Información Importante

La Universidad Santo Tomás, informa que el(los) autor(es) ha(n) autorizado a usuarios internos y externos de la institución a consultar el contenido de este documento a través del Catálogo en línea del CRAI-Biblioteca y el Repositorio Institucional en la página Web de la CRAI-Biblioteca, así como en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

Se permite la consulta a los usuarios interesados en el contenido de este documento, para todos los usos que tengan **finalidad académica**, nunca para usos comerciales, siempre y cuando mediante la correspondiente cita bibliográfica se le dé crédito al trabajo de grado y a su autor.

De conformidad con lo establecido en el Artículo 30 de la Ley 23 de 1982 y elartículo 11 de la Decisión Andina 351 de 1993, la Universidad Santo Tomás informa que “los derechos morales sobre documento son propiedad de los autores, los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.”

Aplicativo móvil para el registro, almacenamiento y procesamiento de imágenes aéreas asociadas a accidentes de tránsito, facilitando la determinación de distancias y ubicación de involucrados

Saúl Eduardo Villamizar Camelo

Trabajo de grado para optar el título de Ingeniero de Telecomunicaciones

Director

Sergio Andrés Zabala Vargas

Magister en Administración de Proyectos.

Universidad Santo Tomas, Bucaramanga

División de Ingenierías y arquitectura

Facultad de Ingeniería de Telecomunicaciones

junio de 2017

Dedicatoria

A mis padres, SAUL VILLAMIZAR y PATRICIA CAMELO quienes confían en mí y cada día brindan su apoyo incondicional.

A mi hermano, JOSE MIGUEL VILLAMIZAR quien me hace sentir el orgullo de ser un hermano mayor.

A JANETH ROXANA quien con su constante apoyo y amor me da las fuerzas para seguir superándome a mí mismo día a día.

Gracias a ustedes por sus constante persistencia y ayuda en cada paso de mi vida.

Agradecimientos

Agradezco a Dios y a la vida, por ponerme en el camino a personas que día a día aportan un gran conocimiento y enseñanzas que me hacen crecer.

Agradezco a mis profesores por compartir sus conocimientos y experiencias en este gran paso de mi vida, en especial a mi director de proyecto por su gran dedicación y perseverancia.

Tabla de Contenido

Lista de tablas

Lista de figuras

Introducción

El presente proyecto de grado presenta un desarrollo software para plataformas móvil, con el fin de mejorar la captura de datos en los accidentes de tránsito en Bucaramanga. El cual le brinda al usuario las medidas, entre puntos elegidos por el mismo y almacenar los datos, con el fin de colaborar con la dirección de tránsito de Bucaramanga para poder tener pruebas más sólidas al momento de la toma de decisiones.

Palabras claves: Drones, Android, Length measurement, Vehicles, Cameras, Computer Vision.

1 Aplicativo móvil para el registro, almacenamiento y procesamiento de imágenes aéreas asociadas a accidentes de tránsito, facilitando la determinación de distancias y ubicación de involucrados

1.1 Formulación del problema

Según un estudio en la Universidad de Antioquia del experto en seguridad vial Gustavo Alonso Cabrera, hay más de 150.000 personas que han quedado con consecuencias de por vida tras sufrir un choque automovilístico. Además, el gobierno nacional reconoce y asegura que más gente muere en las vías que por el conflicto armado [1].

En el año 2014 entre enero y noviembre ocurrieron 31.050 accidentes de tránsito y 13.921 lesionados, dejando 172 personas fallecidas, esto implicó un incremento del 18 por ciento en comparación al año 2013 [2].

Debido a estas alarmantes cifras es importante considerar mecanismos para el seguimiento y trazabilidad de los accidentes, para poder realizar la toma de decisiones y medidas al respecto. Es vital recordar que las autoridades de transito son las responsables de este proceso, a través de la toma y consolidación de informes de tránsito, donde se requiere recolectar datos de forma actualizada y completa, para poder realizar estadísticas cuantitativas y cualitativas, permitiendo de esta forma poder tener una evaluación del riesgo, frecuencia y severidad de los accidentes de tránsito ocurridos [3].

En el contexto regional, se ha determinado que la Dirección de Transito de Bucaramanga (responsable del tema en la ciudad) no tiene implementado ningún proceso de registro de accidentes con herramientas de Tecnología de Información y Comunicación (TIC); a pesar que el Ministerio de Transporte estableció en la resolución número 0011268 de 2012, artículo 3 el cual dice que “Las autoridades de transito podrán implementar nuevas tecnologías que

permitan la captura, diligenciamiento, almacenamiento y lectura de la información contenida en el IPAT.” [4].

El grupo de investigación UNITEL, adscrito a la Facultad de Ingeniería de Telecomunicaciones, llevó a cabo el desarrollo del proyecto de investigación: Implementación de una metodología para el acceso, registro, análisis y trazabilidad de información para el informe policial de accidentes de tránsito (IPAT); dirigido a las necesidades de las direcciones de tránsito y transporte; en el marco de la VIII Convocatoria de proyectos internos, logrando contar con un aplicativo software que permite el registro, almacenamiento y transmisión de información del IPAT.

A pesar de ese aporte el registro de información para el croquis se sigue haciendo manualmente por ello es posible contar con la contribución de las TIC para realizar el registro fotográfico.

Debido a que cada Dirección de Transito puede aplicar las TIC para poder optimizar el Informe Policial de Accidentes de Tránsito (IPAT), el cual fue adoptado en la resolución número 0011268 de 2012 por el ministerio de transporte. En base a ello, se opta por realizar un aporte al registro grafico de un accidente de tránsito a través del uso de un Drones y de metodologías de Procesamiento digital de imágenes, para poder facilitar la captura y lectura de la información.

Esta iniciativa surge a partir de una revisión global de la literatura, donde se pueden observar casos en los que se está implementando esta tecnología con el fin de realizar una mejora en los procesos de los accidentes de tránsito. En Japón, por ejemplo, actualmente la compañía de seguros Sompo Japan Nipponkoa Insurance, se encuentra como un pionero del país en la implementación de drones para llevar el control sobre los accidentes de tránsito [5]. A su vez, la Comisión de Transito del Ecuador (CTE) se encuentra en la utilización de drones

sobre los accidentes de tránsito con el fin de realizar un proceso de mejora en las investigaciones realizadas por el CTE [6].

1.2 Justificación

El levantamiento de un croquis debe reconstruir un accidente de tránsito, ya que este documento sirve de base para la construcción de las teorías que puedan confirmar o desmentir lo que digan los implicados [7].

Sin embargo, en la actualidad los oficiales se encuentran haciendo dichos croquis como dibujos a mano alzada, donde se toman varias fotos y se mide con un metro las distancias de los afectados. Esto produce que, en la mayoría de los casos, se dejen de recolectar datos importantes, que los datos se vean afectados o simplemente no sean lo suficientemente exactos. Todos estos elementos causan problemas a la hora de llegar a una conclusión sobre el culpable del accidente y haciendo más complicada la toma de decisiones, afectando a los jueces encargados de estos casos cuando los accidentes deben ir a juicio como a las aseguradoras de los vehículos para medir los daños causados.

Para poder lograr una reconstrucción de hechos actualizada y completa, es necesario aplicar recursos tecnológicos los cuales no se están aprovechando actualmente. Por ello, se debe realizar este aporte a los problemas que están ocurriendo con los croquis realizados por los agentes de tránsito.

Al aplicar las TIC se obtendrán imágenes más acertadas y datos más confiables. Se puede observar en el informe “Las Tic y su aplicación en la seguridad vial” realizado en Costa Rica en el 2010, el cual habla sobre cómo se han aplicado las TIC a lo largo de los años en los proyectos de seguridad vial. Se menciona que:

El éxito actual y futuro de las herramientas que conforman los TIC, se basa principalmente en:

- La no existencia de otras soluciones, incapaces de solucionar por si mismas problemas relacionados con eficiencia y capacidad.
- La universalización de los medios de comunicación e información
- La disminución de sus costos en los últimos años
- El aumento de estándares de vida, para lo que se requiere una mayor seguridad, y un mejor aprovechamiento del tiempo e información [8].

Por ello es crucial la utilización de las TIC en estos proyectos para minimizar la afectación de la información por el error humano y que a su vez con una inversión en proyectos de las TIC en la seguridad vial, se van a poder reducir costos en los procesos que implica un accidente de tránsito; ya que, al tener una imagen aérea del accidente se podrá tomar una decisión más óptima con respecto al hecho ocurrido.

Recopilando, el presente proyecto permitirá aportar en mejoramiento de la seguridad bien, en la toma de decisiones judiciales más acordes con la realidad, el apoyo a las aseguradoras al momento de verificar daños y aplicación de pólizas y al sistema de salud para llevar un mejor control y seguimiento de los afectados.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un aplicativo software para facilitar el registro y extracción de distancias (medidas) de una imagen aérea con el fin de facilitar la realización del croquis del Informe Policial de Accidentes de Tránsito (IPAT).

1.3.2 Objetivos Específicos

- Realizar una revisión bibliográfica sobre la utilización de drones para aplicaciones de seguimiento de accidentes, desastres, entre otros.
- Diseñar un prototipo de aplicativo software para móvil que permita el registro y procesamiento de las imágenes capturadas por un dron; estableciendo la metodología de uso y orientado a la obtención de distancias del accidente.
- Determinar las condiciones requeridas para el registro de la imagen aérea tomada por el dron, determinando recomendaciones y restricciones
- Validar el aplicativo, a través de una prueba piloto con condiciones controladas, que permita establecer capacidades y limitaciones de la aplicación.

2 Marco de Referencia

2.1 Marco conceptual

Para la correcta realización del proyecto se necesitan la investigación y profundización en diversos conocimientos, los cuales se pueden observar de forma sintetizada en esta sección.

2.1.1 Transmisión de Video e Imágenes (Streaming)

La tecnología de Streaming se utiliza para realizar descargar y reproducciones de archivos de audio y video que suelen tener cierto peso. [9]

Como se puede observar en la Figura 1 Se envia una señal de audio/video a los servidores de streaming (broadcast servers) y sus oyentes se conectan al servidor de streaming atraves de internet para escuchar la señal de audio o ver el video en directo. [10]

Figura 1 Cómo funciona el Streaming. Adaptado de [10]

2.1.2 Android

Android es el sistema operativo utilizado en más de mil millones de teléfonos inteligentes y tablets. Cada versión de Android hace que algo nuevo sea posible. [11] Centrándose en el proyecto se enfocará a dos versiones en particular la 4.1 Jelly Bean y la 4.4 KitKat, se ha tomado esta decisión ya que son las dos versiones más populares en el mercado; motivado en que la mayoría de los dispositivos de gama media las tienen y también cabe resaltar que son en las que se desarrollan más aplicaciones.

2.1.2.1 Android Jelly Bean

La actualización única del año 2012, Android 4.1 Jelly Bean se estrenó en el evento de desarrolladores de Google en el Google I/O 2012. Las mejoras que añade Android 4.1 Jelly Bean [12] son sintetizadas a continuación:

- Mejoras de rendimiento del sistema y de los gráficos con Project Butter
- Se mejora la entrada de datos táctiles
- Aparece Google Now, el servicio-asistente de voz inteligente de Google
- Se añade el navegador propio Google Chrome
- Búsqueda mediante voz mejorada
- Se rediseña la tipografía principal Roboto
- Las notificaciones interactivas de escritorio añaden novedades de diseño
- Se añaden mejoras relacionadas con los ajustes de tamaño de los widgets
- Se mejora la corrección ortográfica y la predicción en el teclado

- Dictado de voz offline

2.1.2.2 Android KitKat

La siguiente definición enmarca, en términos generales, el concepto de esta versión de Android comparada con la descrita en la sección anterior.

"Android 4.4 KitKat ofrece una experiencia más eficiente y con menor consumo de recursos. También cambia visualmente el aspecto del teclado del teléfono; permite el uso de iconos Emoji; aplicaciones a pantalla completa y mejoras de productividad, como la inclusión de QuickOffice para acceder desde la nube a todo nuestro contenido de textos, bases de datos y presentaciones." [13]

2.1.3 Android Studio

Sabiendo que se trabajará con Android para el desarrollo del proyecto, es hora de hablar sobre cómo se desarrolla en este sistema operativo, Android aparte de contar con la gran ventaja de ser un sistema operativo libre, también, es un poco más sencillo de programar, ya que, utilizamos un IDE llamado Android Studio, esta es una herramienta que nos facilita el desarrollo para dispositivos Android, como es bien sabido, Android utiliza el código Java como su código de raíz, por ello al principio se venía utilizando la herramienta Eclipse para el desarrollo de las aplicaciones, para facilitarle el trabajo a los desarrolladores, google nos ofrece este nuevo IDE llamado Android Studio.

Esta herramienta es un nuevo entorno de desarrollo integrado para el sistema operativo Android lanzado por Google, diseñado para ofrecer nuevas herramientas para el desarrollo de aplicaciones y alternativa al entorno Eclipse [14]

Al crear un nuevo proyecto en Android Studio, la estructura del proyecto aparece con casi todos los archivos dentro del directorio SRC, un cambio a un sistema de generación basado Gradle que proporcionará una mayor flexibilidad para el proceso de construcción. Además, gracias a su sistema de emulación integrado, Android Studio permite ver los cambios que realizamos en nuestra aplicación en tiempo real, pudiendo además comprobar cómo se visualiza en diferentes dispositivos Android con distintas configuraciones y resoluciones de forma simultánea (ver Figura 2).

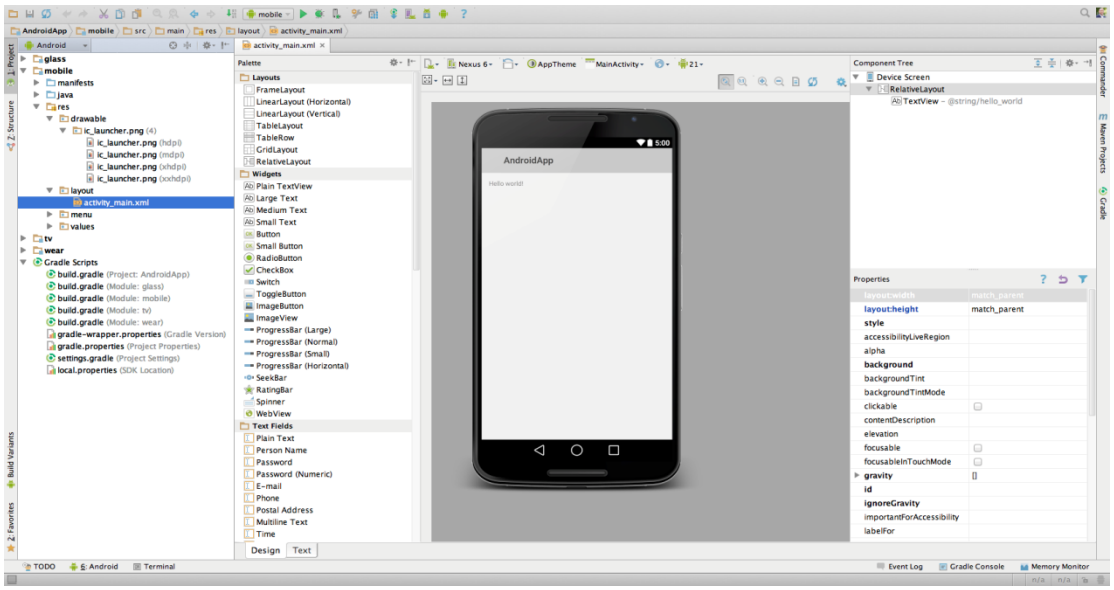

Figura 2 Android Studio. Adaptado de [15]

Entre las muchas características de Android Studio destacan sus herramientas de empaquetado y etiquetado de código que permite organizar de una mejor manera al implementar grandes cantidades de código, sirviéndose además de un sistema drag & drop para mover los componentes a través de la interfaz de usuario. Además, este nuevo entorno cuenta con Google Cloud Messaging, una funcionalidad con la que se puede enviar datos desde el servidor a terminales Android a través de la nube, siendo esta una forma de enviar notificaciones Push a las distintas aplicaciones [15]

2.1.4 UAV (Vehículo Aéreo no Tripulado)

Para hablar de los UAV primero se debe conocer un poco de su historia. Todo comienza el 22 de agosto de 1849, cuando el ejercito austriaco lanzo globos llenos de explisivos hacia la ciudad de Venecia. Al llegar a la vertical de la ciudad se hacian caer mediante un dispositivo electronico. [16].

