto, es la segmentación de imágenes rgbd capturadas por la cámara. La segmentación permitirá la sintetización de sonidos que representen la escena de manera más o menos fiel.

Sobre la segmentación de imágenes gran variedad de artículos, usando enfoques diferentes, por ejemplo en P.F. Felzenszwalb et al. [19], interpretan una imagen como un grafo, en el que cada color de un pixel es un componente. La segmentación se realiza buscando grupos de vecinos cuyas componentes estén dentro de un umbral.

La implementación del anterior algoritmo sería relativamente sencilla, la profundidad de cada pixel se trata como una componente más. Los resultados del artículo son buenos, en entornos relativamente controlados, pero a pesar de ser un algoritmo eficiente, no alcanzaría la tasa de segmentación de imágenes, necesaria para el tiempo real.

El algoritmo propuesto por Zhuo Deng et al. [20], muestra resultados superiores al 90%, en el mismo artículo se compara con otros de precisión similar, en la segmentación de 654 imágenes rgbd.

El precio a pagar por esa precisión, es el tiempo de cálculo. En un ordenador de sobremesa potente, 8 núcleos a 3.1GHz y 16GB de ram, el más rápido procesa una imagen en 7 segundos de media, el resto tarda entre 60 y 300 segundos.

Una parte de los algoritmos de segmentación, parte de la detección de planos para la segmentación de escenas. El artículo de Can Erdogan et al [21], describe un complejo y preciso, pero rápido, método de segmentación usando planos. En la Ilustración 2.7, se puede ver a la izquierda la segmentación usada como referencia, y a la derecha el resultado del algoritmo.

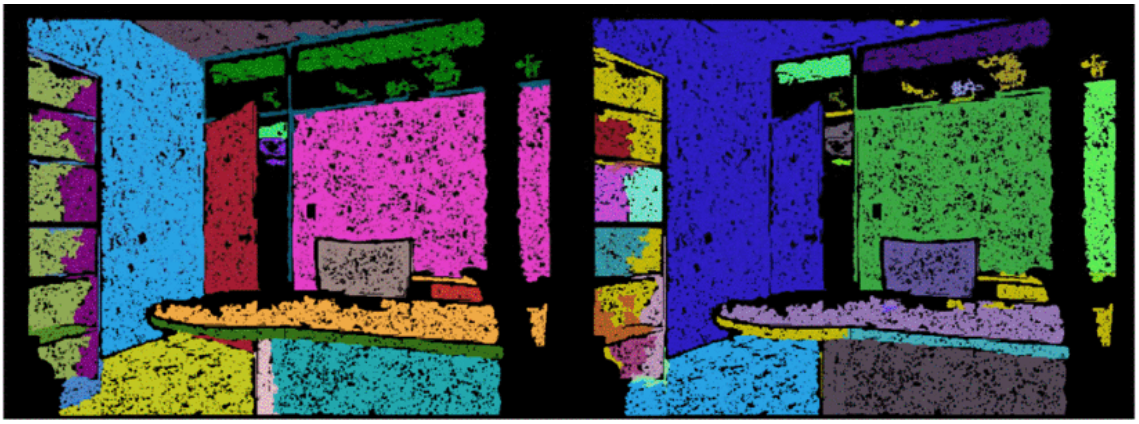
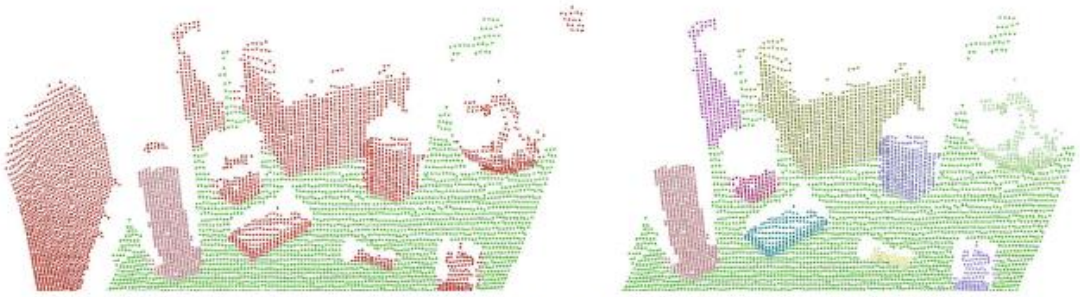


Ilustración 2.7 - Algoritmo de Can Erdogan

Los autores reconocen que no han conseguido alcanzar el tiempo real, pero que, refinando el sistema podrían alcanzarlo. Tomando como referencia este artículo, encontramos otros tres con autores en común [22] [23] [24].

El primero [22], fija las bases teóricas de un algoritmo de detección de planos en imágenes rgbd, posteriormente usa los planos como una máscara de recorte, para segmentar el resto de la escena usando segmentación por distancias. En la Ilustración 2.8, se puede, a la izquierda, el plano detectado en verde y los puntos a los que se le aplicará una segmentación por profundidad.



Resolution	Processing time	Avg. Deviation	Considerable deviations
640 × 480 (VGA)	61.84 ± 7.73 ms	11.75 ± 3.02 deg	roughly 2%
320 × 240 (QVGA)	15.08 ⊥ 2.19 ms	12.39 ⊥ 3.81 deg	roughly 1.2%
160 × 120 (QQVGA)	4.32 + 0.57 ms	9.32 + 2.44 deg	roughly 1%

Ilustración 2.8 - Segmentación basada en planos

Para conseguir tiempos tan bajos y buena precisión, hacen uso de nubes de puntos organizadas y de imágenes integrales, para optimizar el algoritmo de cálculo de normales en cada punto.

Las nubes de puntos organizadas, son aquellas que se estructuran de tal forma que es posible saber el vecino de cada punto, sin necesidad de calcular distancias, como una imagen rgbd.

Una imagen integral, es una matriz en la que cada posición se calcula como la suma de valores del área superior izquierda, de la matriz original. Esta estructura permite calcular la suma de los valores de un área en tiempo constante, O(1). En la Ilustración 2.9, se puede ver como el cálculo de un área se reduce a una suma de cuatro valores, la esquina inferior derecha, más la esquina superior izquierda, menos la suma de las dos esquinas restantes.

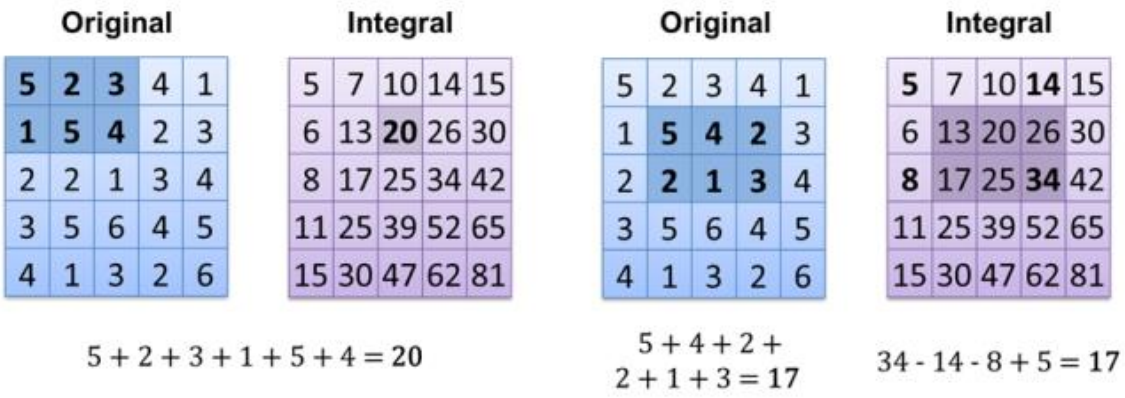


Ilustración 2.9 - Imagen integral

Para obtener la normal en un punto, se calcula el vector perpendicular a un plano, representado como un conjunto fijo de puntos vecinos, Ilustración 2.10. Una vez calculadas las normales, se comparan los pixeles/puntos vecinos para agruparlos en planos.

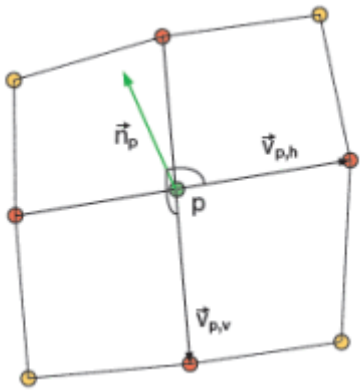


Ilustración 2.10 - Normal de punto

El segundo es una ampliación a la propuesta anterior, que sí alcanza el tiempo real. Esta mejora es presentada por el equipo de S. Holzer [23], que introducen una mejora principal, el número de vecinos escogido para el cálculo de las normales en cada punto es dinámico.

Para calcular el número de vecinos de cada punto, primero es necesario una matriz que indique la distancia de ese punto a una frontera. La frontera se define, como una zona donde la diferencia de distancia entre un pixel y algún vecino es mayor a un umbral. Este umbral depende de la distancia al sensor y

de una constante, que varía entre diferentes sensores, al aumentar la distancia aumenta el ruido, por lo que el umbral aumenta, para detectar falsas fronteras.

Una vez calculada la matriz de distancias punto-frontera, a cada punto se le aplica la función siguiente, Ilustración 2.11. Donde T(m, n), es el valor en la matriz anterior y B(m, n) es función que devuelve un número de vecinos dependiendo de la distancia del punto, y una vez más, de una constante asociada al sensor. En el apartado de desarrollo se explicarán las variables para la tablet Tango y como se han calculado.

$$\mathcal{R}(m,n) = \min(\mathcal{B}(m,n), \frac{\mathcal{T}(m,n)}{\sqrt{2}}),$$

Ilustración 2.11 - Cálculo del número de vecinos

La fórmula anterior devuelve el mínimo entre los vecinos por distancia y los vecinos por cercanía a una frontera, esto ayuda a evitar normales erróneas en las fronteras y a suavizar el error en zonas alejadas, donde la cámara es menos precisa. En la Ilustración 2.12, se puede visualizar el resultado de la fórmula, el número de vecinos, en este caso, tiene un rango de 3 a 8, de más oscuro a más claro, el negro representa puntos no capturados por el sensor.

Ilustración 2.12 - Representación visual del número de vecinos

Los dos últimos algoritmos dan como resultado un sistema fiable y que asegura una tasa, de frames por segundo, suficiente para considerarse en tiempo real, pero en un ordenador. Para acelerar el clustering, de puntos en planos, se decidió implementar la tercera solución, aportada por Alexander J.B. Trevor, et al [24].

La matriz de normales resultantes del algoritmo anterior se trata como un grafo, en el que cada celda está conectada con las 4 celdas colindantes. Con esa estructura, se lanza un algoritmo de etiquetado recursivo, “Connected components labeling” [25]. Este algoritmo se discutirá con detalle en el capítulo 3. Desarrollo del prototipo.

2.1.3. Generación de audio 3D y procedural

Nuestro objetivo en la generación de sonidos, es crear sonidos que se puedan asociar a obstáculos en el mundo real, de tal forma que no interfieran en el día a día. Una de las decisiones tomadas al inicio del proyecto, es la de usar

auriculares de conducción ósea, en concreto el modelo Bluez 2S de aftershock [26].