En 1915 Nikola Tesla propone al Reino Unido una flota de vehiculos aereos no tripulados. La idea se materializo con el Aerial Target o AT (ver Figura 3), pensando para la defensa contra zeppelines y como una bomba dirigida.

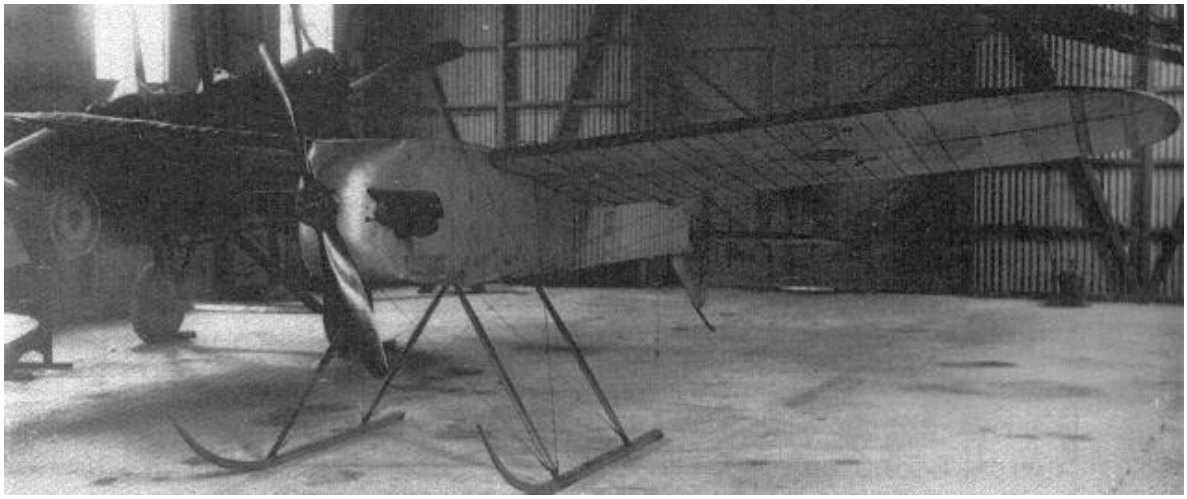

Figura 3 El Aerial Target. Adaptado de [17]

En 1931 se realizan pruebas con éxito del primer UAV, el Queen Bee, desarrollado a partir de un hidroavión biplano Fairey IIIF. El cual se pilotaba por medio de radiocontrol desde un barco. Se utilizó por la Marina como blanco de pruebas hasta 1943 [16]

En 1994 Se realiza el primer vuelo de un RQ-1 Predator (ver Figura 4), el primer UAV operativo que usa GPS en lugar de estar programado o usar imágenes de televisión. [16]

Figura 4 RQ-1 Predator. Adaptado de [18]

En 2010 Se universaliza el uso de UAVs en el ámbito civil (tanto con objetivos comerciales como para ocio y recreo). Los países comienzan a legislar sobre su utilización en espacios abiertos [16]

Actualmente se pueden encontrar UAVs de bajo costo como lo es el Ardrone 2.0 (Ver Figura 5.), el cual consta de una cámara frontal HD (1280x720), tecnología WiFi 802.11 b/g/n, sensor de control absoluto y sensor de estabilidad a gran altitud, alcanzando una distancia máxima de 50 metros [19], es utilizado principalmente para ocio y recreo.


Figura 5 Ardrone 2.0. Adaptado de [19]

Sin embargo, se pueden observar UAVs comerciales con más capacidad, tal como el Phantom (ver Figura 6). El cual cuenta con un sistema de posicionamiento visual, ideal para áreas que no cuentan con una cobertura GPS, un sistema ruso de geo posicionamiento satelital, denominado Glonass, controladores de nueva línea que permiten al operador un acceso a las vistas en HD en una distancia de hasta 1.6 Kilómetros, una cámara de 4k y una capacidad de registrar imágenes de 12 MP junto con una capacidad de hasta 94 grados, a esto se le suma un sistema de estabilización para lograr mejores tomas aéreas [20].

Figura 6 Phantom 3. Adaptado de [20]

2.1.5 Procesamiento de imágenes

EL procesamiento de imágenes tiene como objetivo mejorar el aspecto de las imágenes y hacer más evidentes en ellas ciertos detalles que se desean hacer notar. La imagen puede haber sido generada de muchas maneras, por ejemplo, fotográficamente, o electrónicamente. El procesamiento de las imágenes se puede en general hacer por medio de métodos ópticos, o bien por medio de métodos digitales. [21]

2.1.5.1 Imágenes digitales

Si el dominio (valores de (x , y)) y el rango (valores de f (x , y)) son continuos, la imagen es continua o análoga; si el dominio y el rango son discretos, la imagen es digital,

para convertir una imagen de tonos continuos en formato digital, la imagen análoga es dividida en valores de brillos individuales a través de dos procesos denominados muestreo y cuantificación. [22]

2.1.5.2 Muestreo

El muestreo consiste en la tarea de discretizacion del escenario en el plano sensor, la imagen es convertida en una matriz discreta de MxN pixeles, por ello el muestreo está relacionado con la resolución espacial de la imagen [23], tal como se puede apreciar en la Figura 7 Resoluciones de una toma sobre el mismo escenario.

Figura 7 Resoluciones de una toma sobre el mismo escenario. Adaptado de [23]

2.1.5.3 Cuantificación.

La cuantificación de la señal consiste en el número finito de valores de cada píxel. Los niveles de cuantificación suelen ser potencias de 2 para facilitar su almacenamiento en el computador. En general, se suele emplear un byte de

cuantificación por píxel, consiguiendo 256 niveles de grises. El cero será el color negro y el valor 255 corresponderá con el blanco. Entre el cero y el valor 255 estarán los diversos valores de grises. Para el caso del color, la cuantificación se vuelve vectorial; por cada píxel se representara una terna de valores que haga reflejar la luminancia y crominancia en cada píxel. [23]. Un ejemplo claro de esto se puede ver en la Figura 8. Niveles de cuantificación en una imagen.

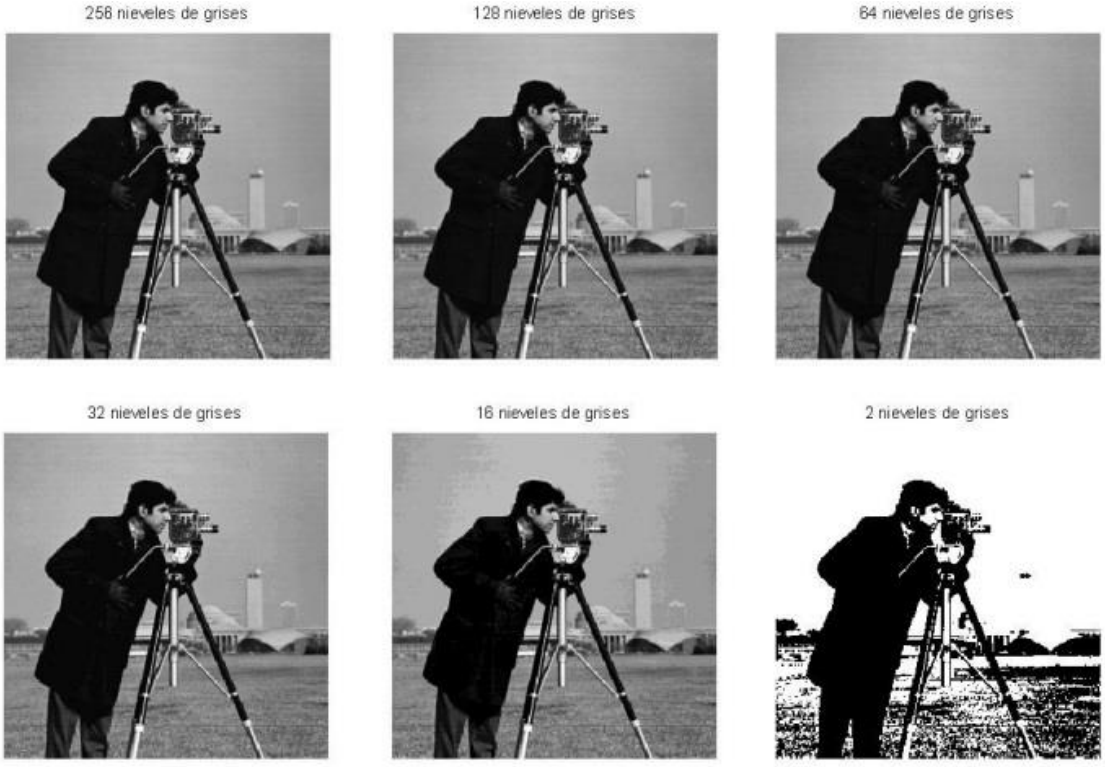

Figura 8 Niveles de cuantificación en una Imagen. Adaptado de [23]

3 Ejecución y resultados del proyecto de grado

En este capítulo se utilizara una tabla para la presentación de los resultados del proyecto, asociados a los objetivos del mismo. Esta información se observa en la Tabla 1.

Tabla 1.

Relación de entregables del proyecto de grado

Criterio	Descripción
Marco de Antecedentes	Se realiza una revisión bibliográfica de los últimos 7 años a partir de la autoría del presente proyecto; realizando su análisis y presentación.
Metodología Propuesta	Se ha planteado una metodología de desarrollo evolutivo para la realización del mismo.
Diseño del software	Se presentan los requerimientos y el diagrama de flujo el aplicativo; como base para la orientación del software a desarrollar.
Prototipo software	Se realiza la implementación de la solución en Android Studio como versión prototipo del aplicativo.
Pruebas en ambientes controlados	Se han realizado pruebas en ambientes controlados, las cuales se presentaran con sus respectivos resultados.

Nota: Explicación de la estructura para la ejecución del proyecto.

3.1 Marco de antecedentes

3.1.1 Revisión de aplicaciones de seguimiento de accidentes con Drones

Los criterios de búsqueda para la revisión realizada son presentados en la Tabla 2

Tabla 2

Criterios de Revisión

Criterio	Descripción
Periodo de revisión	Últimos siete años a partir de la autoría del presente proyecto (2010-2017)
Bases de datos/repositorios	Scopus, e IEEE.
Condiciones de revisión	• Las palabras claves de búsqueda serán principalmente: Drones, Vehicles, Computer vision, Cameras, Roads, Android, Length measurement. • Los trabajos seleccionados deben evidenciar impacto en el sector productivo de aplicación. • Se deben encontrar los artículos en texto completo.

Nota: Criterios de evaluación para la búsqueda de los diferentes proyectos utilizados en el marco de antecedentes

Tabla 3

Revisión general del estado del arte de la iniciativa

NOMBRE DEL ARTICULO	INSTITUCIÓNES INVOLUCRADAS	AÑO DE PUBLICACIÓN	RESUMEN	IMPORTANCIA EN LA REVISIÓN DE INTERÉS PARA UNITEL
Fast and accurate car detection in drone-view	• Consumer Electronics-Asia (ICCE-Asia). • IEEE International	2016	Con el desarrollo de los Drones, las imágenes aéreas son utilizadas en una variedad de aplicaciones, se ha propuesto una forma de detectar carros con la vista del dron de una forma rápida y certera. Por este propósito se ha propuesto una característica llamada G.-ORF. Además, se ha diseñado un clasificador y una ancha línea de discriminación de análisis, clasificada para cada pose y clasificación binaria por cada pose respectivamente. Nuestro método muestra en tiempo real una presentación	De este proyecto se destaca la importancia de la evolución en la calidad de video de los drones, debido a ello, se pueden implementar aplicaciones móviles o de escritorio con el fin de aprovechar esta

			en video de calidad HD (1280x720), en un ambiente de PC con una elevada exactitud en la detección.	alta calidad y generar mejores resultados.
A drone-base image processing system for car detection in a smart transport infrastructure	• Electro technical Conference (MELECON) • 2016 18th Mediterranean	2016	En este documentó presentamos un prototipo de un sistema para la detección de carros con un proyecto de experimento, El cual analiza una proyección de video grabada por Drones volando sobre un ambiente urbano. Con la intención de proveer una información de ayuda, como los lugares de parqueo disponibles y el nivel de congestión de las calles. El sistema se ha probado en computadoras y en sistemas embebidos. Los resultados del experimento muestran una alta precisión,	En este proyecto se destaca la utilización de los drones en sistemas de tránsito, en este caso, para la detección de lugares de estacionamiento y zonas de alto tráfico.
A computer-vision approach to	• Intelligent Transportation	2016	En recientes años, se ha fijado un interés en el detalle del monitoreo del tráfico de las vías,	En este proyecto se puede destacar

traffic analysis over intersections	Systems(ITSC) • 2016 IEEE 19th International conference		en particular para las intersecciones, en orden para obtener un modelo de la elevación de los vehículos. Estos modelos han apuntado a la optimización del manejo del tráfico permitiendo un sistema de transporte inteligente. Cuando los métodos de los sensores convencionales para cada intersección de entrada/salida permiten el conteo, están limitados en el sentido de que es imposible el seguimiento de un vehículo desde el origen hasta el destino. Una solución es presentada utilizando la visión por computadora, los algoritmos de seguimiento obtenidos desde los drones volando sobre una intersección, para poder seguir los vehículos y darnos un modelo estático quizá exacto, este modelo es base para la asociación de un origen	nuevamente la utilización de drones para un sistema de transporte inteligente, en el cual se realiza un seguimiento a los vehículos, este proyecto es de ayuda para dar una vista hacia la tendencia en el uso de drones en el ámbito de transportes.

			y un destino. El algoritmo emplea una sustracción de fundo y un seguimiento a vehículos con filtros Kalman, el reto más grande fue como compensar el movimiento de la cámara, basada en la implementación para la cual fue desarrollada, esta aplicación provee una utilidad para el seguimiento de carros, buses y camiones bajo ciertas circunstancias.	
Railway infrastructure monitoring by drones	• Electrical systems for Aircraft, Railways, Ship, Propulsion and Road Vehicles & International Transportation Electrification	2016	En recientes años he habido un crecimiento en el interés para la adaptación de los Drones en el monitoreo y la variedad de las infraestructuras. El monitorio de las infraestructuras están bastante adoptadas para la detección de errores antes de que estos se puedan convertir en severas fallas, ellos pueden ser encontrados en un número de estructuras	En este proyecto se destaca la utilización de los drones, no solo para la detección de lugares de parqueo, tráfico o seguimientos de vehículos, además, se puede ver una

	Conference (ESARS-ITEC) • International Conference.		civiles, incluyendo puentes, viaductos, y en muchas otras aplicaciones para un mantenimiento inteligente, pronostico, salud, y un monitoreo de seguridad. En este artículo se explora el uso de drones en líneas férreas para una temprana advertencia, evaluación y decisiones a tomar. El punto principal del estudio es para realizar una evaluación preliminar de las capacidades del evaluación del dron para el monitoreo de la infraestructura incluyendo las fallas estructurales y los hilos de seguridad detectados también la investigación de las consecuencias de los riesgos naturales y los ataques con intención.	utilización de los mismos para un modelo de evitar accidentes, en este caso ferroviarios.
Egg size classification on	• Student project Conference	2016	Los huevos de gallina son un ingrediente común en la comida humana, utilizado como un	Aun que el proyecto presente, no muestra

Android mobile device using image processing and machine learning	• 2016 fifth ICT International		ingrediente en casi todas las comidas del mundo. Juzgando el tamaño y el peso de los huevos es a veces importante en muchas recetas de comida. En este documento se propone un algoritmo de procesado de imágenes para la clasificación de los huevos por tamaño, las imágenes son mostradas en un dispositivo Android. Una moneda de un tamaño conocido es utilizada en la imagen como un objeto de referencia. El radio de la moneda y la dimensión del huevo son automáticamente detectados y medidos utilizando técnicas de procesamiento de imágenes. El tamaño de los huevos son clasificados en base a sus componentes desde las medidas dimensionadas utilizando una maquina vectorial de soporte	una utilización de drones, ha sido de gran utilidad, debido a que nos da una ventana en el uso de un dispositivo Android para realizar pruebas métricas, las cuales fueron de vital importancia en este proyecto.

			(MVS) clasificadora.	
Implementación de una estrategia para el acceso, registro, análisis y trazabilidad de información para el informe policial de accidentes de tránsito (ipat); dirigido a las necesidades de las organizaciones de tránsito y transporte.		2017	En este proyecto se determinó un conjunto de alineamientos, metodológicos y tecnológicos, con el fin de mejorar el proceso del registro, almacenamiento, transmisión y análisis de los accidentes viales; basado en las orientaciones propuestas por la incorporación del informe policial de accidentes de tránsito (IPAT), aplicado a la dirección de tránsito de Bucaramanga.	En este proyecto se destaca la utilización de nuevas tecnologías en para el Informe Policial de Accidentes de Tránsito (IPAT) en la dirección de tránsito de Bucaramanga, lo cual fue una base principal para este informe.
A survey of	• Unmanned Aircraft	2013	El foco principal de este documento es la	El articulo presente

unmanned aerial vehicles (UAVs) for traffic monitoring	Systems(ICUAS) • 201 International Conference on		examinación de los sistemas VAT para el monitoreo del tráfico, Aunque ha habido una gran cantidad de investigación sobre ello, los Vehículos Aéreos no Tripulados (VAT) han demostrado ser la vía más confiable y de menos consumo de tiempo para la alternativa del monitoreo en tiempo real del tráfico y la forma de proveer una vista aérea del problema	fue clave para la realización del estado del arte del proyecto, y una vista más detallada en la búsqueda de la aplicación de los drones en el monitoreo del trafico

Nota: Diferentes proyectos los cuales fueron tomados para el marco de antecedentes, con su debido aporte al proyecto.

Del ejercicio realizado anteriormente se pueden generar las siguientes conclusiones iniciales:

- El uso de los drones para el tráfico, control, seguimiento, y seguridad, en los aspectos de tránsito o viales, ha venido evolucionando en los resientes años, de forma exponencial.
- Los sistemas de calidad de video y control de los drones han venido evolucionando, lo que permite realizar aplicaciones más robustas, tales como las enmarcadas en el cuadro anterior, para el seguimiento de vehículos, búsqueda de lugares de estacionamiento, y la vista aérea para detectar el volumen de tráfico en una zona.
- Se puede observar una fuerte demanda en la utilización de drones, no solo para fines como el seguimiento de vehículos, sino también, para la observación de vías y detección temprana de accidentes.
- Se resalta la utilización de los aplicativos móviles para la medición de objetos en imágenes fotográficas, lo que muestra una ventana la cual al unirla con la alta calidad de las imágenes que los drones pueden ofrecer, se abren posibilidades de crear aplicaciones móviles precisas en la medición de datos.

3.1.2 Unmanned Aerial Vehicle (UAV) para la detección de accidentes.

Un caso particular del uso del video y el tratamiento de imágenes para el reconocimiento y control de tráfico vehicular, así como al detección y registro de accidentes; es el uso de drones o UAV (Vehículos aéreos no tripulados) para realizar la captura de las imágenes, facilitar el procesamiento de la información; permitiendo el análisis de diferentes situaciones en las vías. En la experiencia presentada en [24] se revisa la integración de los sistemas de visión artificial (AVS) en UAV; permitiendo facilitar el control del tráfico en las vías y el análisis automático de accidentes. Para tal fin, se realiza el registro de las imágenes, el escalamiento de las mismas, la generación de la región de interés (ROI), la segmentación de

la misma, el filtrado y generación de descriptores, para finalmente lograr el reconocimiento de vehículos y sus velocidades. En la **¡Error! No se encuentra el origen de la referencia.** se muestra un ejemplo del piloto realizado, donde recurriendo al procesamiento de imágenes áreas se logra detectar un posible accidente de tránsito de forma automática.

Figura 9 Comparación entre situación inicial y detección automática de accidente. Adaptado de [24]**.**

Ahora bien, en [25] y en [26] se realizan revisiones de literatura sobre el uso de los UAV en el monitoreo de tráfico. Se resalta la experiencia propuesta [27] donde se introduce una solución para el monitoreo de tráfico utilizando sistemas de radar e imágenes obtenidas de sistemas no tripulados. El eje central de esta investigación es la posibilidad de obtener imágenes de amplias zonas de terreno utilizando los UAV, e integrando con radares de apertura sintética (SAR) y sistemas de Indicación de movimiento de objetos en terreno (GMTI).

En [28] se presenta una alternativa de procesamiento de video basado en algoritmos de seguimiento (tracking), específicamente el algoritmo *Mean Shift*, con el cual se logran rápidos tiempos de convergencia y altas tasas de detección. En la **¡Error! No se encuentra el origen de la referencia.** se presenta el diagrama de flujo que desarrolla el algoritmo de filtrado propuesto por los autores de [28].

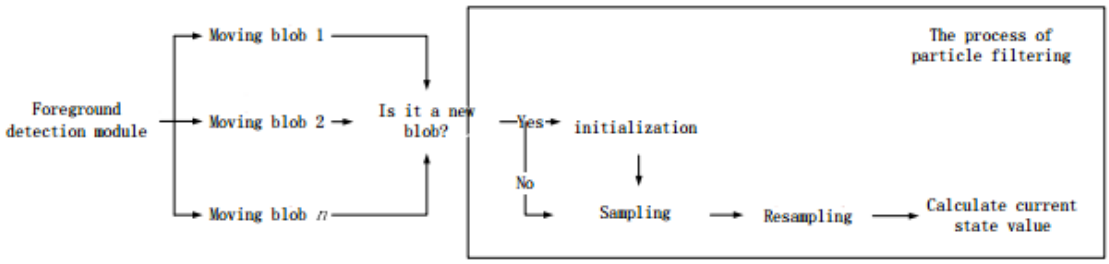

Figura 10 Diagrama de flujo propuesto para el filtrado MS.Adaptado de [28].

Finalmente, Puri, citado por [26] expone los siguientes prototipos o dispositivos comerciales, que integran UAV para la gestión y vigilancia del tráfico y control de accidentes:

- **Airbone Traffic Surveillance Systems (ATSS):** proyecto desarrollado por equipo de investigación de Universidad de Florida, el Departamento de Transporte de Florida (FDOT), el Aeropuerto comercial de Tallahassee, y la Universidad del Norte de Florida. El proyecto se orienta al monitoreo de áreas rurales y vías de la Florida, para caracterizar y monitorear vías e integrarlo con los sistemas de información existentes. En la **¡Error! No se encuentra el origen de la referencia.** se muestra un ejemplo del UAV (o aero-sonda como se denomina en el proyecto) utilizado en el sistema.

Figura 11 Ejemplo de la aero-sonda diseñada para el proyecto. Adaptado de [25]**.**

Es importante resaltar que la información captada por la sonda es remitida a una antena receptora en tierra, la cual se encarga de realizar la recepción y procedimiento del video; para ser remitido, a través de red de datos, al sistema de información del FDOT.

- **Sistema UAV de la Universidad Ohio State:** sistema desarrollado por el Departamento de Transporte de Ohio y la Universidad de Ohio State. Se desarrolló el prototipo de UAV denominado BAT, el cual cuenta con una altura máxima de 500 ft y hasta 30 km/h. BAT cuenta con un GPS elata precisión, un modem para transmisión de datos de 900 MHz y la capacidad de transmisión de video por 2.4 GHz. En la estación base de cuenta con un sistema automático de procesamiento y registro de situaciones anormales, como alta cantidad de tráfico, accidentes, bloqueos en las vías, entre otros. Un ejemplo de dicho prototipo es presentado en la **¡Error! No se encuentra el origen de la referencia.**.

Figura 12 Prototipo del sistema BAT de U. Ohio State. Adaptado de [25]**.**

Es términos generales esta sección ha permitido contar con un panorama general de herramientas tecnologías que contribuyen al registro de accidentes, al análisis y prevención

de los mismos y a la contribución del mejoramiento del tráfico vehicular en diferentes entornos viales. En el siguiente capítulo se detallan los productos obtenidos en el proyecto.

3.2 Metodología

La metodología del proyecto estará orientada al desarrollo de un aplicativo software que apoye el registro del croquis del Informe Policial de Accidentes de Tránsito, como apoyo en la gestión de la Dirección de Tránsito y Transporte.

La población a la que va dirigida el proyecto es principalmente los agentes operativos de la Dirección de Tránsito y Transporte, sin embargo se espera que el mismo genere impacto en las personas que se ven afectadas por la falta de datos o alteración de los mismos en el IPAT, las cuales son las aseguradoras, jueces, centros de salud, peatones, los conductores públicos y todos aquellos afectados en algún accidente de tránsito.

El proyecto está planeado en 4 (cuatro) grandes fases diferentes.

Fase 1: Iniciación del Proyecto.

Actividades:

1. Se realizará una revisión sobre cómo se encuentra el panorama actual de las TIC's utilizadas en sistemas de control del tráfico.
2. Se llevará a cabo una búsqueda sobre los componentes principales que deben ir integrados en la realización del proyecto
3. Se ejecutará una lectura sobre el funcionamiento de los componentes planteados anteriormente para su correcto uso en el proyecto.

Fase 2: Levantamiento de requerimientos y Diseño preliminar

Actividades:

1. Se plantearán las condiciones de registro de las imágenes y los requerimientos básicos del software para el cumplimiento de las recomendaciones del IPAT [4].

2. Se planteará el diseño preliminar del aplicativo software y un borrador de la interfaz de usuario.

Fase 3: Desarrollo de la aplicación

Actividades:

1. Se desarrollará una primera versión del software para dispositivos Android que permita realizar el registro y procesamiento de la imagen.

2. Se realizarán pruebas con un ambiente controlado para ver su correcto funcionamiento y realizar la corrección de errores presentados.

3. Se llevarán a cabo pruebas de campo determinando cuales serían las medidas en las que se desempeñe mejor.

Fase 4: Cierre del Proyecto.

1. Se realizará un informe con las pruebas de campo obtenidas donde se darán las especificaciones para el mejor funcionamiento del aplicativo móvil

2. Se presentará lineamientos y recomendaciones para mejorar futuras e incorporación en otras aplicaciones.

3.3 Descripción del alcance del software y requerimientos iniciales

3.3.1 Alcance detallado del software

El software debe permitir facilitar el registro de distancias entre puntos de referencia que ayuden al agente de tránsito (usuario) para calcular métricas del accidente, se proyecta que la aplicación tome como base la imagen registrada por un drone (vista superior) de toda la escena del accidente.

A su vez el software debe permitir el almacenamiento de los datos hallados con la aplicación en una base de datos local en el dispositivo, esta herramienta se podrá conectar con la solución de registro del IPAT desarrollada por el grupo UNITEL.

En la figura 11 se puede apreciar el Mock-up[1] inicial de la aplicación, la cual cuenta con todos los requerimientos mencionados anteriormente.

Figura 13 Mock-up Aplicación.

[1] Modelo o replica de una máquina, estructura, software o cualquier proyecto, el cual aún no se encuentra construido, con el objetivo de mostrar cómo podría quedar.

3.3.2 Lista de los requerimientos (funcionales y no funcionales):

1. Realizar el registro gráfico desde una vista superior, recurriendo posiblemente a un drone o similar.
2. Facilitar el registro de las distancias (medidas) entre los diferentes puntos de referencia.
3. Ejecutable en la mayoría de los dispositivos móviles o Tablets. El sistema operativo propuesto es Android.
4. Utilizar software libre para su desarrollo permitiendo no generar costes de licencia en su implementación o futuro uso.
5. Crear una base de datos local que permita almacenar nombre de la imagen y las medidas registradas en la imagen.
6. Permitir al usuario cargar imágenes desde la galería de fotografías del dispositivo.
7. Tener la capacidad de calcular un máximo de 3 (tres) puntos de longitud.
8. Permitir limpiar las líneas de medición en la pantalla.
9. Permitir el uso de diferentes calidades de imágenes.

3.4 Condiciones generales

En la siguiente tabla se pueden apreciar las condiciones generales para el correcto funcionamiento y desempeño del aplicativo móvil.

Tabla 4

Condiciones generales de la app

Condiciones Generales
El aplicativo debe ser instalado en móviles los cuales tengan un sistema operativo inferior al de Android 7.
Las imágenes que se carguen a la aplicación deberán ser capturadas desde una altura no mayor a 10 metros para poder ver con claridad los puntos de referencia.
Emplear el dron en días soleados, con el fin de obtener una imagen de buena calidad.
Cargar a la aplicación imágenes que no estén oscuras o sean fotos tomadas en la noche.
Utilizar una buena calidad en las imágenes, para poder detallar con precisión los puntos de referencia y los puntos de medición.
Usar la aplicación en dispositivos con pantalla grande, para poder detallar mejor la imagen.
Hacer uso de un lapicero magnético para seleccionar los puntos, debido a que esto va a dar una mayor precisión y por lo tanto se obtendrán mejores resultados.

Nota Lista de condiciones generales para el buen funcionamiento de la aplicación.

3.5 Diseño del software

La idea del diseño del aplicativo ha sido tomada de la aplicación ON 2D CAMERA MEASURE, la cual funciona en base a una línea guía y una línea para hallar las mediciones. Se ha optado por presentar en forma de diagrama de flujo el diseño del funcionamiento de la app, el cual se puede observar en la figura 12

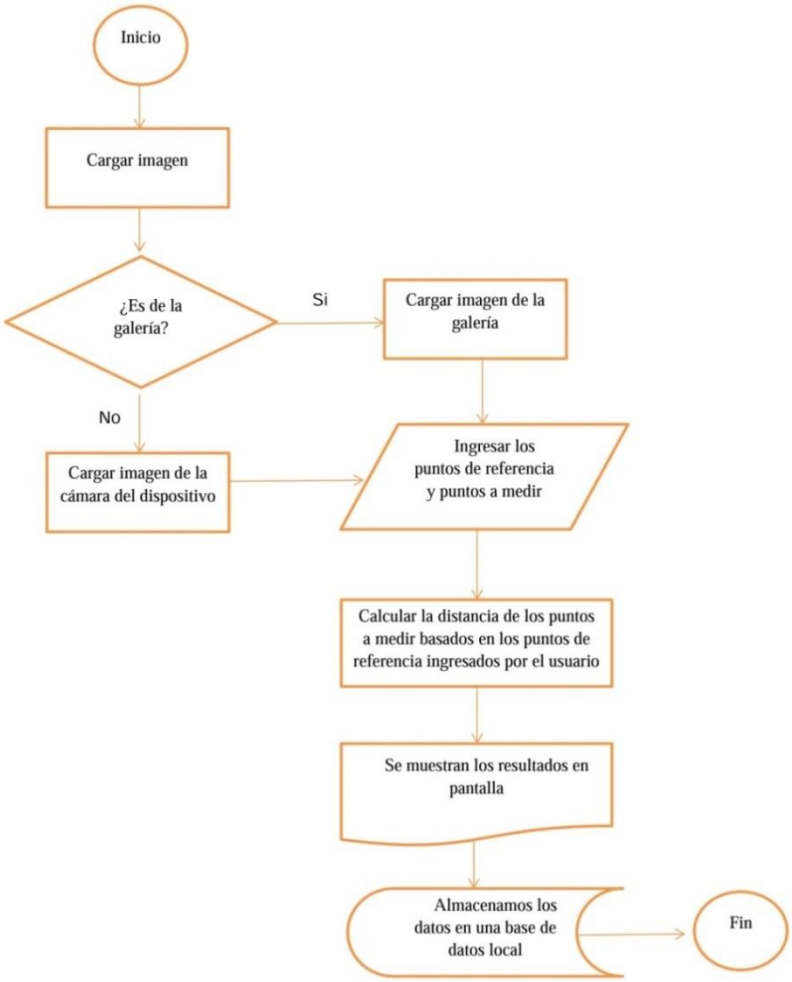

Figura 14 Diagrama de flujo del software

Tal como se puede observar en el diagrama de flujo, la estructura de trabajo del software comienza preguntando si la imagen a utilizar será capturada o subida de la galería del dispositivo, una vez se tenga la imagen (de la cámara o la galería), se procede a cargar la imagen en la ventana principal del software, donde se esperara el ingreso de los puntos de referencia del usuario, los serán de color amarrillo, seguidos por dos puntos de medición con un color rojo (el usuario tiene la posibilidad de agregar un máximo de 3 puntos de referencia y de medición), una vez puestos los puntos de referencia y de medición, se procede a calcular la distancia de los puntos de medición en base a los puntos de referencia ingresados anteriormente por el usuario, el o los resultados de las mediciones se muestran en una lista, en

la cual se pregunta al usuario, si desea guardar dicha información, la cual quedara almacenada en una base de datos local, con un nombre característico que el usuario digite y los datos de las mediciones.