Estos sonidos deberán sintetizarse a partir de las variables que definan al obstáculo, posición espacial, tamaño, etc. Por ello, hemos separado la búsqueda en generación audio procedural y localización espacial de audio.

2.1.3.1. Audio procedural

La literatura referente a esta área está enfocada principalmente a videojuegos, serious game y experiencias multimedia. El artículo escrito por Karen Collins [27], define lo que se considera audio dinámico o procedural y da unas breves pinceladas sobre la generación de música en videojuegos.

2.1.3.2. Sonido 3D

En la búsqueda de información nos hemos encontrado con los denominados modelos binaurales, que buscan simular las modificaciones que sufre un sonido cuando es escuchado desde un lateral del oyente.

El primer pensamiento, puede ser que la única diferencia en el audio que reciben ambos oídos, es el tiempo en el que el sonido llega a cada tímpano. Pero el sistema empleado de forma inconsciente es mucho más complejo, en el artículo de Michele Geronazzo [28] explican las modificaciones en frecuencia y amortiguación que sufre un sonido al atravesar diferentes partes del cuerpo humano y como se pueden usar los modelos binaurales para simular fuentes de audio procedentes de diferentes puntos del espacio.

Como hemos podido comprobar, estos dos campos han sido ampliamente estudiados y usados en multitud de aplicaciones, por lo que ya existen multitud de librerías que implementan estas funciones y que describiremos más adelante.

2.2. Herramientas hardware y software disponibles

En este apartado, haremos un breve repaso a los dispositivos hardware y software disponibles en el mercado.

2.2.1. Dispositivos hardware

Para desarrollar un prototipo que pueda cumplir los objetivos propuestos inicialmente, necesitamos un sistema hardware con la capacidad de capturar imágenes con el componente de profundidad.

Con las limitaciones de portabilidad, coste y potencia de procesamiento, encontramos 3 posibilidades, todas ellas están disponibles en el grupo de investigación de gráficos y geomática de la Universidad de Jaén [29]. Una vez descritas, compararemos detalladamente cada una de ellas.

2.2.1.1. Kinect 2 + Raspberry Pi 3B

La primera candidata, Ilustración 2.13, es la cámara Kinect 2 [30], desarrollada inicialmente para usarla en una videoconsola, he tenido cabida en multitud de proyectos tanto de investigación como de arte o multimedia. La Kinect no tiene capacidad de computación por sí misma, para poder usarla en el proyecto es necesaria una placa controladora, como puede ser la Raspberry Pi 3B [31].

Actualmente, Microsoft ha parado la producción de Kinect y no se esperan futuras versiones de esta. Eso no ha impedido que se siga usando en proyectos y prototipos, principalmente por su bajo coste comparado con productos comerciales similares.

Ilustración 2.13 - Kinect 2 + RPi 3B

2.2.1.2. Tablet de desarrollo Google Tango

La tablet de Google [16], Ilustración 2.14, es uno de los dispositivos del proyecto del mismo nombre, Tango. Nacido para potenciar la realidad

aumentada, mixta y virtual de dispositivos móviles, integrando hardware específico en estos, principalmente cámaras de profundidad e IMUs de gran precisión, acelerómetro y giroscopio.

Desafortunadamente, unos meses antes de escribir esta memoria, Google paró el desarrollo de Tango, para dar más recursos al proyecto ARCore [32]. La idea detrás de ARCore es similar a Tango, pero sin recurrir a hardware específico y que solo incluyen un grupo reducido de dispositivos. Aprovechando la calidad de las cámaras y la potencia de los procesadores, incluidos en móviles y tablets modernos, aplican técnicas de SLAM y aprendizaje del entorno. Actualmente, estas técnicas no alcanzan el nivel de precisión de una cámara rgbd, pero suficiente para el usuario final.

Ilustración 2.14 - Tablet Tango

2.2.1.3. Intel RealSense + Raspberry Pi 3

Las cámaras RealSense de Microsoft [33], tienen un tamaño y estética similar a webcams de alta gama, pero en su interior incluyen un hardware similar a Kinect. Para la comparativa hemos seleccionado el modelo superior, D435, al igual que ocurre con Kinect, es necesaria una placa controladora, Ilustración 2.15.

En las fotos de muestra de Microsoft y en algunos proyectos independientes, hemos podido comprobar resuelve algunos de los problemas

típicos de las cámaras rgbd basadas en infrarrojos, la detección de superficies textiles y la presencia de luz solar. Aunque al igual que Kinect y Tango, fallan al capturar datos en superficies translúcidas, reflectantes o absorbentes del espectro infrarrojo de la luz.

Ilustración 2.15 - RealSense + RPi 3B

2.2.1.4. Comparativa

Una vez conocidas las distintas alternativas, vamos a proceder a valorarlas en relación a la portabilidad, potencia de cálculo, precisión de la medida de profundidad y resolución, en la Tabla 2.1.

Características	Kinect2 + RPi 3B	Tablet Tango	RealSense D435 + RPi 3B
Portabilidad	Baja, necesidad de cables y batería	Alta, un solo cuerpo	Baja, necesidad de cables y batería
Potencia CPU	Baja, 4 x 1.2Ghz ARM 64Bits	Media, 4 x 2.3Ghz ARM 64Bits	Baja, 4 x 1.2Ghz ARM 64Bits
Precisión	Alta, 1.5cm a 1m y 5cm a 5m,	Alta, 1cm a 1m y 5cm a 5m,	Muy alta, no hay datos oficiales
Resolución	512x424px, 30fps	320x180px, 30fps	1280x720, 90fps
Angulo FOV	70˚	63˚	86˚

Tabla 2.1 - Comparativa dispositivos hardware

En portabilidad, es superior la tablet Tango, porque integra todos los componentes necesarios, batería, procesador, cámaras, etc. en un solo cuerpo. Los otros dos sistemas necesitan una batería capaz de suministrar suficiente energía, solo el consumo de la Raspberry, 3.5A, nos obliga a descartar las baterías externas de teléfonos móviles.

En potencia, gana Tango, con casi el doble de velocidad de procesador. Este es un apartado importante, ya que como hemos visto anteriormente, los algoritmos de segmentación son muy dependientes de la potencia de cálculo.

La precisión de los tres dispositivos es suficiente para el proyecto, por lo que no será un factor decisivo para descartar una u otra solución.

La resolución, es menos importante a la hora de elegir, ya que se reducirá la resolución de las imágenes, para asegurar el tiempo real del sistema.

Como conclusión, nos decantamos por la tablet Tango, principalmente por la portabilidad y potencia disponible. Es necesario recalcar, que este hardware se utilizará para la construcción del prototipo, no para un producto final de usuario.

2.2.2. Herramientas software

Durante el estudio de segmentación de imágenes, pudimos constatar, que no hay ningún algoritmo, que asegure el tiempo real, en librerías estándar, como pueden ser OpenCV [34], o Point Cloud Library, PCL [35]. Por lo que, se tomó la decisión de implementar un algoritmo de segmentación propio.

En relación a la literatura de generación de audio procedural, admitimos que, dadas las particularidades del motor de audio de cada plataforma y la falta de experiencia en el campo, crear una pequeña librería es inabarcable en el tiempo designado para esta tarea. Por lo tanto, nos vemos obligados a buscar alguna librería, o herramienta que permita agilizar el trabajo.

Tras una extensiva búsqueda, encontramos dos posibles candidatas, YSE [36] y Superpowered [8]. Ambos son librerías de audio muy completas, multiplataforma (Windows, Linux, MacOS, iOS y Android), optimizadas para dispositivos móviles. Las funcionalidades que integran y nos interesan son, efectos de eco, reverberación y cambio de tono o de velocidad de reproducción, en cada uno de los canales de audio usados, los cuales están limitados por la potencia del dispositivo.

Ilustración 2.16 – Motores de audio

La elección de Superpowered como motor de audio para el proyecto, viene dada por la dependencia del motor de juegos Unity [37], en los sistemas operativos iOS, Android y Linux. Existe la intención de liberar una versión nativa para estos sistemas.

3. Desarrollo del prototipo

En este capítulo describiremos todo el proceso de desarrollo, para permitir un mayor control en la depuración del algoritmo de segmentación, se desarrollará una primera versión de escritorio en C++. Una vez desarrollado la versión final del algoritmo, portaremos esta versión a Android.

El siguiente paso es la generación de sonidos 3D, primero crearemos sonidos estáticos, esto nos permitirá comprobar las posibilidades de Superpowered.

En la fase de integración buscaremos que parámetros del sonido, como el tono o la velocidad, se asocian mejor a las propiedades de los obstáculos, tamaño, posición, etc.

Por último, realizaremos pruebas que permitirán ajustar la sintetización del sonido y validarán la madurez del desarrollo.

3.1. Segmentación en ordenador

Como hemos comentado anteriormente, primero crearemos una versión de escritorio en C++ del algoritmo, “Efficient Organized Point Cloud Segmentation with Connected Components” [23]. Para obtener datos de entrada de entornos reales, hemos usado el dataset de imágenes rgbd de freiburg1 [8], en concreto desk, floor y xyz, un total de 5056 imágenes rgb tomadas con la Kinect en una oficina.

Para el desarrollo, hemos seleccionado el IDE Visual Studio [6], al ser el más completo para Windows y por ser gratuito para la comunidad universitaria. Para la lectura de imágenes del dataset, en formato png, dependemos de la librería LodePNG [38], muy compacta y que permite su uso y modificación en todo tipo de proyectos.

En el proyecto se han incluido tres clases propias, Punto3D, Elemento3D y Plano3D:

- Punto3D, en esta clase se guarda la posición 3D del punto, su normal y otros datos espaciales del mismo. Por optimización, en cada punto se almacena un puntero al Elemento3D que lo contiene.
- Elemento3D, es una clase en la que se a guardarán todos los puntos pertenecientes a cada elemento encontrado en la segmentación, además de su identificador y posición espacial.
- Plano3D, hereda de Elemento3D y amplía su funcionalidad con datos y operaciones de plano, solo se han incluido las estrictamente necesarias, como la distancia punto-plano.

Para facilitar la explicación del algoritmo, hemos dividido en subsecciones la segmentación de la imagen, en concreto:

1. Lectura de imagen rgbd
2. Generación de imágenes integrales
3. Calcular el mapa dinámico de vecinos
4. Calcular las normales en cada punto
5. Agrupar con componentes conexas (normales)
6. Eliminar planos no válidos
7. Extender planos
8. Combinar planos
9. Agrupar con componentes conexas (profundidad)
10. Detectar suelo
11. Detectar elementos relevantes

En algunos apartados, incluiremos una imagen del resultado.

3.1.1. Lectura de la Imagen rgbd

En la implementación de escritorio, las imágenes se leen de un archivo png en escala de grises, con una resolución de 32 bits por pixel, en el que cada nivel de gris representa una distancia.

En esta fase se genera el vector de Punto3D, que se usará en todo el proceso, cada pixel se transforma en un Punto3D. En cada Punto3D, se guarda si es válido, si el sensor ha podido capturarlo, o no, y en caso afirmativo, su

posición en el espacio. Hay que destacar, que esta posición es relativa a la cámara, en el momento de captura de la imagen.