3.5.1 Codificación del aplicativo

El desarrollo se llevó a cabo con Android Studio versión 2.3, recurriendo a diferentes librerías, las cuales van desde la librería Math para el soporte de las funciones matemáticas realizadas, hasta Draw, para poder pintar sobre la pantalla los diferentes puntos del diagrama, las principales clases de la aplicación son las listadas en la Tabla 4, los códigos de las distintas clases del proyecto se puede encontrar en la sección 5.7 Anexos del presente informe.

Tabla 5
Clases del Software

Clases del Software	Descripción
MainActivity	Es la clase principal del proyecto, debido a que en ella se inicia la aplicación, se realiza el llamado al layout principal el cual tiene como nombre activity_main, para poder darle una interfaz de trabajo al usuario, para así poder realizar las funciones principales del aplicativo móvil.
ImageSurface	Es la clase utilizada para poder manejar en la pantalla del dispositivo la imagen cargada o fotografía del usuario, es una de las clases más importantes, debido a que se encarga de pintar o

	dibujar, la imagen en la pantalla del dispositivo móvil, esta clase se realizó basada de un proyecto creado por el desarrollador Gonçalo.
DrawView	Es una clase muy importante debido a que se encarga de dibujar y eliminar los puntos de referencia y puntos de medición en la pantalla del dispositivo, a su vez, envía los datos a la clase Ruler, para realizar los debidos cálculos, esta clase se realizó basada de un proyecto creado por el desarrollador por el desarrollador Gonçalo.
Ruler	En esta clase se realizan todas las operaciones matemáticas del software (conversión de unidades, y la toma de las distancias).
Utils	En esta clase se tienen almacenadas todas las herramientas a utilizar por la aplicación para su correcto funcionamiento, tales como las fórmulas de conversión de unidades.
InputDialog	Esta clase se extiende a un DialogFrame, el cual le permite al usuario agregar los datos como la distancia del punto de referencia y la unidad métrica.
DatabaseHandler	En esta clase se crea la base de datos interna para almacenar los resultados, a su vez, se utiliza para el almacenamiento de los mismos.

Results	Es la clase encargada de mostrar en una lista los resultados almacenados en la base de datos, esta clase se realizó basada de un proyecto creado por el desarrollador por el desarrollador Douglas.
Values	Se guardan los valores utilizados por las clases DatabaseHandler y Results, esta clase se realizó basada de un proyecto creado por el desarrollador por el desarrollador Gonçalo.

Nota Explicación de las principales clases de la aplicación.

3.5.1 Uso de la aplicación- pasó a paso.

En esta sección se presentará el paso a paso que el agente de tránsito (usuario) debe realizar con el aplicativo, para su correcto funcionamiento.

1. Instalar el aplicativo a partir del apk denominado Grado_Metrico, encontrado en el siguiente link:

   https://www.dropbox.com/sh/zxjvn6cg4x16qpq/AAARrV4KcNRfmt31RdOvoVl0a?dl=0

2. Ingresar al aplicativo, en este momento, se cargara la interfaz de usuario la cual se muestra en la Figura 15.

Figura 15 inicio del aplicativo

3. Dar un click en “Tomar fotografía” para abrir la cámara del dispositivo y capturar la imagen, en caso de que ya tenga la fotografía en la galería de imágenes, seleccionar “Cargar Imagen”, al cargar la imagen se produce un letrero explicándole al usuario cómo funciona el proceso de agregar la medida de referencia y la de medición, como se observa en le Figura 16

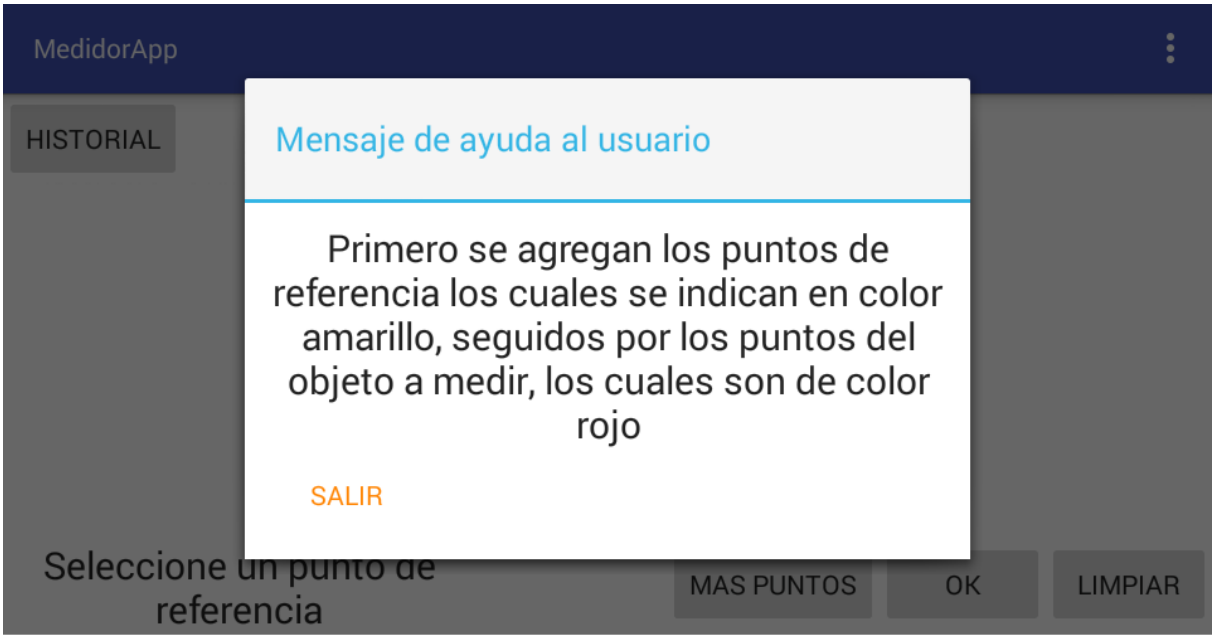

Figura 16 Mensaje de explicación del aplicativo

4. Se carga la imagen de la galería o la cámara fotográfica, en la pantalla principal de la aplicación, antes de ingresar los puntos de referencia y de medición, le dice al usuario el orden de ingreso como se observa en la Figura 16, una vez cargada se puede observar que la imagen se encuentra ahora en el fondo de la aplicación ver Figura 17.

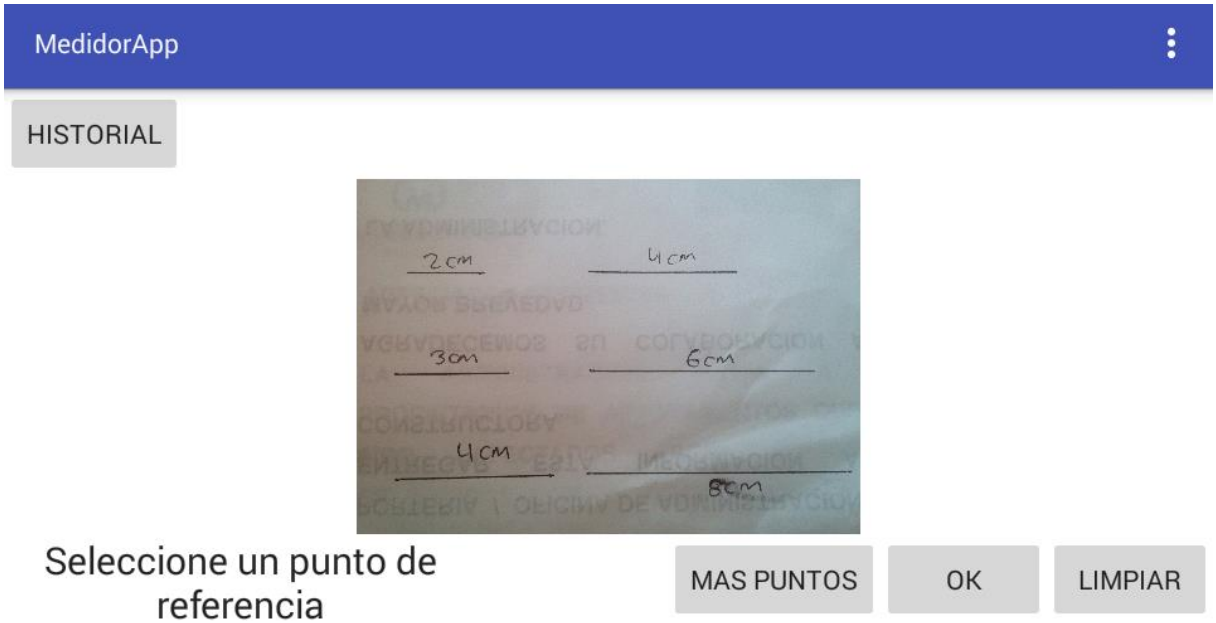

Figura 17 Imagen cargada en el aplicativo

5. Se tocan los puntos de referencia, los cuales aparecen con una línea de color amarillo, seguidos por los puntos a medir, los cuales aparecen con una línea de color rojo, tal como se puede observar en la Figura 18

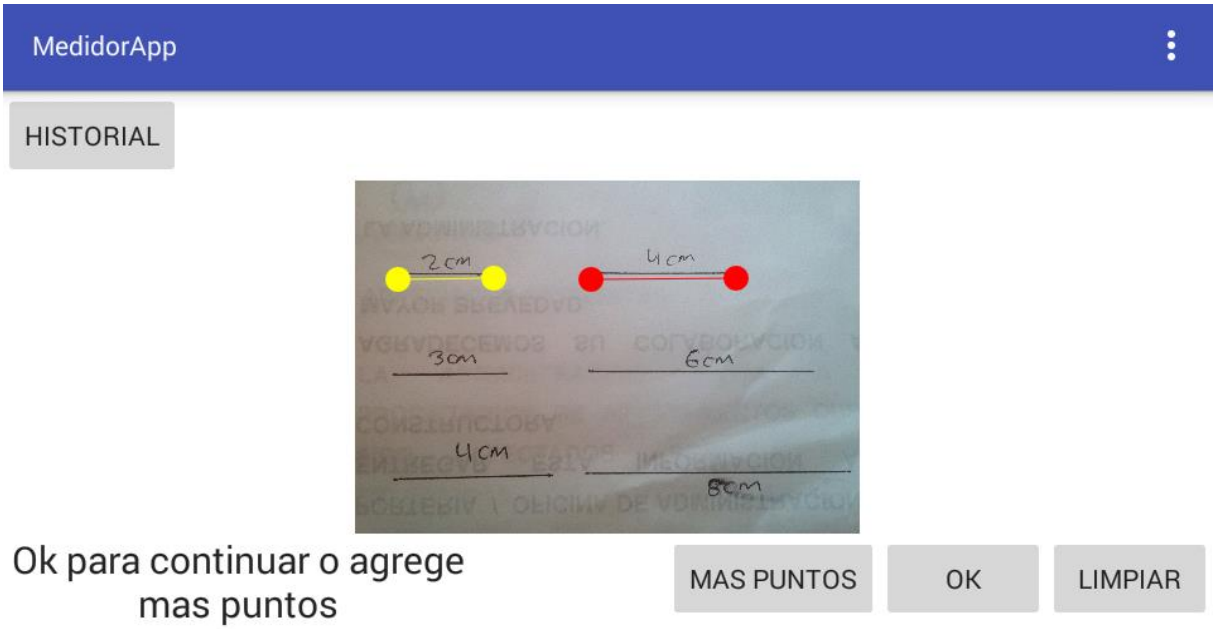

Figura 18 Puntos de referencia y medición

6. En caso de querer agregar más puntos de referencia y medición, oprimir click en “Mas Puntos” y repetir el paso 5 listado anteriormente, se irán pintando los nuevos puntos ver Figura 19.

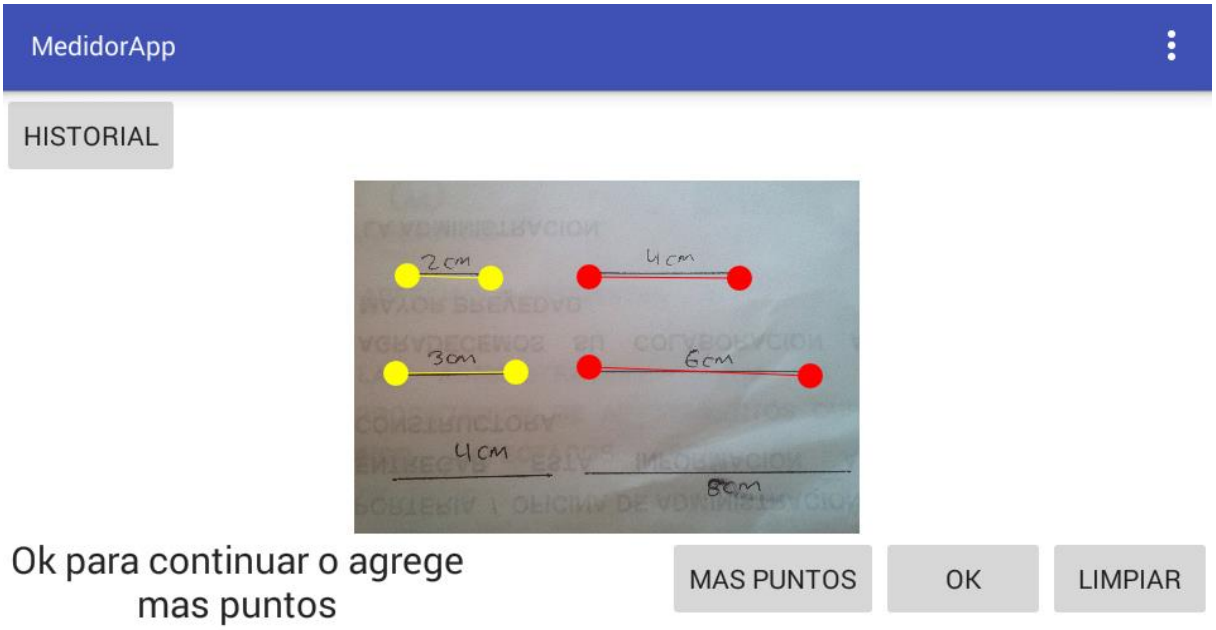

Figura 19 Nuevos puntos de selección

7. Se da click en el botón “ok” y se despliega un nuevo cuadro en el cual se puede observar una lista con la unidad de medidas y espacios para agregar la medida de referencia como se puede ver en la Figura 20.