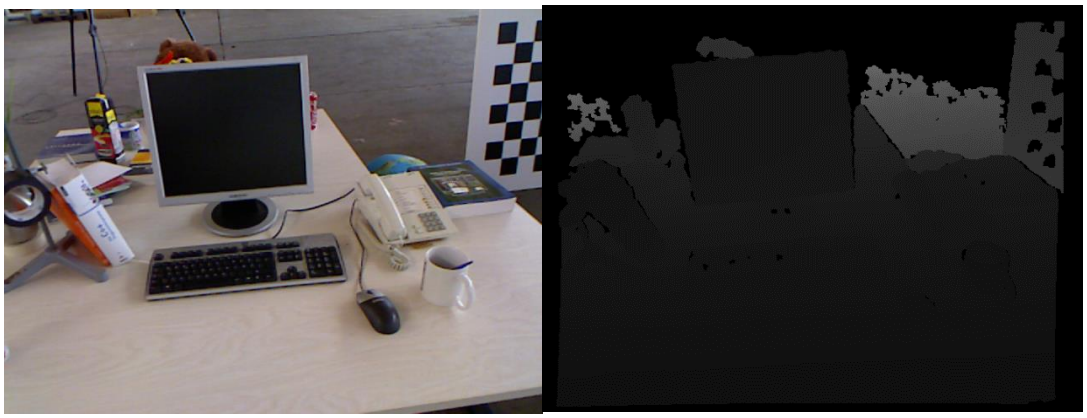


Ilustración 3.1- Imagen rgb y depth

3.1.2. Generación de imágenes integrales

Como se comentó anteriormente, es necesario calcular las imágenes integrales de cada componente del espacio, x, y, z. Para ello, inicializamos tres vectores con el mismo tamaño que la imagen y procesamos los puntos.

Para una imagen integral, donde V es la matriz original e I la imagen integral, hay que seguir los siguientes pasos:

1. Rellenar el valor superior izquierda con el valor original
2. Rellenar la primera fila, `I(x,y) = V(x,y) + I(x-1,y)`
3. Rellenar la primera columna, `I(x,y) = V(x,y) + I(x,y-1)`
4. Rellenar el resto de celdas,
   `I(x,y) = V(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)`

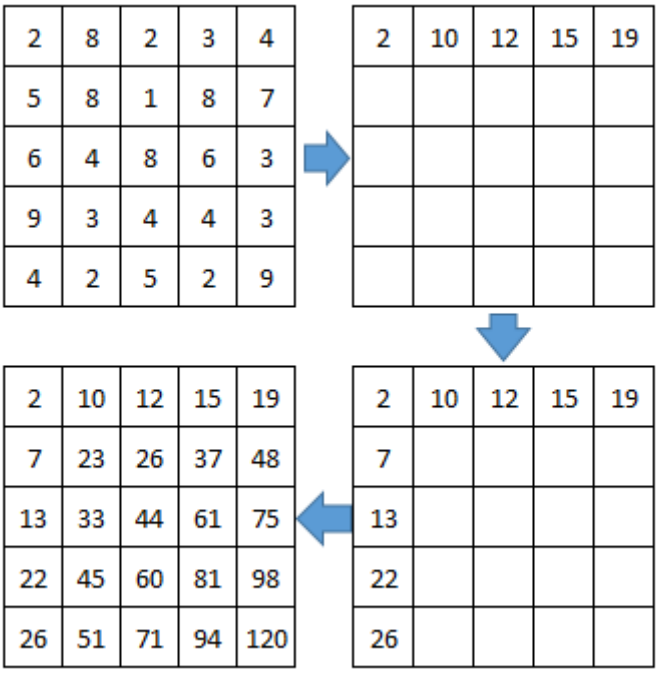


Ilustración 3.2 - Imagen integral

3.1.3. Calcular el mapa dinámico de vecinos

La variación propuesta por S.Holzer et al [22], para mejorar en zonas fronteras y alejadas, necesita previamente un array donde cada pixel tiene un número de vecinos para calcular su normal.

Para calcular este mapa de vecinos, necesitamos saber la distancia de cada punto a una zona frontera, una frontera es aquella donde un pixel tiene una distancia superior a un umbral, con cualquiera de los vecinos. Este umbral, Ilustración 3.3Ilustración 3.3 - Umbral frontera, depende de tres constantes que no vienen especificados en el artículo, porque depende de la precisión necesaria.

$$Umbral(m,n) = \gamma 1 + \gamma * \alpha * d(m,n)^2$$

Ilustración 3.3 - Umbral frontera

La variable d, es la distancia al sensor, los valores para las constantes son, delta1 0.01, delta 1.25 y alfa 0.0028. En la Tabla 3.1, se pueden consultar los valores resultantes a varias distancias.

Distancia (metros)	Umbral
0,5	0,010875
1	0,0135
1,5	0,017875
2	0,024
2,5	0,031875
3	0,0415

Tabla 3.1 - Umbral frontera

Una vez calculados los puntos frontera, se lanza un algoritmo de crecimiento, donde las semillas son los mismos puntos, y su valor es 0. En cada paso del algoritmo, se suma uno al valor de distancia y se le asigna, a los puntos contiguos a los procesados en el paso anterior, Ilustración 3.4.

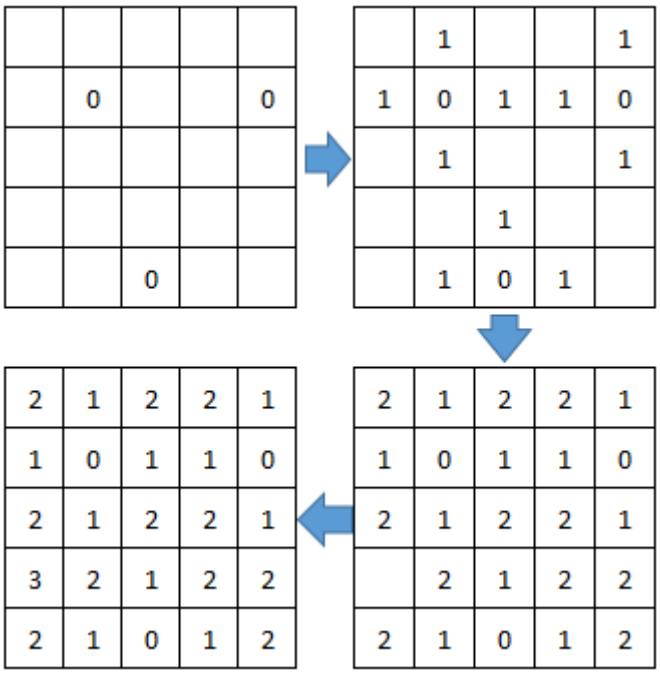


Ilustración 3.4 - Algoritmo de crecimiento

También es necesario calcular el valor B(m,n),Ilustración 3.5. Que una vez más depende de la distancia al cuadrado y dos constantes que tenemos que obtener empíricamente. En este caso el valor de beta1 es 3 y el de beta es 30, dando como resultado valores entre 3 y 5 vecinos, dependiendo de la distancia.

$$B(m,n) = \beta 1 + \beta * \alpha * d(m,n)^2$$

Ilustración 3.5 - Formula B(m,n)

Una vez obtenidos los valores de distancia a fronteras, T(m,n) y el valor B(m,n), podemos calcular el número de vecinos, R(m,n), con la fórmula de la Ilustración 3.6.

$$R(m,n) = \min(B(m,n), \frac{T(m,n)}{\sqrt{2}})$$

Ilustración 3.6 - Calculo del número de vecinos

Ilustración 3.7 - Imagen del número de vecinos

3.1.4. Calcular las normales en cada punto

Para calcular la normal en cada punto de la imagen, se necesitan los vectores perpendiculares al punto. Estos vectores se calculan con los puntos superior e inferior, y derecho e izquierdo, o más bien, con un valor media usando los vecinos calculados con anterioridad.

Para calcular el valor medio de cada componente, en su imagen integral, usamos la fórmula de la Ilustración 3.8. El vector perpendicular horizontal, se obtiene con en el punto superior e inferior y el vector perpendicular se obtiene con el punto derecho e izquierdo, Ilustración 3.9.

$$S(I,m,n,r) = \frac{1}{4 * r^2} * (I(m+r,n+r) + I(m-r,n-r) - I(m+r,n-r) - I(m-r,n+r))$$

Ilustración 3.8 - Calculo valor medio en la integral

$$VPV(m,n) = \Big(S(I_x, m+1, n, R(m,n)), S\big(I_y, m+1, n, R(m,n)\big), S(I_z, m+1, n, R(m,n))\Big) - \Big(S(I_x, m-1, n, R(m,n)), S\big(I_y, m-1, n, R(m,n)\big), S(I_z, m-1, n, R(m,n))\Big)$$

$$VPH(m,n) = \Big(S(I_x, m, n+1, R(m,n)), S\big(I_y, m, n+1, R(m,n)\big), S(I_z, m, n+1, R(m,n))\Big) - \Big(S(I_x, m, n-1, R(m,n)), S\big(I_y, m, n-1, R(m,n)\big), S(I_z, m, n-1, R(m,n))\Big)$$

Ilustración 3.9 - Vector perpendicular vertical y horizontal

Finalmente, el vector normal se calcula como como el producto vectorial de los vectores perpendiculares. En la Ilustración 3.10, se pueden ver diferentes representaciones de la normales, en la primera imagen, cada componente de la normal, se ha traducido en una componente del color, rojo para la x, verde para la y, y azul para la z. El resto de imágenes muestran cada componente, x, y, z, en escala de grises.

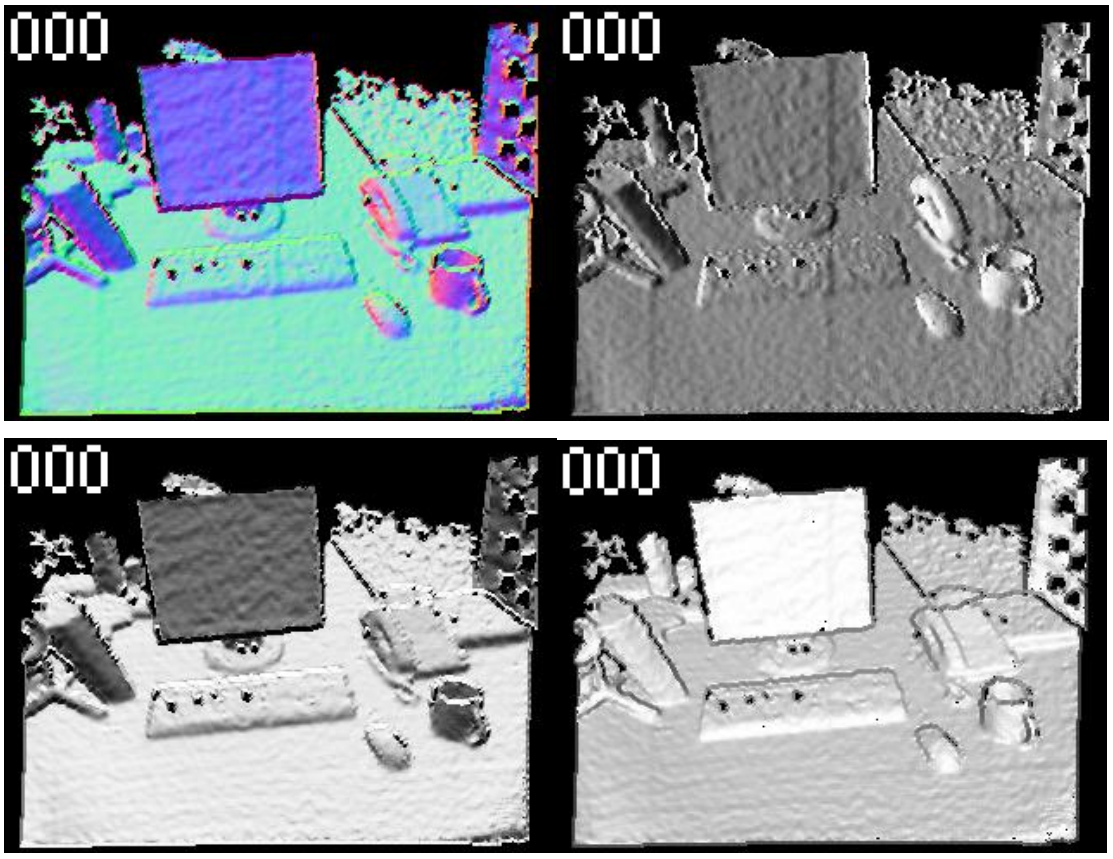


Ilustración 3.10 - Representación de las normales

3.1.5. Agrupar con componentes conexas (normales)

Para procesar el mapa de normales, usaremos el algoritmo de unión-find descrito en [24], para lo cual se tratará la imagen como un grafo, donde cada

pixel está conectado a sus cuatro pixeles colindantes. El resultado será un conjunto de planos.

Para cada pixel, se comprueba si el superior y el izquierdo supera un determinado test, deben de estar a menos de 20cm y el ángulo formado por sus normales no puede ser mayor a 4˚. Pueden ocurrir las siguientes situaciones:

- Se supera el test en alguno de ellos, se consideran que pertenecen al mismo conjunto y se le asigna la misma etiqueta.
- Si ninguno de los puntos supera el test, se genera una nueva etiqueta y se aplica al punto.
- Si los dos superan el test, se le asigna alguna de las dos etiquetas y se guarda en un vector, que una etiqueta es hija de la otra.

El vector donde se guardan las relaciones esta estructura de tal manera, que la posición de una etiqueta corresponde al padre de esta, por ejemplo, v[50]=12, significa que la etiqueta 12 es padre de la etiqueta 50. En esta estructura se pueden observar diferentes árboles, todas las etiquetas en un mismo árbol, son equivalentes, Ilustración 3.11.

1	2	3	4	5	6	7	8
2	3	0	3	7	7	0	3

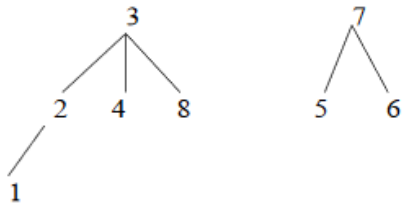


Ilustración 3.11 - Estructura de árbol

Para terminar, se asigna a cada punto un plano con el identificador del nodo raíz del árbol donde se encontraba su etiqueta y se inicializan los Planos3D.

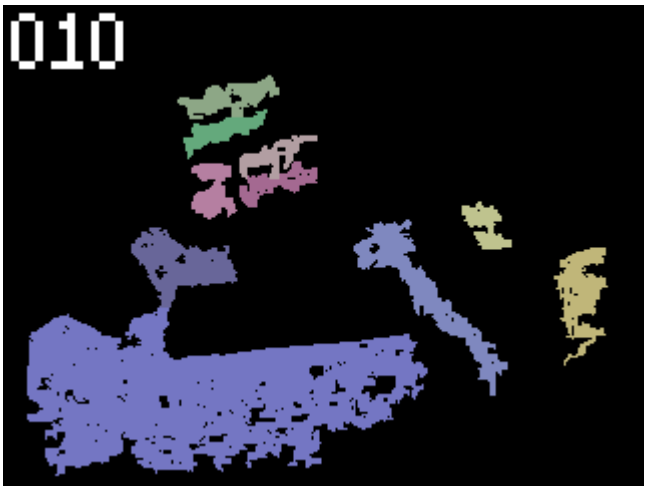


Ilustración 3.12 - Selección de planos

3.1.6. Eliminar planos no válidos

En el paso anterior se sometieron los puntos a un test de convexidad punto a punto, de forma local. Esto puede provocar errores, una superficie puede ser convexa, pero a la vez plana de forma local, por ejemplo, un cuenco bajo.

Para evitar estos casos, se comprueban cuantos puntos tienen una normal diferente a la de la superficie en más de 10˚ al plano, si suponen más del 20% de los puntos pertenecientes a la superficie, esta se descarta.

También se descartan aquellas superficies que no lleguen a un mínimo de puntos, fijado en 250.

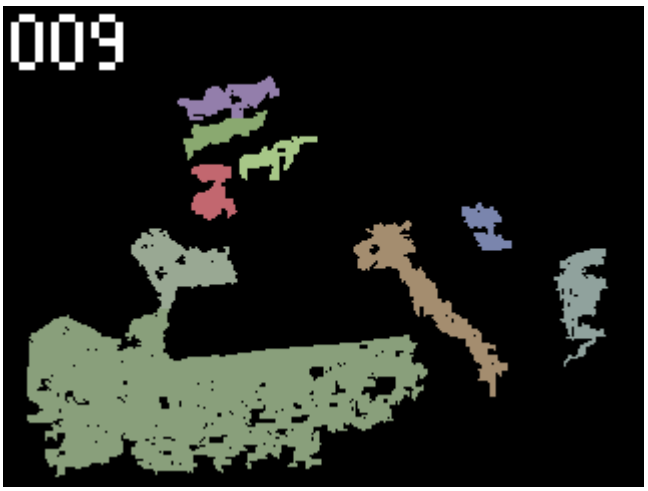


Ilustración 3.13 - Planos válidos

3.1.7. Extender planos

Al igual que se pueden detectar superficies convexas, que localmente son planos, puede ocurrir la situación inversa. Puntos de un plano que se han descartado, porque localmente no cumplían el test componentes conexas.

Para incluir esos puntos, se expanden los planos hasta encontrar todos los puntos que pasen el test, de distancia y normal, con el plano.

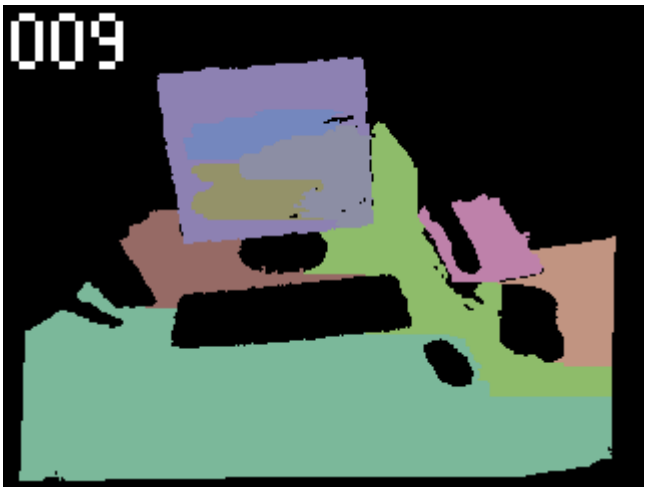


Ilustración 3.14 - Planos extendidos

3.1.8. Combinar planos

Al haber extendido los planos, podemos encontrar planos que anteriormente no eran colindantes, pero ahora sí. Estos planos se comparan entre sí, si la diferencia en sus normales, es menor a 10˚, se consideran que son partes de la misma superficie y se combinan en uno.

Este paso es el último en la detección de planos en la imagen.

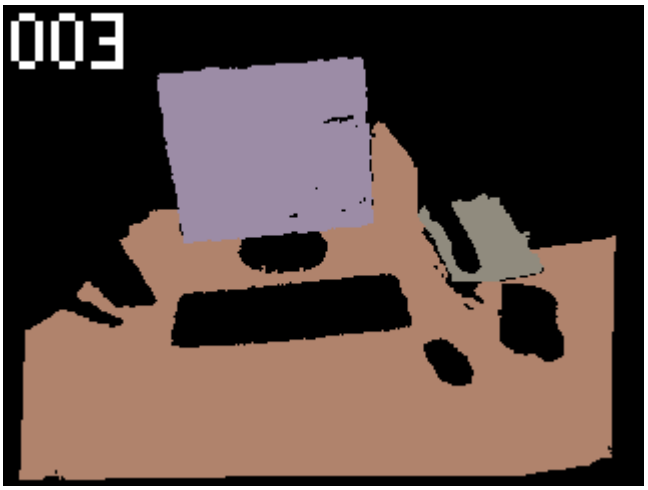


Ilustración 3.15 - Planos extendidos

3.1.9. Agrupar con componentes conexas (profundidad)

Una vez obtenidos los todos los planos, lanzaremos una vez más el algoritmo de componentes conexas, esta vez el test para agrupar puntos será de distancia, Ilustración 3.3. Los puntos que ya tienen asignado un plano, se ignorarán.

Al igual que en el caso de los planos, se ignorarán aquellos que no tengan al menos 300 puntos.

El resultado será la segmentación completa de la imagen, para diferenciar planos, de elementos no planos, las etiquetas para los planos son números enteros positivos y las etiquetas de los elementos no planos, enteros negativos.

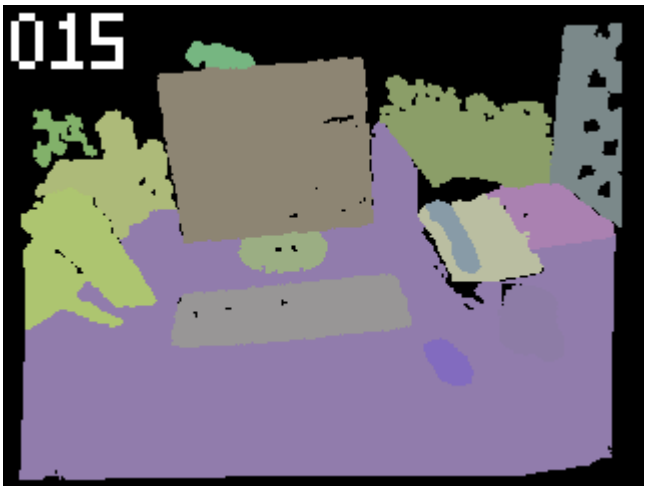


Ilustración 3.16 - Segmentación de la imagen

3.1.10. Detectar suelo

Para evitar falsos positivos con el suelo, debemos ignorarlo en la segmentación, en la versión de escritorio no tenemos los datos de posición y orientación del sensor. Por lo que, hemos marcado como suelo, el plano más bajo encontrado.

En la versión de Android, sí que se podrá detectar de manera precisa el suelo.