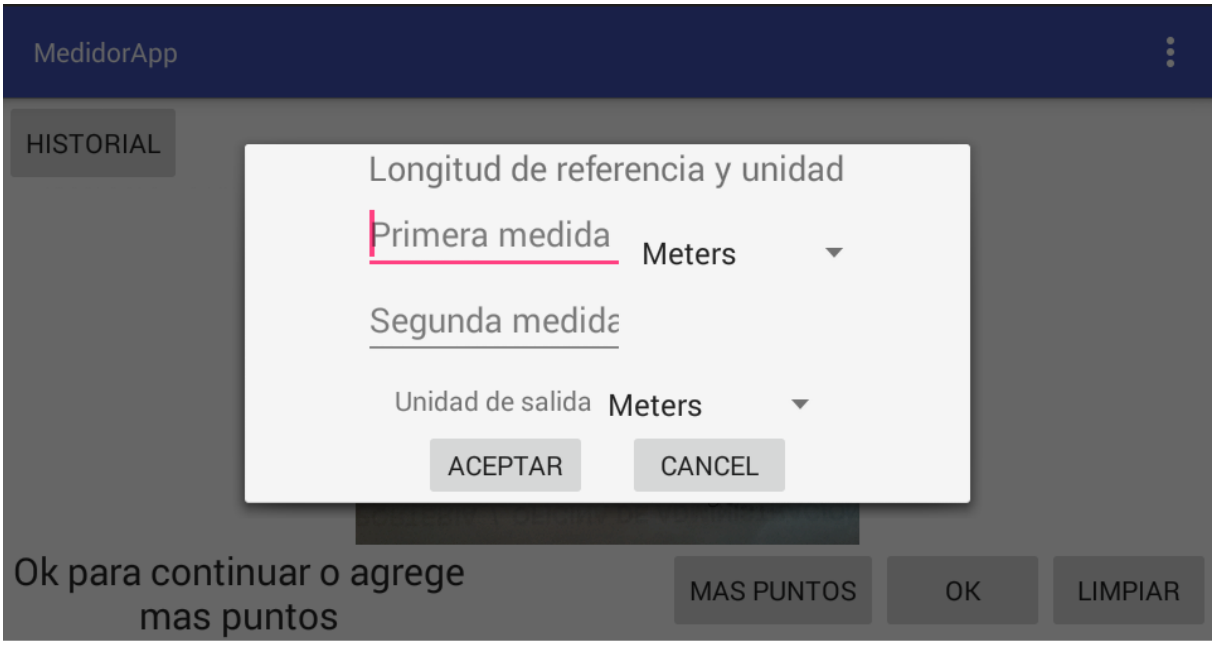

Figura 20 Medidas de referencia

8. Al dar click en la lista en la parte derecha se despliega para el usuario diferentes unidades métrica, de las cuales debe elegir una para los puntos de referencia, ver Figura 21.

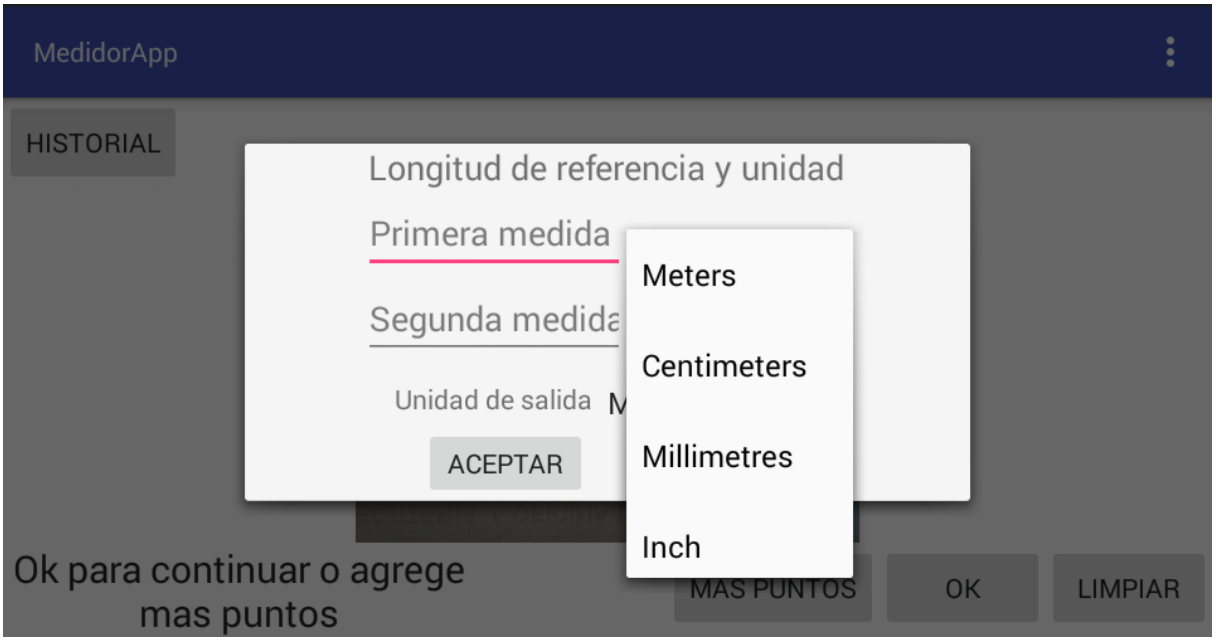

Figura 21 Unidades métricas del punto de referencia

9. En la parte inferior derecha se observa otra lista, la cual le da al usuario la capacidad de elegir la unidad de medida, para la longitud que se quiera hallar, ver Figura 22.

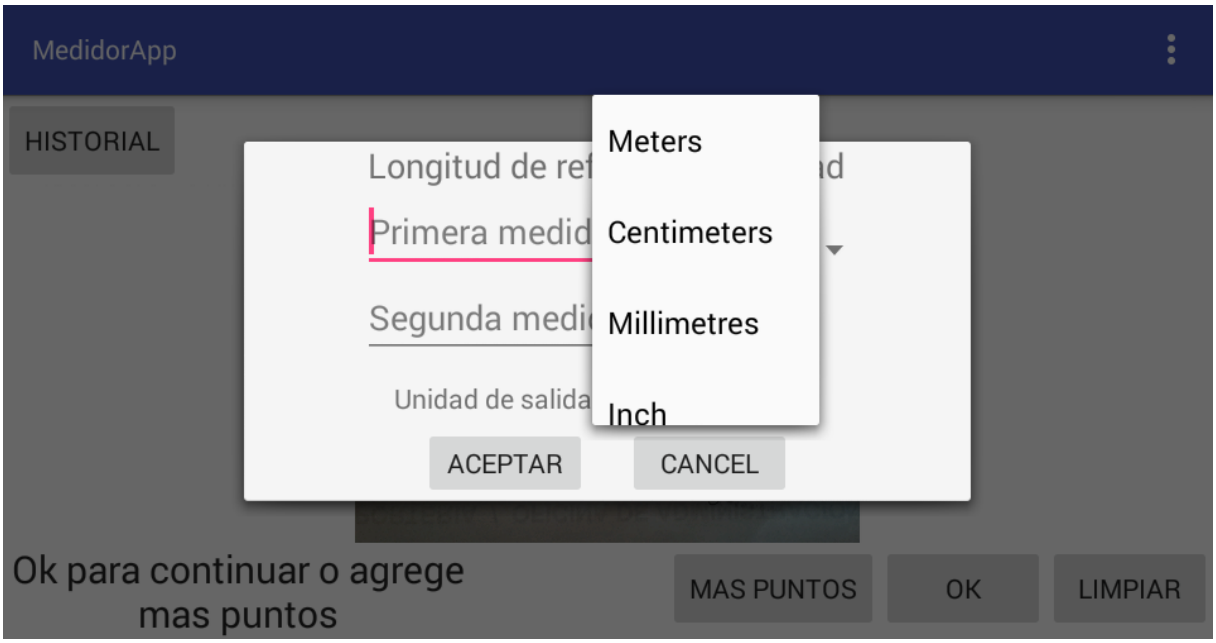

Figura 22 Unidades métricas del punto a medir

10. Una vez agregados los puntos de referencia y la unidad de medida, le damos un click al botón “aceptar”, ver Figura 23.

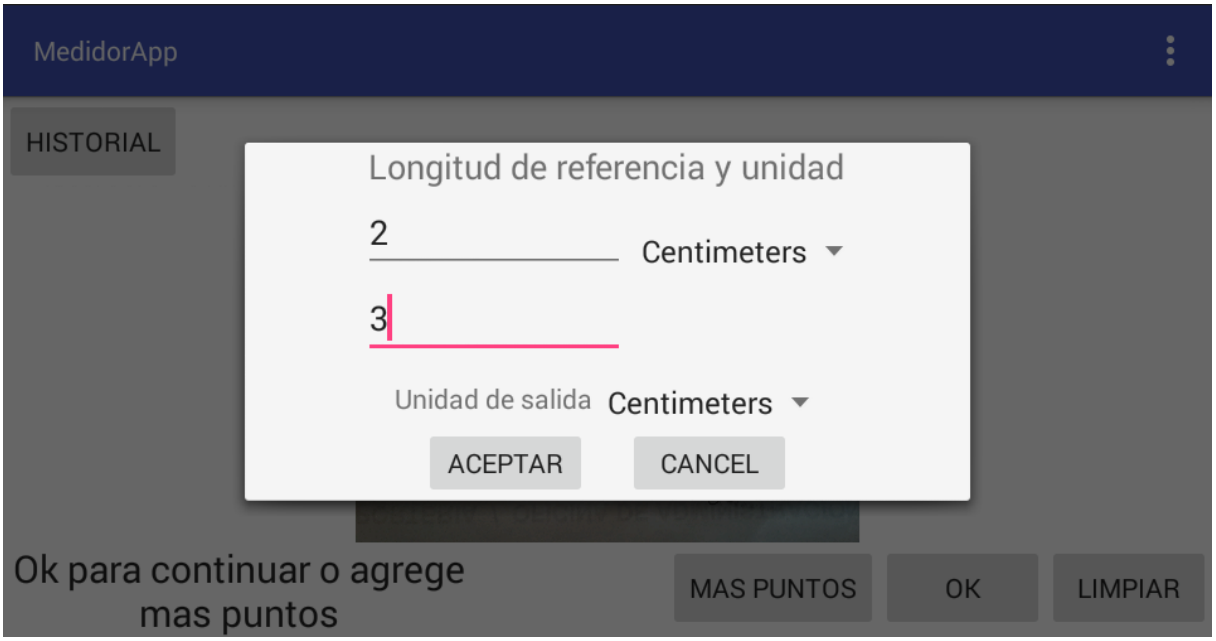

Figura 23 medidas y unidades métricas

11. Como se puede observar en la Figura 24, la aplicación despliega la medida de las líneas en color rojo, en el orden que fueron ingresadas.

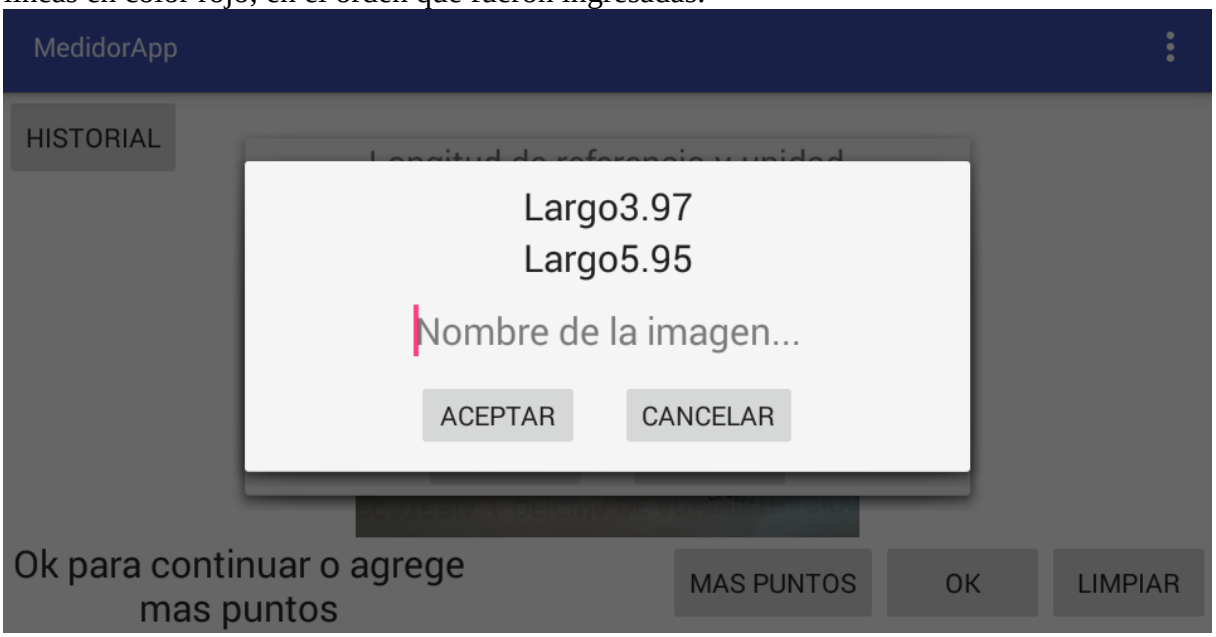

Figura 24 Resultados del aplicativo

12. Se debe agregar un nombre a la imagen, para poder reconocer las medidas obtenidas en un futuro, y archivarlo en la base de datos creada en el dispositivo por la aplicación. Ver Figura 25.

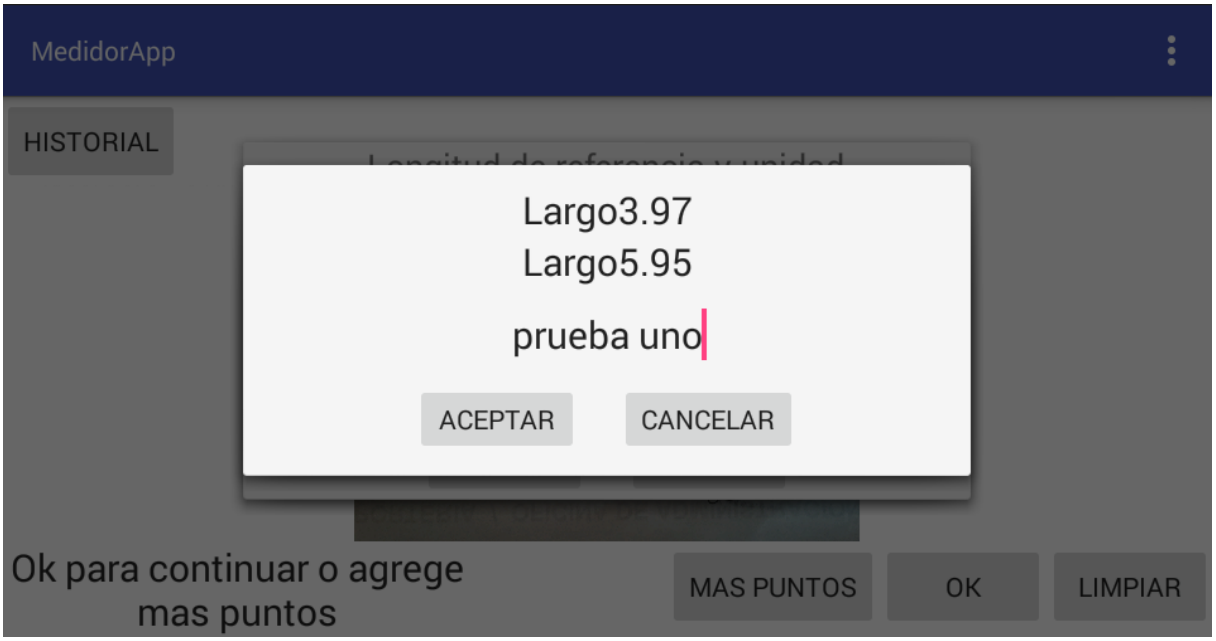

Figura 25 Almacenamiento de la imagen

13. Para revisar el historial de la aplicación, le damos click al botón “Historial” ubicado en la parte superior izquierda de la aplicación. Ver Figura 26.

Figura 26 Botón Historial del aplicativo

14. Como se observa en la Figura 27, se abre una nueva ventana en la cual se observa el nombre dado a la imagen, y los valores que se obtuvieron por la aplicación.