3.1.11. Detectar elementos relevantes

Una vez localizados los elementos en la imagen, no distinguimos entre elemento plano o no plano, hay que distinguir que elementos se consideran relevantes.

Para considerar un elemento relevante, se han tenido en cuenta 3 factores, que se puntúan de 0 a 1, se multiplican por un factor y se suman. Los factores son:

- Tamaño del elemento en la imagen, factor 10.
- Cercanía al centro, foco, de la imagen, factor 6.
- Distancia a la cámara, factor 6.

El valor de estos factores se usa para compensar los valores, es posible detectar elementos en los bordes, o alejados, pero es difícil que un elemento ocupe más de la mitad de la imagen. Se han probado varios multiplicadores, siendo los anteriores, los que, empíricamente, mejores resultados han dado.

En la Ilustración 3.17, se pueden ver los elementos relevantes detectados, siendo los más puntuados, por orden, el rojo, verde y azul.

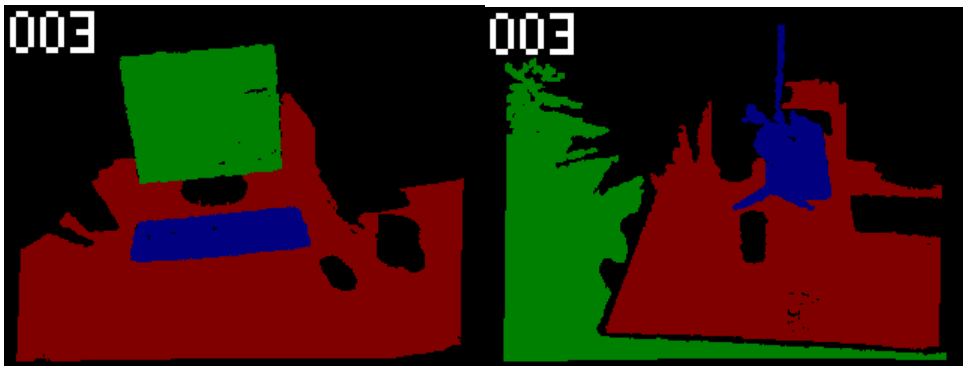


Ilustración 3.17 - Elementos relevantes

3.2. Segmentación en Android

Una vez finalizada y probada la versión de escritorio del algoritmo, procederemos a integrarlo en una aplicación Android. Para lo cual hemos usado el IDE Android Studio [11], es el entorno oficial y mantenido por Google, principal responsable de Android.

Hay dos opciones de lenguaje principales para programar aplicaciones en dispositivos Android, Java o C++ nativo. En cualquier caso, la interfaz se programa en Java, pero existe la posibilidad de usar C++, con el conjunto de herramientas NDK [39].

En nuestro caso, la API de Tango [16], nos obliga a elegir C++, ya que la versión para Java no ofrece algunas funcionalidades necesarias.

Para acelerar el desarrollo, hemos empezado usando una aplicación de ejemplo, incluida en el SDK de Tango. En ella ya están incluidas las rutinas de inicialización del motor de Tango, configuración de los callback del sensor de profundidad y algunas interacciones entre la capa Java y C++.

Una vez preparado el proyecto exportamos el código generado anteriormente, las clases y la mayoría de los métodos pueden usarse directamente, solo se realizaron los siguientes cambios:

- El código de lectura de la imagen y de mostrar en pantalla, es ligeramente diferente.

- La imagen obtenida por el sensor, de 320*180px, se ha redimensionado a 160*90px, para mantener una tasa estable de fps.
- Al reducir la imagen a un cuarto de su tamaño original, se han modificado los umbrales para considerar validos planos y elementos. El nuevo número mínimo de puntos es 100 y 150 respectivamente.
- Una mínima parte del código relativo al manejo de estructuras de datos, usaba funcionalidad de los estándares C++14 y C++17 [40], que han tenido que ser reescritas.
- Para detectar el suelo necesitamos transformar las coordenadas de la cámara, a coordenadas del mundo real con los datos del giroscopio de la tablet. Aquellos planos horizontales que tengan una diferencia de altura superior a un metro, se considerarán suelo.

Además de lo anterior, hemos diseñado una interfaz para depurar los resultados del prototipo. Esta permite seleccionar el modo de visión, el cual se superpone a la imagen en color de la cámara trasera, pudiendo elegir el nivel de transparencia de cada capa, a continuación, se pueden observar todos los modos de visualización posibles.

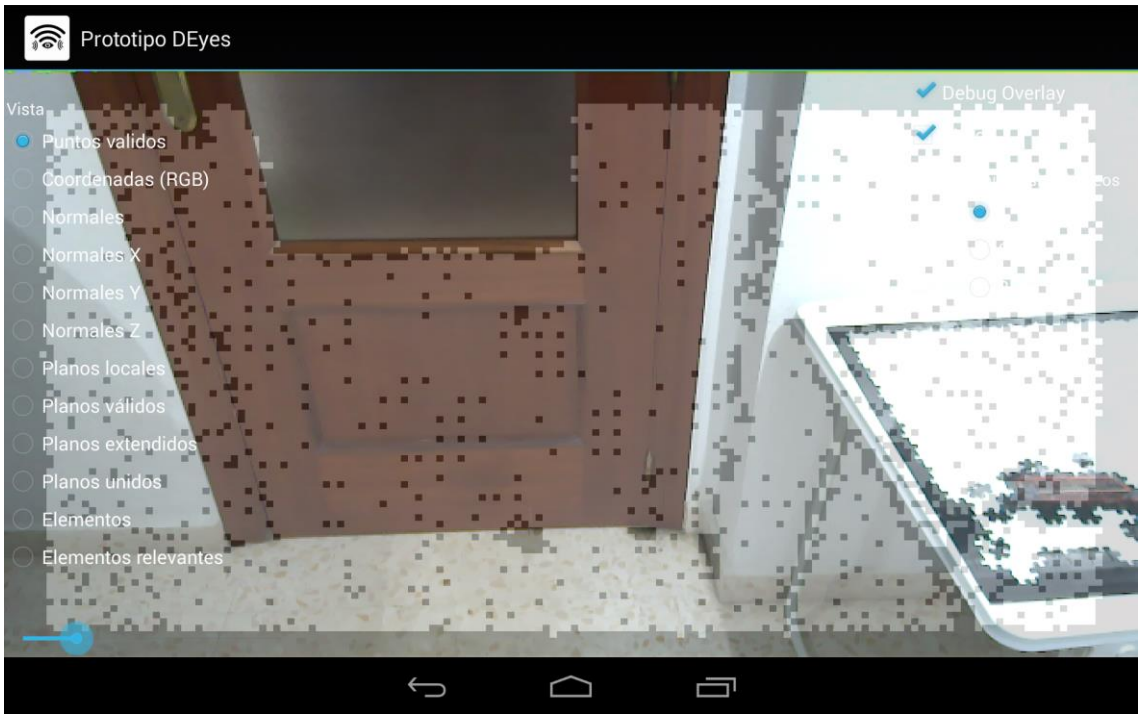


Ilustración 3.18 - Puntos válidos

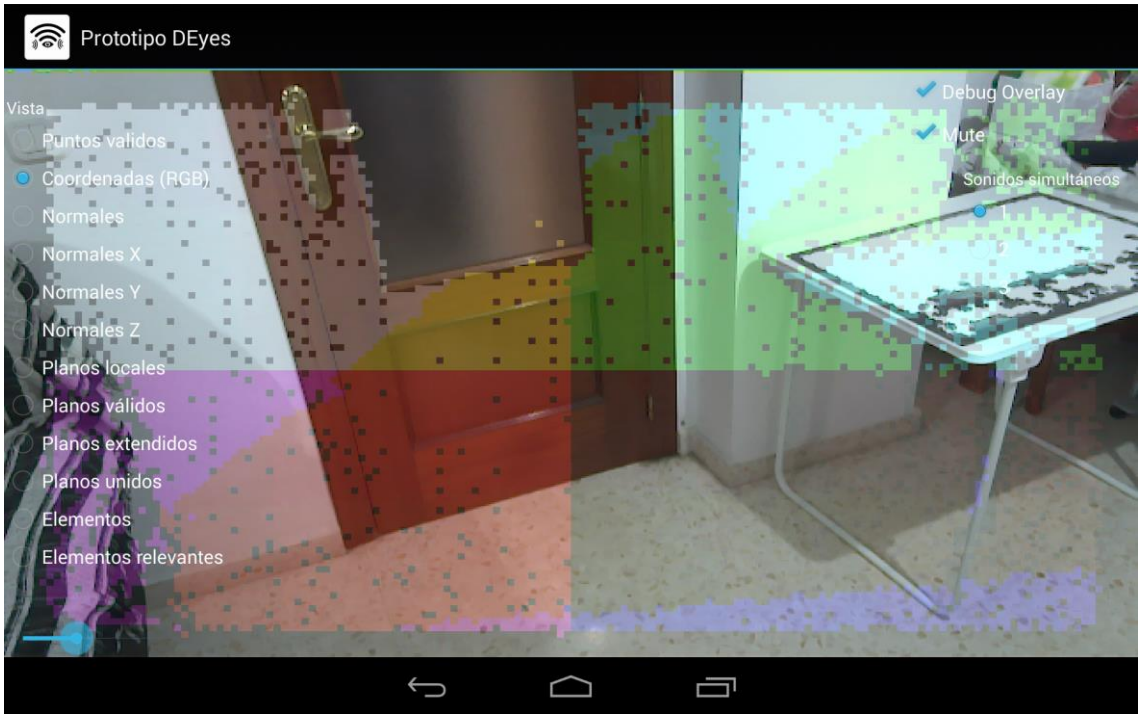


Ilustración 3.19 - Coordenadas rgd

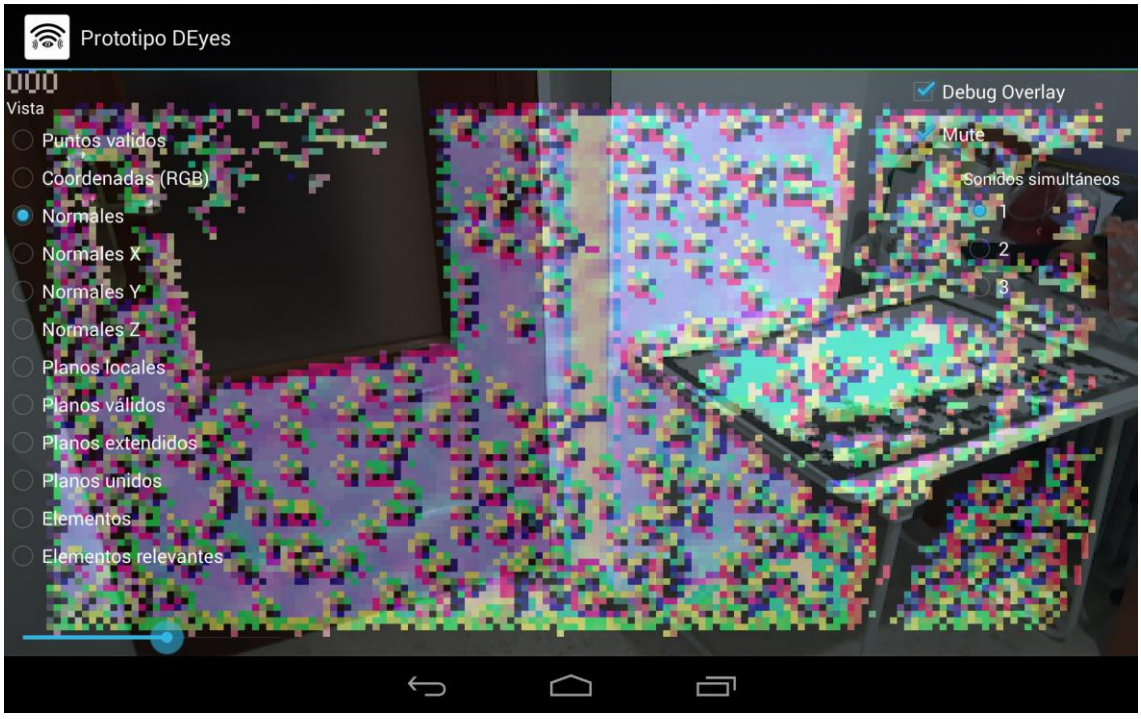


Ilustración 3.20 - Normales rgb

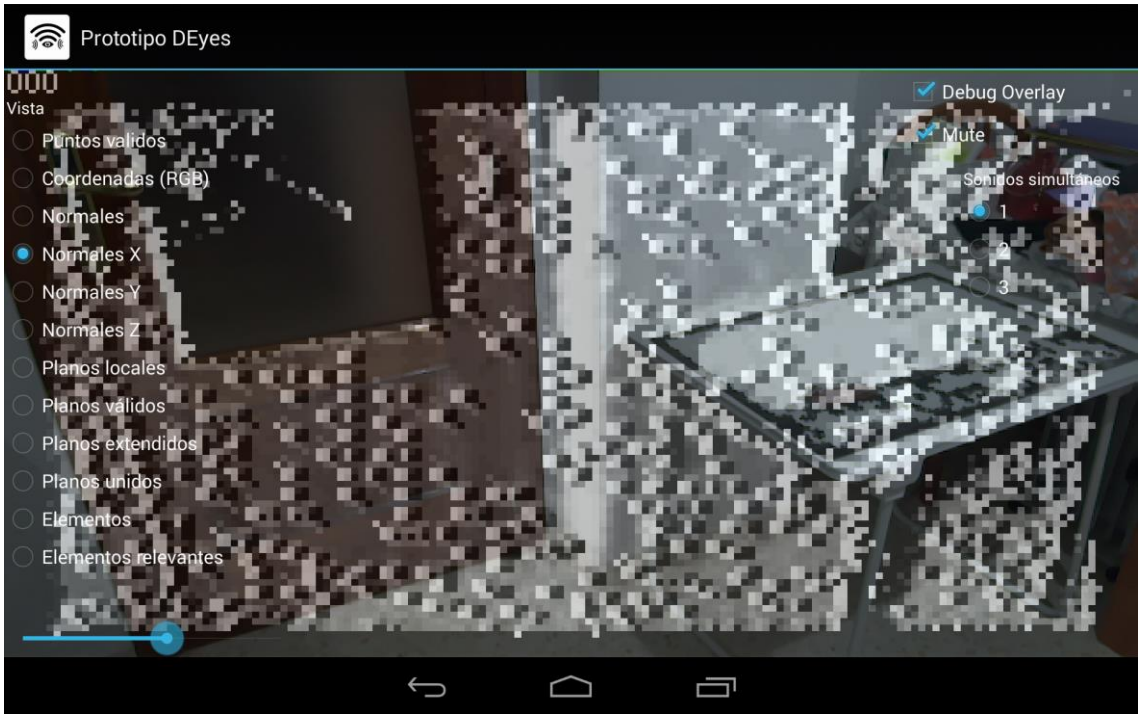


Ilustración 3.21 - Coordenadas X en escala de grises

Ilustración 3.22 - Planos locales

Ilustración 3.23 - Planos locales válidos

Ilustración 3.24 - Planos extendidos

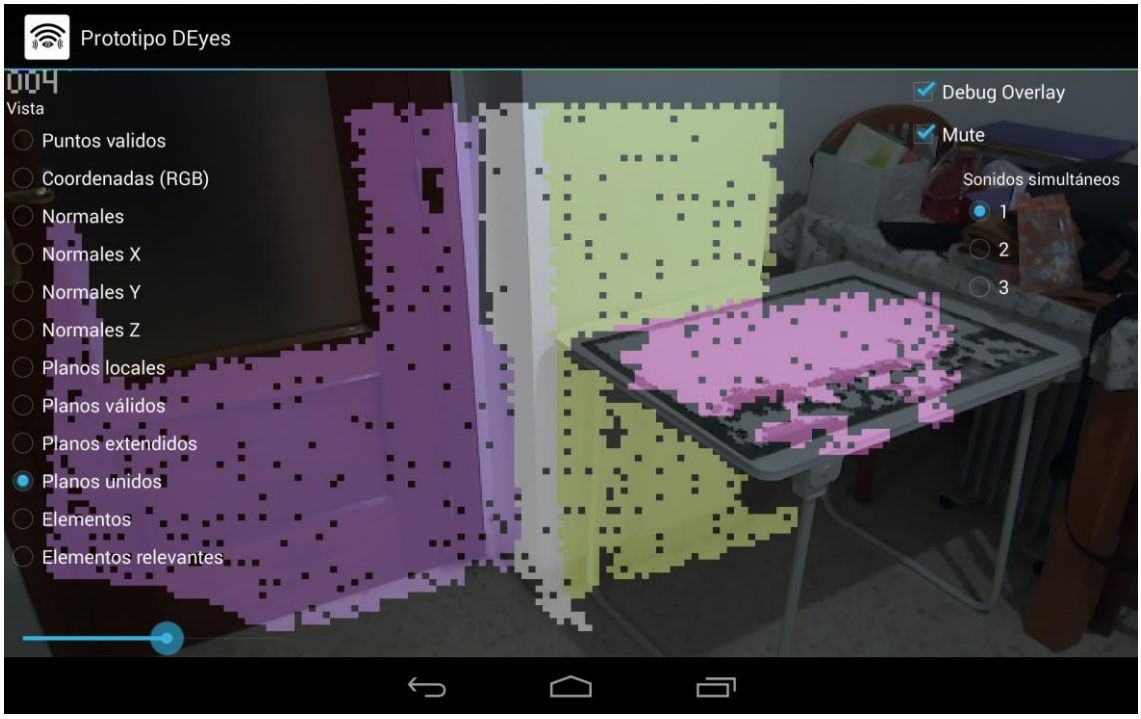


Ilustración 3.25 - Planos unidos

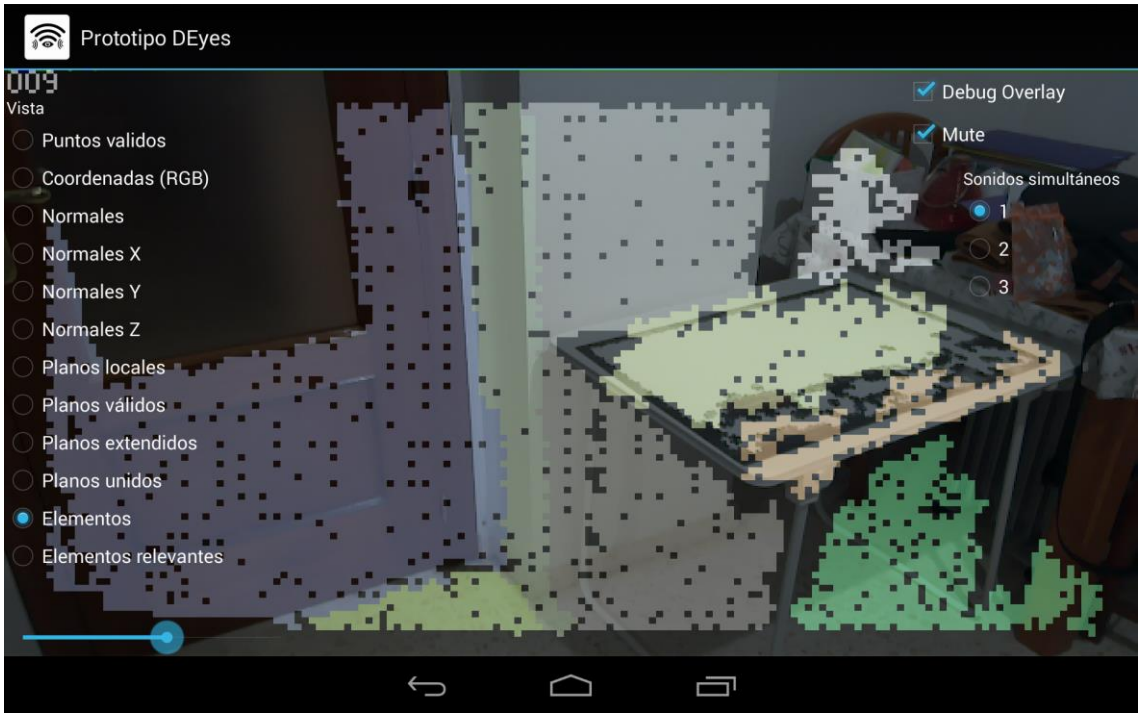


Ilustración 3.26 - Elementos

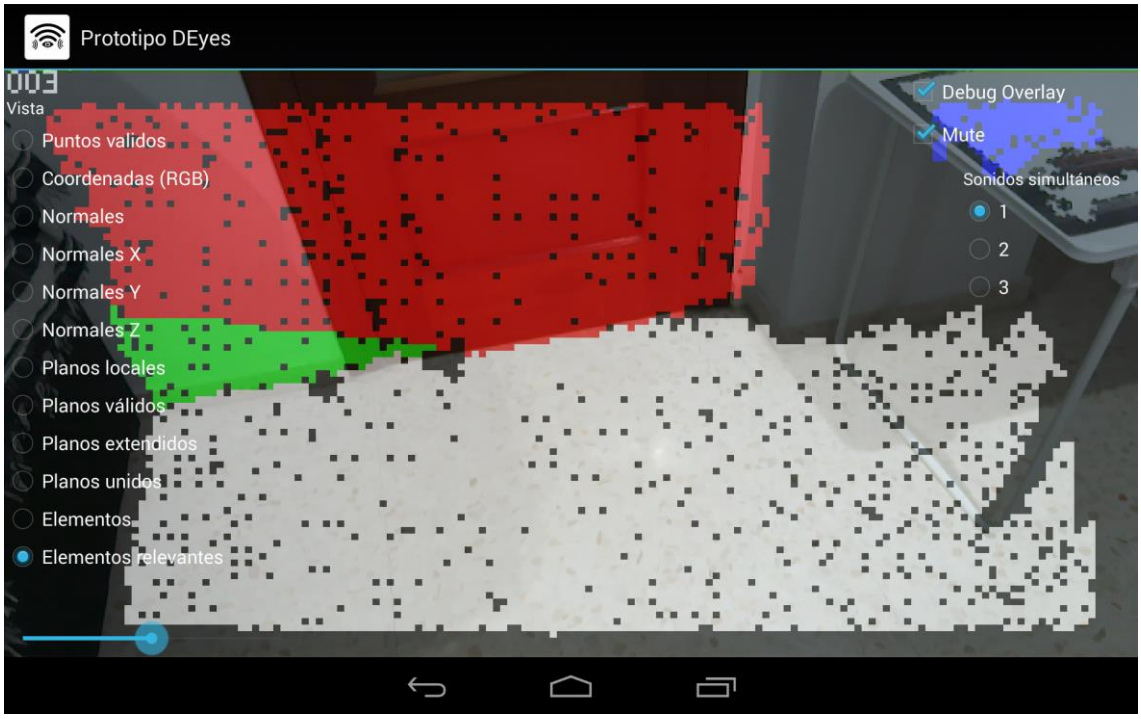


Ilustración 3.27 - Elementos relevantes y suelo

3.3. Sonido procedural 3D

Durante la fase de generación de sonido, hemos probado con 4 tipos de sonidos iniciales diferentes, pitido corto, doble pitido corto, pitido continuo, ruido blanco. Y con 6 modificaciones del sonido, velocidad de reproducción, tono, eco, reverberación, localización horizontal, y localización vertical.