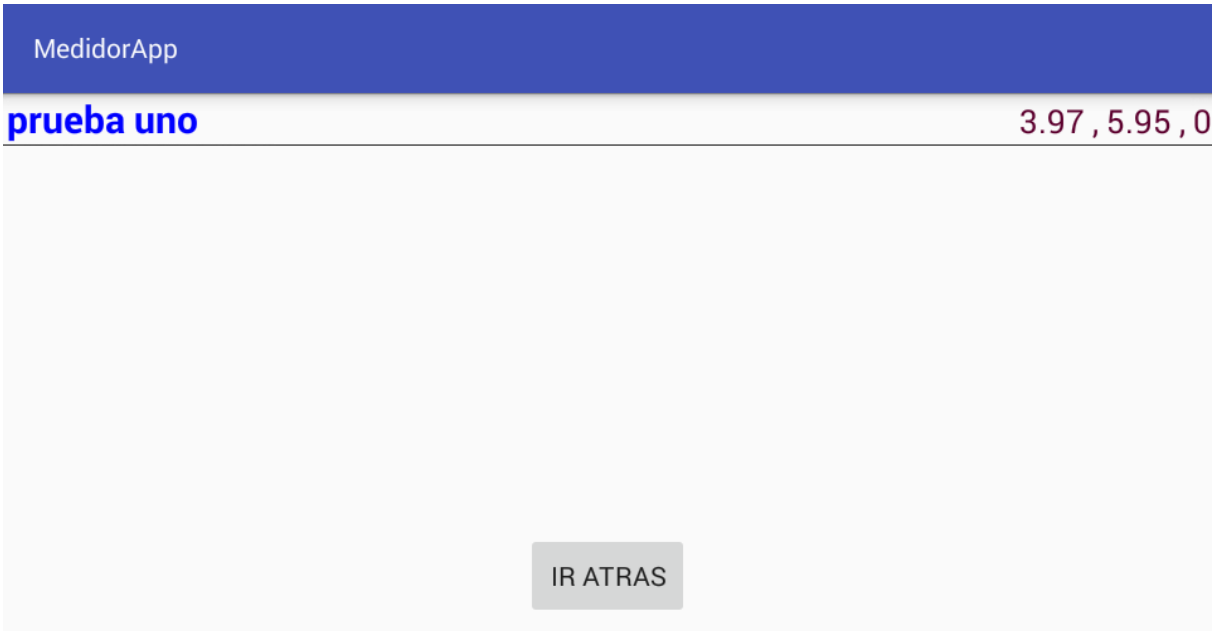

Figura 27 Vista lista de la base de datos interna del aplicativo

3.6 Resultados en ambientes controlados

Las pruebas realizadas se han hecho desde diferentes alturas, las cuales fueron de 5 Metros (Ver Figura 28), 10 Metros (Ver Figura 29), 15 Metros (Ver Figura 30), 20 Metros (Ver Figura 31) y 30 Metros (Ver Figura 32) con una calidad de imagen de 12 Megapíxeles, para ello se utilizó un dron phantom como el presentado en la Figura 6.

Como se puede observar en la Figura 33 Puntos a medir, las líneas de color rojo indican las mediciones que se llevaron a cabo, se optó por tener estas mediciones debido a que son los puntos de referencia primordiales de la prueba.

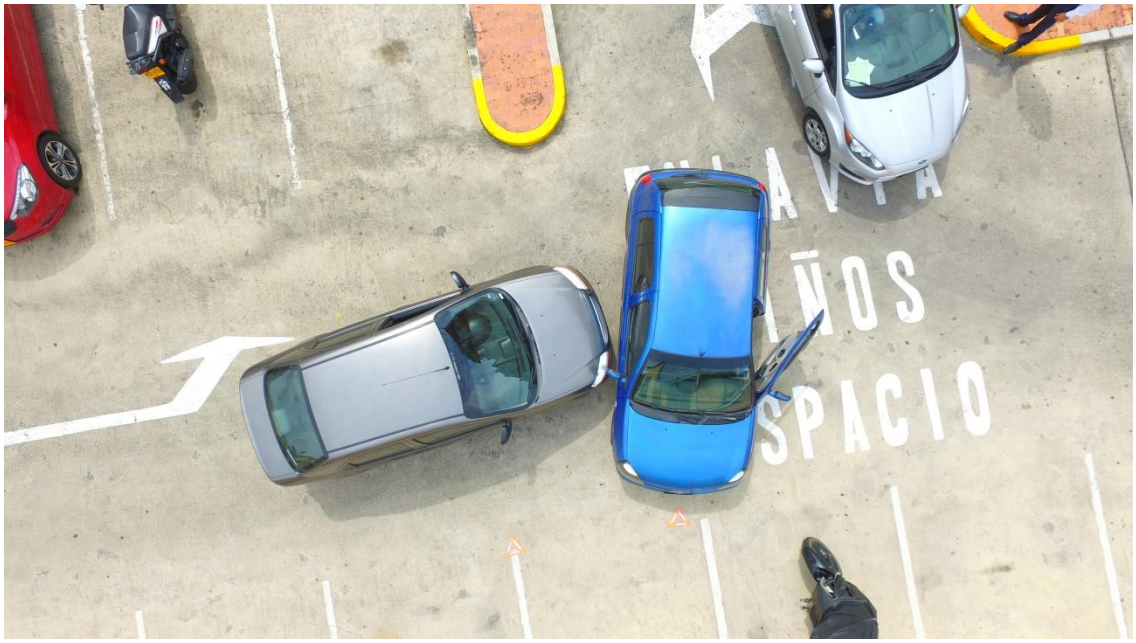

Figura 28 Prueba a 5 Metros de altura

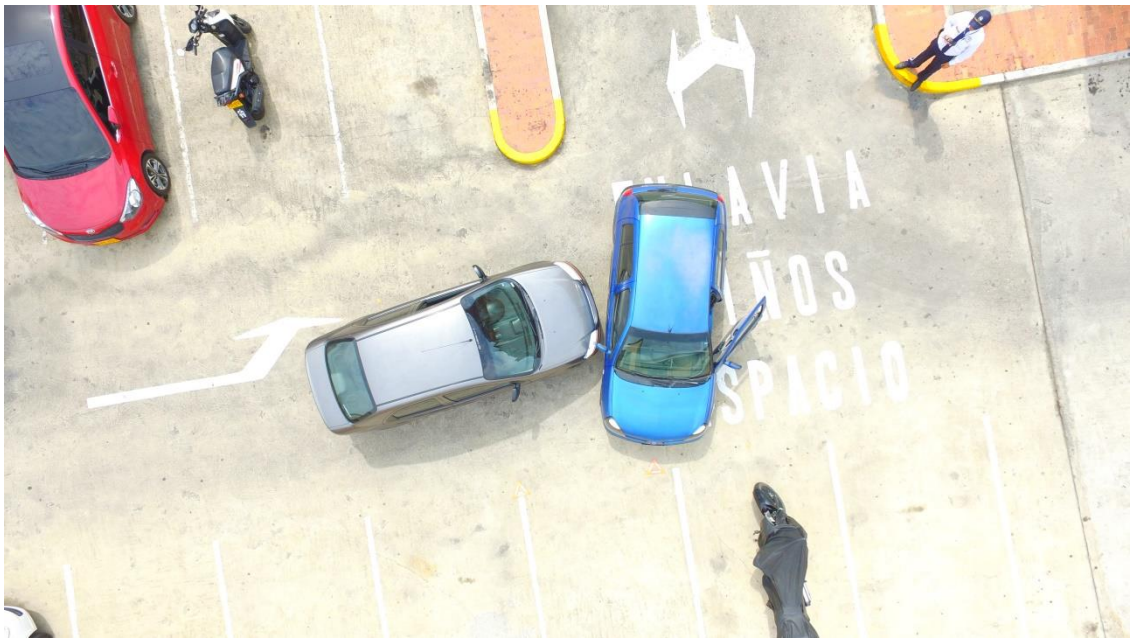

Figura 29 Prueba a 10 Metros de altura

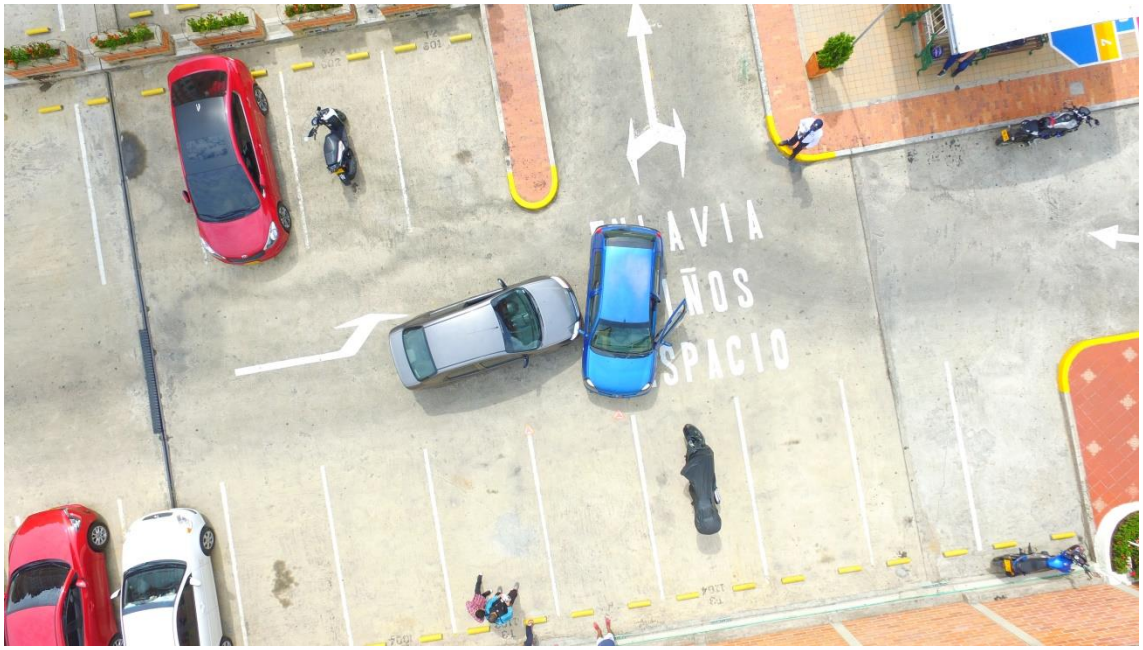

Figura 30 Prueba a 15 Metros de altura

Figura 31 Prueba a 20 Metros de altura

Figura 32 Prueba a 30 Metros de altura

Figura 33 Puntos a Medir

En la Tabla 6 se pueden observar los resultados de la aplicación en ambientes controlados, a cada prueba se le asignó un ID para identificarla y se ha calculado el porcentaje de error del dispositivo en cuanto a la medida real.

Tabla 6
Resultados en ambiente controlado

ID	Medida real (mts)	Medida App (mts)	Altura (mts)	Porcentaje de error
1	1	1	5	0%
2	3,5	3,3	5	6%
3	1,4	1,34	5	4%
4	1	0,95	10	5%
5	3,5	3,3	10	6%
6	1,4	1,3	10	8%
7	1	0,87	15	15%
8	3,5	3,1	15	13%
9	1,4	1,25	15	12%
10	1	1,15	20	13%
11	3,5	3,14	20	11%
12	1,4	1,25	20	12%
13	1	1,12	30	11%
14	3,5	3,9	30	10%
15	1,4	1,6	30	13%

Nota Resultados de los cálculos obtenidos por la aplicación contra las medidas reales con su respectivo porcentaje de error.

Como se puede observar en la tabla anterior, los porcentajes de error más bajos son en las menores alturas (5 y 10 mts), las cuales muestran errores que se encuentran en un rango desde el 0% al 8%, cabe destacar que en las mayores alturas (20 y 30 mts), los porcentajes de

error aun que pueden llegar a ser desde el 10% hasta el 15 %, la mayoría de las medidas no se alejan mucho de la realidad.

A pesar de la precisión del mismo, cabe destacar que el aplicativo se encuentra en una fase inicial, por lo tanto, tiene una carencia de confiabilidad baja, para mejorar dicho aspecto es recomendable como trabajo futuro poder almacenar los datos en la nube y crear una aplicación web del mismo, para poder brindar un mejor servicio y tener una aplicación mas confiable.

3.7 Conclusiones y recomendaciones

3.7.1 Conclusiones

- La revisión bibliográfica de la utilización de drones para aplicaciones de seguimiento de accidentes y desastres presenta un panorama cuyas principales características son: (1) La utilización de drones en accidentes y desastres es cada vez mayor, tanto en el uso de los mismos para accidentes de tránsito como desastres causados por problemas ambientales, (2) La evolución en los algoritmos de procesamiento de imágenes y seguimiento de objetos utilizados en los diferentes drones con el fin de poder llegar a tener una mayor fiabilidad de los mismos, (3) El uso de drones para el seguimiento de controles de calidad, con el fin de evitar accidentes a futuro.
- Al realizar el diseño de un prototipo software para móvil que permite el registro y procesamiento de imágenes capturadas por un dron, se puede destacar que existe una carencia con respecto a los proyectos de procesamiento de imágenes en este tipo de dispositivos, esto debido a que los mismos se encuentran enfocados en aplicaciones para la conectividad del usuario.

- En el momento de determinar las condiciones requeridas para el registro de la imagen aérea tomada por el dron, se puede destacar que: (1) Aunque existen drones los cuales pueden llegar a tener distancias de hasta 1 kilómetro de altura, solo se pueden utilizar un máximo de 10 metros para asegurar un buen desempeño de la aplicación, (2) el clima es un factor que afecta en gran medida al dron, debido a que los vientos fuertes en días lluviosos pueden llegar a desestabilizarlo y hacer que el usuario pierda el control del mismo, por eso es recomendable utilizarlo en días sin presencia de lluvia.
- Las pruebas del aplicativo en un ambiente controlado permiten evidenciar que: (1) La capacidad de precisión del aplicativo es bastante alta siempre y cuando las imágenes sean capturadas desde una altura optima y con una buena calidad de imagen, (2) El aplicativo está limitado al uso nocturno, ya que sin una buena fuente de luz resulta imposible poder encontrar los puntos a medir y los puntos de referencia.

3.7.2 Recomendaciones

3.7.2.1 Recomendaciones para la continuidad del proyecto.

- Poder calcular distintas medidas con un solo punto de referencia.
- Poder hacerle zoom a la imagen antes de seleccionar los puntos de referencia.
- Mover los puntos de referencia a gusto.
- Agregar más de 3 distancias a medir.
- Actualizar el aplicativo con las nuevas funciones de Android 7.
- Buscar la forma de guardar los puntos métricos en la imagen.
- Conectar el aplicativo con un API que permita almacenar en la nube la información del mismo.

- Ofrecer la posibilidad de consultar, ver medidas y ver las imágenes desde un portal web.
- Hallar los errores porcentuales en base a una altura determinada.

3.7.2.2 Recomendaciones con la universidad

- En las electivas como programación en dispositivos móviles, no comenzar desde un nivel tan básico, debido a que por el corto tiempo no se pueden tocar temas que son fundamentales.
- Recalco la importancia del idioma inglés, debido a que una gran parte de la documentación se encuentra en este idioma y a su vez, la mayoría de expertos en esta área son de Estados Unidos, Canadá o Francia.
- Acortar los tiempos de revisión de proyectos de grado, debido a que se pueden tardar hasta 1 mes en dar una respuesta.

3.8 Apéndices

En esta sección se encuentra el material complementario al informe, el cual consta de los códigos de las clases principales del proyecto.