Después de realizar múltiples combinaciones de sonidos y efectos hemos llegado a las siguientes conclusiones:

1. El pitido corto no se puede combinar con los efectos, reverberación y velocidad de reproducción. Estos últimos no son apreciables.
2. El doble pitido corto se puede combinar con todas las modificaciones, excepto con la reverberación, el efecto que produce no es reconocible.

3. El pitido continuo no se puede usar con el eco, velocidad de reproducción y reverberación. No se puede usar más de un sonido simultáneo, y provoca molestias al poco rato.
4. El ruido blanco no se le puede aplicar ningún efecto, y es muy difícil de localizar espacialmente.
5. Es fácil localizar un sonido verticalmente, pero no horizontalmente.
6. El efecto de velocidad de reproducción es el más adecuado para reconocer la distancia a un obstáculo, en el día a día se pueden encontrar varios ejemplos, el asistente de aparcamiento de un coche, o el sonido acelerado de los semáforos para peatones.
7. Para trasladar el tamaño de un elemento a un sonido, hemos probado el efecto de eco y el de modificación de tono. El más reconocible, es el cambio de tono, más grave para elementos grandes, y más agudo para los pequeños.

Con las conclusiones anteriores, hemos creado las siguientes asociaciones:

- La velocidad de reproducción cambiará en función de la distancia, a medio metro será tres veces la velocidad original y a 4 metros o más, se reducirá a la mitad. Esta relación es lineal.
- El tamaño del elemento se manifestará como un cambio de tono, los objetos grandes tendrán un tono grave, y los pequeños un tono más agudo. Esta relación es lineal, y se ha ajustado para que el tono más agudo sea lo menos molesto posible.
- La localización espacial del sonido, la implementa Superpowered, solo es necesario indicarle el ángulo, horizontal o vertical, de la fuente de audio.
- Aunque en las pruebas no se ha conseguido distinguir si un sonido proviene desde arriba o abajo, se ha decido mantener esta modificación. Es posible que con suficiente entrenamiento se pueda llegar a diferenciar.

3.4. Integración

Una vez terminadas la fase de segmentación y la de definición del sonido, tenemos que integrarlas en una sola aplicación.

La librería Superpowered está diseñada para permitir una rápida integración, una vez inicializado el entorno motor de audio, este lanzará una función de forma cíclica y cada pocos milisegundos.

Es en esta función, se activarán las diferentes fuentes de sonido, y se realizarán las modificaciones al sonido. En la Ilustración 3.28, se puede ver el fragmento de código para la sintetización del sonido del primer obstáculo. Una vez generadas todas las fuentes de audio, se mezclan en un buffer y se devuelven, Ilustración 3.29

```
player1->setTempo(tempo1, true);
player1->setPitchShift(pitch1);
spatializer1->azimuth = azimuth1;
spatializer1->elevation = elevation1;

if (!player1->process(floatBuffer1, false, (unsigned int) numberOfFrames))
    return false;
if (!spatializer1->process(floatBuffer1, NULL, floatBuffer1, NULL,
                           (unsigned int) numberOfFrames, true))
    return false;

inputs[0] = floatBuffer1;
```

Ilustración 3.28 - Código sonido procedural 3D

```
mixer->process(inputs, outputs, inputLevels, outputLevels, NULL, NULL,
               (unsigned int) numberOfFrames);

SuperpoweredFloatToShortInt(floatBufferMixer, audio, (unsigned int) numberOfFrames);
```

Ilustración 3.29 - Mezcla de audios

Una vez combinadas las dos funcionalidades, se ha generado una apk para Android, se ha decidido ponerle el nombre provisional, DEyes, un juego de palabras entre depth y eyes.

3.5. Pruebas

Las pruebas se han realizado con 10 compañeros de trabajo, a los que se les ha planteado dos situaciones.

En el primer test se han generado dos sonidos estáticos, con diferentes propiedades:

- Primer sonido
  - Distancia, 3 metros.
  - Tamaño grande.
  - Posición, 20 grados a la izquierda y 20 hacia arriba.
- Segundo sonido
  - Distancia, 0.5 metros.
  - Tamaño pequeño.
  - Posición, 40 grados a la derecha y 30 hacia abajo.

Para la realización de esta prueba, se les ha colocado los auriculares de conducción ósea, con el sonido activo durante unos minutos. Y sin explicar las características del sonido, se le han preguntado las siguientes cuestiones:

- ¿Cuántos sonidos eres capaz de reconocer?
  - Todos los participantes reconocieron dos sonidos en la escena.
- ¿Podrías indicar, aproximadamente, la dirección horizontal de procedencia de los sonidos?
  - En esta pregunta, hubo tres resultados diferentes
    - 2 personas necesitaron unos minutos para distinguir si el sonido procedía de la derecha o de la izquierda.
    - 5, localizaron a derecha o izquierda los sonidos, casi instantáneamente.
    - 3, localizaron, aproximadamente, la localización exacta del sonido.
- ¿Podrías indicar, aproximadamente, la dirección vertical de procedencia de los sonidos?

    - Ningún usuario pudo distinguir si alguno de los sonidos estaba arriba o abajo.
- ¿Qué sonido parece más cercano?
    - Todos acertaron, el más agudo y rápido era el más cercano.
- ¿Qué sonido puede interpretarse como un objeto grande?
    - Todos acertaron, el lejano y grave.
- ¿Interfieren los auriculares para escuchar el entorno?
    - Todos indicaron, que el sonido ambiental y el proveniente de los auriculares, se escuchaban perfectamente.

En la segunda parte de la prueba, se le explica al usuario como se generan los sonidos y como se usa la aplicación. Posteriormente, se le deja probarla durante unos minutos para que la pruebe y comente su experiencia.

Las opiniones fueron muy positivas, y se recogieron los siguientes comentarios:

- Varios usuarios indicaron que, si un elemento se encuentra poco desplazado hacia un lateral, no se distingue lo suficiente.
- Si los movimientos no son rápidos, la segmentación es lo suficientemente rápida como para parecer natural.
- Los diferentes modos de visualización han sido destacados, porque permiten ver los diferentes pasos hasta llegar a distinguir los elementos destacados.

Una vez terminadas las pruebas se realizó una corrección menor, el ángulo horizontal de la fuente de sonido se multiplicó por dos. Una vez corregido, todos los usuarios localizaron más rápidamente la localización de los sonidos.

4. Conclusiones

Este ha sido un proyecto muy ambicioso y que he disfrutado realizando. El desafío de implementar algoritmos potentes y actuales, y conseguir hacerlos funcionar en un dispositivo móvil, ha sido muy motivador.

Otra de las razones que me llevaron a elegir este TFM, es la componente social del mismo. Me ha alegrado ver que, aunque no todos se hagan públicos, hay proyectos y productos comerciales con mucho potencial para ayudar al colectivo de personas con discapacidad visual severa.

4.1. Trabajo futuro

El prototipo desarrollado ha tenido un tiempo límite de 300 horas, muchos detalles y mejoras han tenido que ser descartadas. A continuación, detallaremos algunas de ellas:

- Crear un sistema más discreto, Microsoft tiene una versión de RealSense donde el sensor y la circuitería está separados. Esto permitiría integrar fácilmente el sensor en unas gafas y ocultar la placa.
- Incluir la detección de objetos específicos, se podría entrenar un algoritmo de aprendizaje profundo con redes neuronales [41], para funcionar en conjunción con la segmentación. Sonidos concretos se asignarían a objetos como escaleras, pasos de cebra, semáforos, etc.
- Realizar pruebas extensivas con usuarios con ceguera o discapacidad visual severa.
- Contactar con fundaciones y organizaciones que puedan estar interesadas en colaborar y difundir el proyecto.

Bibliografía

[1] O. m. d. l. salud, «OMS, blindness and visual impairment,» [En línea]. Available: http://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Último acceso: Junio 2018].

[2] P. B. L. Meijer, «An experimental system for auditory image representations.,» *IEEE Trans. Biomed. Eng. [Online].,* vol. 39, nº 2, pp. 112-121, 1992.

[3] D. T. F. V. S. Cardin, «A wearable system for mobility improvement of visually impaired people,» *Vis. Comput.,* vol. 23, nº 2, pp. 109-118, 2007.

[4] K. W. S. Meers, «A substitute vision system for providing 3D perception and GPS navigation via electro-tactile stimulation,» *1st Int. Conf. Sens. Technol.,* pp. 21-23, 2005.

[5] Eyesynth, «Eyesynth,» [En línea]. Available: eyesynth.com. [Último acceso: Junio 2018].

[6] COPE, «Le prohíben montar en taxi en compañía de su perro guía,» 2017.

[7] S. P. y. C. M. Robins, Administración. Octava edición, Pearson Educación, 2005.

[8] «SuperpoweredSDK,» [En línea]. Available: http://superpowered.com/. [Último acceso: Mayo 2018].

[9] Microsoft, «Windows,» Junio 2018. [En línea]. Available: www.microsoft.com/windows.

[10] Microsoft, «Visual Studio,» Junio 2018. [En línea]. Available: https://visualstudio.microsoft.com.

[11] Google, «Android Studio,» Junio 2018. [En línea]. Available: https://developer.android.com/studio/.

[12] C. V. G. T. U. o. Munich, «Dataset RGBD Freiburg,» Mayo 2018. [En línea]. Available: https://vision.in.tum.de/data/datasets/rgbd-dataset/download#freiburg1_xyz.

[13] Apache, «OpenOffice,» Junio 2018. [En línea]. Available: https://www.openoffice.org.

[14] GanttProject, «GanttProject,» Junio 2018. [En línea]. Available: https://www.ganttproject.biz/.

[15] «Audacity,» [En línea]. Available: https://www.audacityteam.org/. [Último acceso: Junio 2018].

[16] Google, «Google Tango,» [En línea]. Available: https://get.google.com/tango/. [Último acceso: Febrero 2018].