Tabla 7
Código clase MainActivity

```
public class MainActivity extends ActionBarActivity implements
InputDialog.InputDialogListener {

   private DrawView drawView;
   private FrameLayout preview;
   private Button btn_ok, btn_addmore, btn_addmore2, btn_addmore3;
   private Button btn_cancel;
   private Button btn_takePicture, btn_loadPicture;
   private File photoFile;

   private double result, result1, result2;
   static int inc = 0;
   DatabaseHandler db;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_main);

      db = new DatabaseHandler(this);
      preview = (FrameLayout) findViewById(R.id.camera_preview);
      btn_takePicture = (Button) findViewById(R.id.button_takePicture);
      btn_loadPicture = (Button) findViewById(R.id.button_loadPicture);
      btn_ok = (Button) findViewById(R.id.button_calculate);
      btn_cancel = (Button) findViewById(R.id.button_cancel);
      btn_addmore = (Button) findViewById(R.id.morelines1);
      btn_addmore2 = (Button) findViewById(R.id.morelines2);
      btn_addmore3 = (Button) findViewById(R.id.morelines3);

      btn_addmore2.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {

            inc = 1;
            btn_addmore3.setVisibility(View.VISIBLE);
            btn_addmore2.setVisibility(View.GONE);


         }
      });

      btn_addmore3.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            inc = 2;
            btn_addmore3.setVisibility(View.GONE);
            btn_addmore2.setVisibility(View.VISIBLE);
```

```
        }
    });

    addPictureButton();
    btn_loadPicture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            dispatchChoosePhotoIntent();
        }
    });
    btn_takePicture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            dispatchTakePictureIntent();
        }
    });
    btn_addmore.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent(MainActivity.this, Results.class);
            startActivity(intent);
            finish();
        }
    });

    btn_ok.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(final View v) {
            if (DrawView.check == true) {

//                new InputDialog().show(getFragmentManager(), "input_dialog");
                LayoutInflater li = LayoutInflater.from(MainActivity.this);
                final View promptsView = li.inflate(R.layout.dialog, null);

                final AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                        MainActivity.this);

                // set prompts.xml to alertdialog builder
                alertDialogBuilder.setView(promptsView);
                alertDialogBuilder.setCancelable(false);

                final EditText first = (EditText) promptsView
                        .findViewById(R.id.reference_input);

                final EditText second = (EditText) promptsView
                        .findViewById(R.id.reference_input1);
                final EditText third = (EditText) promptsView
                        .findViewById(R.id.reference_input2);

                if (DrawView.increase ==0) {
                    Log.d("checkingthis","1");
                    first.setVisibility(View.VISIBLE);
                    second.setVisibility(View.GONE);
                    third.setVisibility(View.GONE);
                } else if (DrawView.increase ==1) {
                    Log.d("checkingthis","2");
                    first.setVisibility(View.VISIBLE);
                    second.setVisibility(View.VISIBLE);
                    third.setVisibility(View.GONE);
```

```
            } else if (DrawView.increase==2) {
                Log.d("checkingthis","3");
                first.setVisibility(View.VISIBLE);
                second.setVisibility(View.VISIBLE);
                third.setVisibility(View.VISIBLE);
            }
            Button mSubmit = (Button) promptsView.findViewById(R.id.submit);
            final Button mCancel = (Button) promptsView.findViewById(R.id.cancel);

            // create alert dialog
            final AlertDialog alertDialog = alertDialogBuilder.create();


            mSubmit.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

//                  try {


                    int inputUnit = ((Spinner)
promptsView.findViewById(R.id.input_unit_chooser)).getSelectedItemPosition();
                    int outputUnit = ((Spinner)
promptsView.findViewById(R.id.output_unit_chooser)).getSelectedItemPosition();
                    double reference = 0,reference1=0,reference2=0;
                    if (!first.getText().toString().isEmpty()) {
                        reference = Double.parseDouble(((EditText)
promptsView.findViewById(R.id.reference_input)).getText().toString());
                    }if (!second.getText().toString().isEmpty()) {
                        reference1 = Double.parseDouble(((EditText)
promptsView.findViewById(R.id.reference_input1)).getText().toString());
                    }if (!third.getText().toString().isEmpty()) {
                        reference2 = Double.parseDouble(((EditText)
promptsView.findViewById(R.id.reference_input2)).getText().toString());
                    }
                    if (reference !=0){
                        result = drawView.calculate(reference, inputUnit, outputUnit);}
                    if (reference1 !=0) {
                        result1 = drawView.calculate(reference1, inputUnit, outputUnit);
                    }if (reference2 !=0) {
                        result2 = drawView.calculate(reference2, inputUnit, outputUnit);
                    }
                    showResult();









                }
            });

            mCancel.setOnClickListener(new View.OnClickListener()

            {
                @Override
                public void onClick(View v) {
                    alertDialog.cancel();
                }
            });
```

```
                // show it
                alertDialog.show();


            } else

            {
                Toast.makeText(MainActivity.this, "Select all measurement points first",
Toast.LENGTH_LONG).show();
            }
        }
    });

    btn_cancel.setOnClickListener(new View.OnClickListener()

    {
        @Override
        public void onClick(View v) {
            DrawView.check = false;
            inc=0;
            drawView.clearCanvas();
        }
    });

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    switch (id) {
//        case R.id.action_settings:
//            break;
        case R.id.action_cleanStorage:
            cleanPhotoStorage();
            break;
        case R.id.action_choosePhoto:
            dispatchChoosePhotoIntent();
            break;
        case R.id.action_takePicture:
            dispatchTakePictureIntent();
            break;
    }
    return super.onOptionsItemSelected(item);
}

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

```
        // Ensure that there's a camera activity to handle the intent
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            Long tsLong = System.currentTimeMillis() / 1000;
            String ts = tsLong.toString();
            // Create the File where the photo should go
            photoFile = null;

            photoFile = new File(Environment.getExternalStorageDirectory() + "/screenshot" + ts + ".png");

            Bitmap bitmap = BitmapFactory.decodeFile(String.valueOf(photoFile));

    //necessary arguments and draw whatever you want. thes all are drawn on the bitmap.finally save this bitmap


            // Continue only if the File was successfully created
            if (photoFile != null) {
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
                        Uri.fromFile(photoFile));
                startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
            }
        }
    }


    private void addPictureButton() {
        preview.removeAllViews();

        btn_cancel.setVisibility(View.GONE);
        btn_ok.setVisibility(View.GONE);
        btn_addmore2.setVisibility(View.GONE);
        btn_addmore3.setVisibility(View.GONE);
        btn_takePicture.setVisibility(View.VISIBLE);
        btn_loadPicture.setVisibility(View.VISIBLE);
    }

    private void pictureTaken() {
        preview.removeAllViews();
        btn_cancel.setVisibility(View.VISIBLE);
        btn_ok.setVisibility(View.VISIBLE);
        btn_addmore2.setVisibility(View.VISIBLE);
        btn_addmore3.setVisibility(View.VISIBLE);
        btn_takePicture.setVisibility(View.GONE);
        btn_loadPicture.setVisibility(View.GONE);
        creatDialog();

        ImageSurface image = new ImageSurface(this, photoFile);
        preview.addView(image);
        ((TextView) findViewById(R.id.info_lbl)).setText(getResources().getString(R.string.setReferencePoints));
        drawView = new DrawView(this);
        preview.addView(drawView);
    }


    static final int REQUEST_IMAGE_CAPTURE = 1;
    static final int REQUEST_SELECT_PHOTO = 2;

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch (requestCode) {
            case REQUEST_IMAGE_CAPTURE:
```

```
            if (resultCode == RESULT_OK)
                pictureTaken();
            break;
        case REQUEST_SELECT_PHOTO:
            if (resultCode == RESULT_OK) {
                Uri uri = data.getData();
                String filePath = Utils.getPath(this, uri);
                photoFile = new File(filePath);
                pictureTaken();
            }
            break;
        default:
            super.onActivityResult(requestCode, resultCode, data);
            break;
    }

}

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.US).format(new Date());

    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getExternalFilesDir(null);

    return File.createTempFile(
            imageFileName,  /* prefix */
            ".jpg",         /* suffix */
            storageDir      /* directory */
    );
}

private void cleanPhotoStorage() {
    File storageDir = getExternalFilesDir(null);
    File fList[] = storageDir.listFiles();
    //Search for pictures in the directory
    for (int i = 0; i < fList.length; i++) {
        String pes = fList[i].getName();
        if (pes.endsWith(".jpg"))
            new File(fList[i].getAbsolutePath()).delete();
    }
    Toast.makeText(this, getResources().getString(R.string.storageDeleted),
Toast.LENGTH_SHORT).show();
}

private void dispatchChoosePhotoIntent() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent,
getResources().getString(R.string.action_choosePhoto)), REQUEST_SELECT_PHOTO);
}

@Override
public void onDialogPositiveClick(DialogFragment dialog) {
    int inputUnit = ((Spinner)
dialog.getDialog().findViewById(R.id.input_unit_chooser)).getSelectedItemPosition();
    int outputUnit = ((Spinner)
dialog.getDialog().findViewById(R.id.output_unit_chooser)).getSelectedItemPosition();
    try {
```

```
            double reference = Double.parseDouble(((EditText)
dialog.getDialog().findViewById(R.id.reference_input)).getText().toString());
            double reference1 = Double.parseDouble(((EditText)
dialog.getDialog().findViewById(R.id.reference_input1)).getText().toString());
            double reference2 = Double.parseDouble(((EditText)
dialog.getDialog().findViewById(R.id.reference_input2)).getText().toString());

            result = drawView.calculate(reference, inputUnit, outputUnit);

            result1 = drawView.calculate(reference1, inputUnit, outputUnit);

            result2 = drawView.calculate(reference2, inputUnit, outputUnit);

            showResult();

        } catch (NumberFormatException ex) {
            Toast.makeText(this, getResources().getString(R.string.error_numberFormat),
Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onDialogNegativeClick(DialogFragment dialog) {
        //Do absolutely nothing
    }

    private void showResult() {
        if (result != -1) {
            final DecimalFormat decimalFormat = new DecimalFormat("#.##");
            LayoutInflater li = LayoutInflater.from(MainActivity.this);
            View promptsView = li.inflate(R.layout.prompts, null);

            final AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                    this);

            alertDialogBuilder.setView(promptsView);
            alertDialogBuilder.setCancelable(false);

            final TextView user = (TextView) promptsView
                    .findViewById(R.id.textView1);

            final TextView user1 = (TextView) promptsView
                    .findViewById(R.id.textView2);
            final TextView user2 = (TextView) promptsView
                    .findViewById(R.id.textView3);

            if (DrawView.increase ==0) {
                Log.d("checkingthis","1");
                user.setVisibility(View.VISIBLE);
                user1.setVisibility(View.GONE);
                user2.setVisibility(View.GONE);
            } else if (DrawView.increase ==1) {
                Log.d("checkingthis","2");
                user.setVisibility(View.VISIBLE);
                user1.setVisibility(View.VISIBLE);
                user2.setVisibility(View.GONE);
            } else if (DrawView.increase==2) {
                Log.d("checkingthis","3");
                user.setVisibility(View.VISIBLE);
```

```
            user1.setVisibility(View.VISIBLE);
            user2.setVisibility(View.VISIBLE);
        }

        user.setText(getResources().getString(R.string.result_lbl) + decimalFormat.format(result));
        user1.setText(getResources().getString(R.string.result_lbl) + decimalFormat.format(result1));
        user2.setText(getResources().getString(R.string.result_lbl) + decimalFormat.format(result2));

        final EditText userInput = (EditText) promptsView.findViewById(R.id.editTextDialogUserInput);
        Button mSubmit = (Button) promptsView.findViewById(R.id.submit);
        Button mCancel = (Button) promptsView.findViewById(R.id.cancel);

        // create alert dialog
        final AlertDialog alertDialog = alertDialogBuilder.create();


        mSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (userInput.getText().toString().isEmpty()) {

                    Toast.makeText(MainActivity.this, "Enter image hint...", Toast.LENGTH_LONG).show();

                } else {

                    db.addContact(new Values(userInput.getText().toString(), decimalFormat.format(result),
decimalFormat.format(result1), decimalFormat.format(result2)));
                    alertDialog.cancel();
                }

            }
        });

        mCancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                alertDialog.cancel();
            }
        });
        // show it
        alertDialog.show();

    }

}

public void creatDialog(){
    LayoutInflater li = LayoutInflater.from(MainActivity.this);
    final View helpUser = li.inflate(R.layout.help_user, null);
    final AlertDialog.Builder builder = new AlertDialog.Builder(
            MainActivity.this);
    builder.setView(helpUser)
            .setCancelable(false)
            .setTitle("Mensaje de ayuda al usuario");
    final AlertDialog dialog = builder.create();
    final TextView helpTxt = (TextView) helpUser.findViewById(R.id.txtHelpUser);
    helpTxt.setText("Primero se agregan los puntos de referencia " +
            "los cuales se indican en color amarillo, " +
            "seguidos por los puntos del objeto a medir, " +
            "los cuales son de color rojo");
```

```
        final Button helpBtn = (Button) helpUser.findViewById(R.id.btnHelpUser);
        helpBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dialog.cancel();
            }
        });
        dialog.show();
    }

}
```

Nota Codigo del MainActivity.

Tabla 8

Codigo clase ImageSurface

```
public class ImageSurface extends SurfaceView implements SurfaceHolder.Callback {

    private Bitmap icon;
    private Paint paint;

    public ImageSurface(Context context, File image) {
        super(context);
        getHolder().addCallback(this);
        String imageFile = image.getAbsolutePath();
        icon = BitmapFactory.decodeFile(imageFile);
        paint = new Paint();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int newWidth = canvas.getHeight() * icon.getWidth() / icon.getHeight();
        icon = Bitmap.createScaledBitmap(icon, newWidth, canvas.getHeight(), false);
        canvas.drawColor(Color.WHITE);
        int cx = (canvas.getWidth() - icon.getWidth()) / 2;
        int cy = (canvas.getHeight() - icon.getHeight()) / 2;
        canvas.drawBitmap(icon, cx, cy, paint);
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    }

    @SuppressLint("WrongCall")
    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        Canvas canvas = null;
        try {
            canvas = holder.lockCanvas(null);
            synchronized (holder) {
                onDraw(canvas);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (canvas != null) {
                holder.unlockCanvasAndPost(canvas);
```

```
        }
      }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
    }
}
```

Nota Codigo del ImageSurface

Tabla 9

Codigo clase DrawView

```
public class DrawView extends SurfaceView {
    private Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
    List<Point> circlePoints;

    private Context context;
    public static Boolean check = false;
    public static Boolean c1 = false;
    public static Boolean c2 = false;
    public static Boolean c3 = false;

    public static int increase=0;

    private static int REFERENCE_POINT_COLOR = Color.YELLOW;
    private static int MEASURE_POINT_COLOR = Color.RED;

    public DrawView(Context context) {
        super(context);
        this.context = context;
        paint.setColor(Color.RED);
        paint.setStyle(Paint.Style.FILL);
        circlePoints = new ArrayList<>();
        setWillNotDraw(false);
        getHolder().setFormat(PixelFormat.TRANSLUCENT);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        int size = circlePoints.size();
        for (int i = 0; i < size; i++) {
            //Set color based on order. First 2 points are the reference points.
            if (i < 2) paint.setColor(REFERENCE_POINT_COLOR);
            else if (i < 4) {
                paint.setColor(MEASURE_POINT_COLOR);
            } else if (i > 3 && i < 6) {
                paint.setColor(REFERENCE_POINT_COLOR);
            } else if (i > 5 && i < 8)
                paint.setColor(MEASURE_POINT_COLOR);

            else if (i > 7 && i < 10) {
                paint.setColor(REFERENCE_POINT_COLOR);
            } else if (i > 9 && i < 12)
                paint.setColor(MEASURE_POINT_COLOR);


            Point p = circlePoints.get(i);
```

```
            canvas.drawCircle(p.x, p.y, 10, paint);
            if (i == 1) {
                canvas.drawLine(circlePoints.get(0).x, circlePoints.get(0).y, circlePoints.get(1).x,
circlePoints.get(1).y, paint);
            }
            if (i == 3) {
                canvas.drawLine(circlePoints.get(2).x, circlePoints.get(2).y, circlePoints.get(3).x,
circlePoints.get(3).y, paint);
            }

            if (i == 5) {
                canvas.drawLine(circlePoints.get(4).x, circlePoints.get(4).y, circlePoints.get(5).x,
circlePoints.get(5).y, paint);

            }

            if (i == 7) {
                canvas.drawLine(circlePoints.get(6).x, circlePoints.get(6).y, circlePoints.get(7).x,
circlePoints.get(7).y, paint);
            }

            if (i == 9) {
                canvas.drawLine(circlePoints.get(8).x, circlePoints.get(8).y, circlePoints.get(9).x,
circlePoints.get(9).y, paint);
            }
            if (i == 11) {
                canvas.drawLine(circlePoints.get(10).x, circlePoints.get(10).y, circlePoints.get(11).x,
circlePoints.get(11).y, paint);
            }
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            if (MainActivity.inc == 0) {
                if (circlePoints.size() < 4) {
                    circlePoints.add(new Point(Math.round(event.getX()), Math.round(event.getY())));
                    invalidate();
                    if (circlePoints.size() == 2)
                        check = false;
                    if (circlePoints.size() == 4) {
                        check = true;
                        c1 = true;
                        c2 = false;
                        c3 = false;
                        increase=0;
                        ((TextView) ((Activity)
context).findViewById(R.id.info_lbl)).setText(getResources().getString(R.string.setScaleValue));
                    }
                }
            } else if (MainActivity.inc == 1) {
                if (circlePoints.size() < 8) {
                    circlePoints.add(new Point(Math.round(event.getX()), Math.round(event.getY())));
                    invalidate();
                    if (circlePoints.size() == 2)