[17] N. G. B. Dimitrios Dakopoulos, «Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey,» *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 40, nº 1, pp. 23-35, 2009.

[18] W. R. W. R. L. W. B. B. Blasch, «Foundations of Orientation and Mobility,» *New York:AFB Press,* 1997.

[19] D. P. H. Pedro F. Felzenszwalb, «Efficient Graph-Based Image Segmentation,» *International Journal of Computer Vision,* vol. 59, nº 2, pp. 167-181, 2004.

[20] L. J. L. Zhuo Deng, «Unsupervised Segmentation of RGB-D Images,» *Asian Conference on Computer Vision,* pp. 423-435, 2014.

[21] M. P. F. D. Can Erdogan, «Planar Segmentation of RGBD Images Using Fast Linear Fitting and Markov Chain Monte Carlo,» *Ninth Conference on Computer and Robot Vision,* 2012.

[22] S. H. R. B. R. S. B. Dirk Holz, «Real-Time Plane Segmentation Using RGB-D Cameras,» *RoboCup 2011: Robot Soccer World Cup XV,* pp. 306-317, 2011.

[23] R. B. R. M. D. S. G. N. N. S. Holzer, «Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images,» *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2012.

[24] S. G. R. B. R. H. I. C. Alexander J. B. Trevor, «Efficient organized point cloud segmentation with connected components,» *3rd Workshop on Semantic Perception Mapping and Exploration (SPME),* 2013.

[25] G. C. S. Linda Shapiro, «Chapter 3. Connected components labeling,» de *Computer vision*, Prentice Hall, 2001, pp. 69-75.

[26] Aftershokz, «Bluez 2S,» [En línea]. Available: https://aftershokz.com/blogs/news/30023937-bluez-2-bluetooth-headphones-in-mashable. [Último acceso: Junio 2018].

[27] K. Collins, «An Introduction to Procedural Music in Video Games,» *Contemporary Music Review,* vol. 28, nº 1, pp. 5-15, 2009.

[28] S. S. F. A. Michele Geronazzo, «Mixed structural modeling of head-related transfer functions for customized binaural audio delivery,» *18th International Conference on Digital Signal Processing (DSP),* 2013.

[29] U. d. Jaén, «Grupo de Gráficos y Geomática de Jaén,» [En línea]. Available: https://gggj.ujaen.es/. [Último acceso: Junio 2018].

[30] Microsoft, «Kinect,» [En línea]. Available: https://developer.microsoft.com/es-es/windows/kinect. [Último acceso: Junio 2018].

[31] «Raspberry Pi 3,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Último acceso: Junio 2018].

[32] Google, «ARCore,» [En línea]. Available: https://developers.google.com/ar/. [Último acceso: Junio 2018].

[33] Microsoft, «RealSense,» [En línea]. Available: https://realsense.intel.com/stereo/?_ga=2.213890004.1172903553.1530203345-697842656.1530203345. [Último acceso: Junio 2018].

[34] «OpenCV,» [En línea]. Available: https://opencv.org/. [Último acceso: Junio 2018].

[35] «Point CLoud Library PCL,» [En línea]. Available: http://pointclouds.org/. [Último acceso: Junio 2018].

[36] «YSE audio engine,» [En línea]. Available: http://www.attr-x.net/yse/. [Último acceso: Mayo 2018].

[37] «Unity,» [En línea]. Available: https://unity3d.com/es. [Último acceso: Junio 2018].

[38] L. Vandevenne, «LodePNG,» [En línea]. Available: https://lodev.org/lodepng/. [Último acceso: Mayo 2018].

[39] Google, «NDK,» [En línea]. Available: https://developer.android.com/ndk/. [Último acceso: Junio 2018].

[40] Microsoft, «Características de C++11/14/17,» [En línea]. Available: https://msdn.microsoft.com/es-es/library/hh567368.aspx. [Último acceso: Junio 2018].

[41] L. B. X. R. D. F. Kevin Lai, «Sparse distance learning for object recognition combining RGB and depth information,» *IEEE International Conference on Robotics and Automation,* 2011.

Manual de usuario

En este manual de usuario se explicará el funcionamiento de la aplicación DEyes, hay que tener en cuenta que es un prototipo, y la interfaz está pensada para facilitar la depuración y testeo.

En la aplicación solo existe una vista, Ilustración 0.1, desde la que se controla toda la funcionalidad.

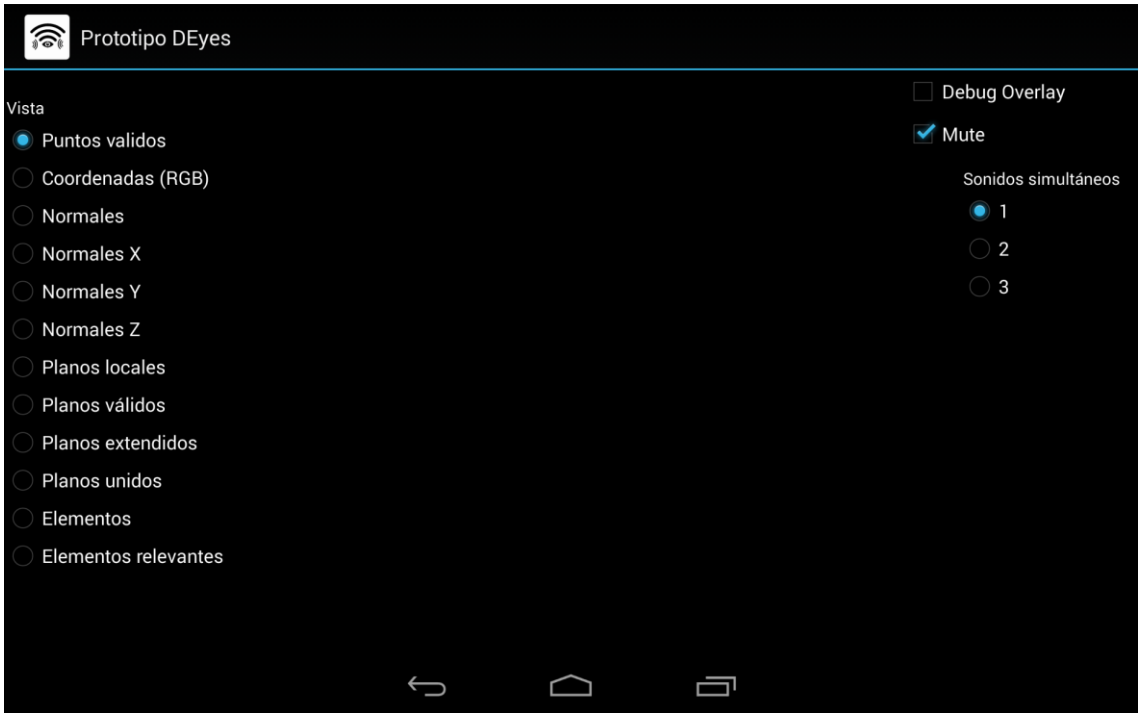


Ilustración 0.1 - Interfaz de la aplicación

El botón “Debug Overlay” activa o desactiva la capa de depuración, el contenido depende del modo de visión seleccionado. La barra de desplazamiento inferior sirve para hacer transparente u opaca la vista de depuración, por defecto está al 50%, lo que permite ver la depuración superpuesta a la cámara de color.

El botón “Mute” activa o desactiva el sonido, el número de sonidos simultáneos se puede elegir con el grupo de botones “Sonidos simultáneos”. Solo se reproducirá sonido si la vista “Elementos Relevantes” está activa y el botón “Mute” desactivado.

El grupo de botones “Vista” permite seleccionar las diferentes imágenes creadas durante la segmentación. Para no volver a incluir imágenes, se pondrán las referencias a las imágenes del apartado 3.2 Segmentación en Android.

- Puntos válidos: se muestran en blanco los puntos que el sensor ha sido capaz de detectar. Ilustración 3.18 - Puntos válidos
- Coordenadas (RGB): se muestran las coordenadas codificadas con colores, rojo x, verde y, y azul z. Ilustración 3.19 - Coordenadas rgd
- Normales: se muestran las normales en cada punto codificadas con colores, rojo x, verde y y azul z. Ilustración 3.20 - Normales rgb
- Normales X: se muestra la componente x de la normal en cada punto codificada como escala de grises. Ilustración 3.21 - Coordenadas X en escala de grises
- Normales Y: se muestra la componente y de la normal en cada punto codificada como escala de grises.
- Normales Z: se muestra la componente z de la normal en cada punto codificada como escala de grises.
- Planos locales: se muestra los planos locales en colores aleatorios. Ilustración 3.22 - Planos locales
- Planos válidos: se muestra los planos válidos en colores aleatorios. Ilustración 3.23 - Planos locales válidos
- Planos extendidos: se muestra los planos extendidos en colores aleatorios. Ilustración 3.24 - Planos extendidos
- Planos unidos: se muestra los planos extendidos y unidos en colores aleatorios. Ilustración 3.25 - Planos unidos
- Elementos: se muestran todos los elementos, planos o no, válidos que se han detectado. Ilustración 3.26 - Elementos
- Elementos relevantes: se muestran los 3 elementos más relevantes, en orden de relevancia, rojo, verde y azul, y el suelo en blanco. Ilustración 3.27 - Elementos relevantes y suelo

Manual de desarrollo

En este manual se comentarán los pasos necesarios para importar los proyectos incluidos con la memoria, y seguir ampliando o modificando.

Proyecto Visual Studio

La versión de escritorio ha sido desarrollada usando el IDE Microsoft Visual Studio 2017, en concreto la versión 15.4.3.

La única dependencia es LodePNG [38], que está incluida.

Los archivos de proyecto del IDE están incluidos, por lo que se podría importar con la versión 2017 o más reciente, sin problemas.

Proyecto Android Studio

La versión para Tango, sí que tiene más dependencias, en concreto el SDK de Tango y la librería Superpowered [8], todas ellas están incluidas junto al proyecto.

Al igual que el proyecto de Visual Studio, se han incluido los archivos para importarlo en el IDE, para lo que es necesario una versión 3.0 de Android Studio o superior, con los siguientes complementos:

- Android NDK 16.0 o superior
- SDK para Android 4.4
- Gradle Versión 3.1.2 o superior

Para la construcción de la versión release de la apk, ha sido necesario firmar la aplicación, esta firma se encuentra en el archivo TFM.jks, dentro del proyecto. Para usarlo se tiene que usar la siguiente información:

- Nombre de la firma: key0
- Contraseña de la firma: tfmtfm

Descripción de los anexos

Junto a esta documentación se entregan una serie de anexos:

- Imágenes, todas las imágenes presentes en este documento con la resolución original.
- Proyectos, en esta carpeta se incluyen los dos proyectos, Android Studio y Visual Studio, desarrollados durante el TFM.
- Sonido, el sonido usado como base en la generación de sonidos 3D.
- Video, el video de demostración de la aplicación, para evitar problemas de reproducción, se ha subido una copia a la plataforma YouTube, https://youtu.be/l5kI2oR84lk
- DEyes-release.apk, la aplicación final del proyecto, solo se puede instalar en dispositivos Tango.
- TFM_AdrianLuque.pdf, la presente documentación.