                        if (circlePoints.size() == 4) {

                        }
```

```
                if (circlePoints.size() == 6) {
                    check = false;
                }
                if (circlePoints.size() == 8) {
                    check = true;
                    c1 = false;
                    c2 = true;
                    c3 = false;
                    increase=1;
                    ((TextView) ((Activity)
context).findViewById(R.id.info_lbl)).setText(getResources().getString(R.string.setScaleValue));

                }
            }
        } else if (MainActivity.inc == 2) {
            if (circlePoints.size() < 12) {
                circlePoints.add(new Point(Math.round(event.getX()), Math.round(event.getY())));
                invalidate();
                if (circlePoints.size() == 2)

                    if (circlePoints.size() == 4) {

                    }
                if (circlePoints.size() == 6) {

                }
                if (circlePoints.size() == 8) {

                }

                if (circlePoints.size() == 10) {
                    check = false;
                }
                if (circlePoints.size() == 12) {
                    check = true;
                    c1 = false;
                    c2 = false;
increase=2;
                    c3 = true;
                    ((TextView) ((Activity)
context).findViewById(R.id.info_lbl)).setText(getResources().getString(R.string.setScaleValue));

                }
            }
        }
    }

    return false;
}
public void clearCanvas() {
    circlePoints.clear();
    ((TextView) ((Activity)
context).findViewById(R.id.info_lbl)).setText(getResources().getString(R.string.setPicture));
    invalidate();
}

public double calculate(double reference, int inputUnitIndex, int outputUnitIndex) {

    if (circlePoints.size() == 4) {
//        c1 = true;
```

```
//          c2 = false;
            c3 = false;
            return Ruler.compute(circlePoints, reference, inputUnitIndex, outputUnitIndex);

        } else if (circlePoints.size() == 6) {
//          c2 = true;
//          c1 = false;
//          c3 = false;
            return Ruler.compute(circlePoints, reference, inputUnitIndex, outputUnitIndex);

        } else if (circlePoints.size() == 8) {
//          c3 = true;
//          c2 = false;
//          c1 = false;
            return Ruler.compute(circlePoints, reference, inputUnitIndex, outputUnitIndex);

        } else if (circlePoints.size() == 10) {
//          c3 = true;
//          c2 = false;
//          c1 = false;
            return Ruler.compute(circlePoints, reference, inputUnitIndex, outputUnitIndex);

        } else if (circlePoints.size() == 12) {
//          c3 = true;
//          c2 = false;
//          c1 = false;
            return Ruler.compute(circlePoints, reference, inputUnitIndex, outputUnitIndex);

        }

        Toast.makeText(context, getResources().getString(R.string.error_noPoints),
Toast.LENGTH_SHORT).show();
        return -1;
    }
}
```

Nota Codigo de la clase *DrawView*

Tabla 10

Codigo Clase Ruler

```
public class Ruler {

    private Ruler(){}

    public static double compute(List<Point> points, double scale, int inputUnitIndex, int outputUnitIndex){
        if(points.size() < 4) return -1;

        //Get reference points
        Point ref1 = points.get(0);
        Point ref2 = points.get(1);
        //Get the measurement points
        Point m1 = points.get(2);
        Point m2 = points.get(3);

        double reference = getDistance(ref1, ref2);
        double measurement = getDistance(m1, m2);

        measurement = (measurement * scale) / reference; //Get the actual distance
        //Convert to the right unit
        measurement = convertUnits(inputUnitIndex, reference, outputUnitIndex, measurement);
```

```
        return measurement;
    }

    private static double getDistance(Point p1, Point p2){
        double x = Math.pow(p2.x - p1.x, 2);
        double y = Math.pow(p2.y - p1.y, 2);
        return Math.sqrt(x+y);
    }

    private static double convertUnits(int refUnit, double reference, int meaUnit, double measurement){
        if(refUnit == meaUnit)
            return measurement;

        measurement = toMeters(measurement, refUnit);
        switch (meaUnit){
            case 0:
                return measurement;
            case 1:
                return Utils.metersToCentimeters(measurement);
            case 2:
                return Utils.metersToMillimeters(measurement);
            case 3:
                return Utils.metersToInch(measurement);
            case 4:
                return Utils.metersToFeet(measurement);
            case 5:
                return Utils.metersToYards(measurement);
            default:
                return -1;
        }
    }

    private static double toMeters(double measurement, int refUnit){
        switch (refUnit){
            case 0:
                return measurement;
            case 1:
                return Utils.centimetersToMeters(measurement);
            case 2:
                return Utils.millimetersToMeters(measurement);
            case 3:
                return Utils.inchesToMeters(measurement);
            case 4:
                return Utils.yardsToMeters(measurement);
            default:
                return -1;
        }
    }
}
```

Nota Codigo de la *Clase Ruler*

Tabla 11

Codigo clase Utils

```
public class Utils {
    private Utils(){}

    @TargetApi(19)
```

```
public static String getPath(final Context context, final Uri uri) {

    final boolean isKitKat = Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT;

    // DocumentProvider

    if (isKitKat && DocumentsContract.isDocumentUri(context, uri)) {
        // ExternalStorageProvider
        if (isExternalStorageDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            if ("primary".equalsIgnoreCase(type)) {
                return Environment.getExternalStorageDirectory() + "/" + split[1];
            }

            // TODO handle non-primary volumes
        }
        // DownloadsProvider
        else if (isDownloadsDocument(uri)) {

            final String id = DocumentsContract.getDocumentId(uri);
            final Uri contentUri = ContentUris.withAppendedId(
                    Uri.parse("content://downloads/public_downloads"), Long.valueOf(id));

            return getDataColumn(context, contentUri, null, null);
        }
        // MediaProvider
        else if (isMediaDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            Uri contentUri = null;
            switch (type) {
                case "image":
                    contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
                    break;
                case "video":
                    contentUri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
                    break;
                case "audio":
                    contentUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
                    break;
            }

            final String selection = "_id=?";
            final String[] selectionArgs = new String[] {
                    split[1]
            };

            return getDataColumn(context, contentUri, selection, selectionArgs);
        }
    }
    // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {
        return getDataColumn(context, uri, null, null);
    }
    // File
```

```
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}

public static String getDataColumn(Context context, Uri uri, String selection,
                                   String[] selectionArgs) {

    Cursor cursor = null;
    final String column = "_data";
    final String[] projection = {
            column
    };

    try {
        cursor = context.getContentResolver().query(uri, projection, selection, selectionArgs,
                null);
        if (cursor != null && cursor.moveToFirst()) {
            final int column_index = cursor.getColumnIndexOrThrow(column);
            return cursor.getString(column_index);
        }
    } finally {
        if (cursor != null)
            cursor.close();
    }
    return null;
}

public static boolean isExternalStorageDocument(Uri uri) {
    return "com.android.externalstorage.documents".equals(uri.getAuthority());
}

public static boolean isDownloadsDocument(Uri uri) {
    return "com.android.providers.downloads.documents".equals(uri.getAuthority());
}

public static boolean isMediaDocument(Uri uri) {
    return "com.android.providers.media.documents".equals(uri.getAuthority());
}

public static double inchesToMeters(double measurement) {
    return measurement * 0.0254;
}

public static double yardsToMeters(double measurement) {
    return measurement * 0.9144;
}

public static double millimetersToMeters(double measurement) {
    return measurement * 0.001;
}

public static double centimetersToMeters(double measurement) {
    return measurement * 0.01;
}

public static double metersToYards(double measurement) {
    return measurement * 1.0936133;
```

```
    }

    public static double metersToFeet(double measurement) {
        return measurement * 3.2808399;
    }

    public static double metersToInch(double measurement) {
        return measurement * 39.3700787;
    }

    public static double metersToMillimeters(double measurement) {
        return measurement * 1000;
    }

    public static double metersToCentimeters(double measurement) {
        return measurement * 100;
    }
}
```

Nota Codigo complete de la Codigo clase Utils

Tabla 12
Codigo clase InputDialog

```
public class InputDialog extends DialogFragment{
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState){
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        LayoutInflater inflater = getActivity().getLayoutInflater();
        //Inflate layout
        builder.setView(inflater.inflate(R.layout.dialog, null));
        //Set buttons
        builder.setPositiveButton(R.string.btn_OK, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                listener.onDialogPositiveClick(InputDialog.this);
            }
        });
        builder.setNegativeButton(R.string.btn_cancel, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                listener.onDialogNegativeClick(InputDialog.this);
            }
        });
        return builder.create();
    }
    public interface InputDialogListener{
        public void onDialogPositiveClick(DialogFragment dialog);
        public void onDialogNegativeClick(DialogFragment dialog);
    }

    InputDialogListener listener;

    @Override
    public void onAttach(Activity activity){
        super.onAttach(activity);
        try{
            listener = (InputDialogListener) activity;
        }catch(ClassCastException ex){
```

```
            throw new ClassCastException(activity.toString()
                    + " must implement InputDialogListener");
        }
    }
}
```

Nota Codigo complete de la Codigo clase InputDialog

Tabla 13
Codigo clase DatabaseHandler

```
public class DatabaseHandler extends SQLiteOpenHelper {
  private static final int DATABASE_VERSION = 2;
  private static final String DATABASE_NAME = "contactsManager";
   private static final String TABLE_CONTACTS = "contacts";
    private static final String KEY_ID = "id";
   private static final String KEY_NAME = "name";
   private static final String KEY_PH_NO = "phone_number";
   private static final String KEY_PH_NO1 = "phone_number1";
   private static final String KEY_PH_NO2 = "phone_number2";

   public DatabaseHandler(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
      //3rd argument to be passed is CursorFactory instance
   }

   // Creating Tables
   @Override
   public void onCreate(SQLiteDatabase db) {
      String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
            + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
            + KEY_PH_NO + " TEXT,"  + KEY_PH_NO1 + " TEXT," + KEY_PH_NO2 + " TEXT"+ ")";
      db.execSQL(CREATE_CONTACTS_TABLE);
   }

   // Upgrading database
   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      // Drop older table if existed
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);

      // Create tables again
      onCreate(db);
   }

    // code to add the new contact
    void addContact(Values contact) {
      SQLiteDatabase db = this.getWritableDatabase();

      ContentValues values = new ContentValues();
      values.put(KEY_NAME, contact.get_name()); // Contact Name
      values.put(KEY_PH_NO, contact.get_values());
       values.put(KEY_PH_NO1, contact.get_values1());
       values.put(KEY_PH_NO2, contact.get_values2());// Contact Phone
      // Inserting Row
      db.insert(TABLE_CONTACTS, null, values);
      //2nd argument is String containing nullColumnHack
      db.close(); // Closing database connection
   }
```

```
// code to get the single contact
Values getContact(int id) {
   SQLiteDatabase db = this.getReadableDatabase();

   Cursor cursor = db.query(TABLE_CONTACTS, new String[] { KEY_ID,
         KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
         new String[] { String.valueOf(id) }, null, null, null, null);
   if (cursor != null)
      cursor.moveToFirst();

   Values contact = new Values(Integer.parseInt(cursor.getString(0)),
         cursor.getString(1), cursor.getString(2), cursor.getString(3), cursor.getString(4));
   // return contact
   return contact;
}

// code to get all contacts in a list view
public List<Values> getAllContacts() {
   List<Values> contactList = new ArrayList<Values>();
   // Select All Query
   String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

   SQLiteDatabase db = this.getWritableDatabase();
   Cursor cursor = db.rawQuery(selectQuery, null);
   if (cursor.moveToFirst()) {
      do {
         Values contact = new Values();
         contact.set_id(Integer.parseInt(cursor.getString(0)));
         contact.set_name(cursor.getString(1));
         contact.set_values(cursor.getString(2));
         contact.set_values1(cursor.getString(3));
         contact.set_values2(cursor.getString(4));
         // Adding contact to list
         contactList.add(contact);
      } while (cursor.moveToNext());
   }

   // return contact list
   return contactList;
}

// code to update the single contact
public int updateContact(Values contact) {
   SQLiteDatabase db = this.getWritableDatabase();

   ContentValues values = new ContentValues();
   values.put(KEY_NAME, contact.get_name());
   values.put(KEY_PH_NO, contact.get_values());

   // updating row
   return db.update(TABLE_CONTACTS, values, KEY_ID + " = ?",
         new String[] { String.valueOf(contact.get_id()) });
}

// Deleting single contact
public void deleteContact(Values contact) {
   SQLiteDatabase db = this.getWritableDatabase();
   db.delete(TABLE_CONTACTS, KEY_ID + " = ?",
         new String[] { String.valueOf(contact.get_id()) });
```

```
        db.close();
    }

    // Getting contacts Count
    public int getContactsCount() {
        String countQuery = "SELECT  * FROM " + TABLE_CONTACTS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();

        // return count
        return cursor.getCount();
    }
}
```

Nota Codigo complete de la clase DatabaseHandler

Tabla 14

Codigo clase Results

```
public class Results extends AppCompatActivity {
    ListView listView;
    TestAdapter smsListAdapter;
   Button back;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_results);
        listView = (ListView) findViewById(R.id.list);
        smsListAdapter = new TestAdapter();
        listView.setAdapter(smsListAdapter);
        back=(Button)findViewById(R.id.back);
        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent(Results.this,MainActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }


    class TestAdapter extends BaseAdapter {


        DatabaseHandler db = new DatabaseHandler(Results.this);
        List<Values> contacts = db.getAllContacts();
        List<String> name;
        List<String> values;

        public int getCount() {
            return contacts.size();
        }

        public Object getItem(int arg0) {
            return arg0;
        }
```

```
    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View arg1, ViewGroup viewGroup) {
        View row = arg1;
//
        if (row == null) {
            LayoutInflater l = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
            row = l.inflate(R.layout.listview_each_item, viewGroup, false);

            TextView textView1 = (TextView) row.findViewById(R.id.tv1);
            TextView textView2 = (TextView) row.findViewById(R.id.tv2);
            name = new ArrayList<>();
            values = new ArrayList<>();

            // Set the Sender number and smsBody to respective TextViews

//          contacts = db.getAllContacts();


            for (Values cn : contacts) {
                String log = "Id: " + cn.get_id() + " ,Name: " + cn.get_name() + " ,Value: " +
                        cn.get_values();
                // Writing Contacts to log
                name.add(cn.get_name());
                values.add(cn.get_values()+" , "+cn.get_values1()+" , "+cn.get_values2());
                Log.d("Name: ", log);

            }
            textView1.setText(name.get(position));
            textView2.setText(values.get(position));

        }
        return row;
    }
  }
}
```

Nota Codigo complete de la *clase Results*

Tabla 15

Codigo clase Values

```
public class Values {
    int _id;
    String _name;
    String _values;
    String _values1;
    String _values2;

    public int get_id() {
        return _id;
    }

    public void set_id(int _id) {
        this._id = _id;
    }

    public String get_name() {
        return _name;
    }

    public void set_name(String _name) {
        this._name = _name;
    }

    public String get_values() {
        return _values;
    }

    public void set_values(String _values) {
        this._values = _values;
    }

    public Values() {
    }

    public String get_values1() {
        return _values1;
    }

    public void set_values1(String _values1) {
        this._values1 = _values1;
    }

    public String get_values2() {
        return _values2;
    }

    public void set_values2(String _values2) {
        this._values2 = _values2;
    }

    public Values(int id, String name, String values, String values1, String values2) {
        this._id = id;
        this._name = name;
```

```
        this._values = values;
        this._values1 = values1;
        this._values2 = values2;
    }

    public Values(String name, String values, String values1, String values2) {

        this._name = name;
        this._values = values;
        this._values1 = values1;
        this._values2 = values2;
    }
}
```

Nota Codigo completo de la *clase Values*

4 Bibliografía

[1] S. Palomino, "En Colombia muere más gente por accidentes viales que por el conflicto," *El Pais,* 23 Diciembre 2015.

[2] EL Tiempo, "El Tiempo," 22 Enero 2015. [Online].

[3] WORDPRESS, "WORDPRESS," 11 NOVIEMBRE 2014. [Online]. Available: https://ticyseguridadvial.wordpress.com/2014/11/11/las-tic-y-el-analisis-de-accidentes-de-transito/.

[4] MINISTERIO DE TRANSPORTE, "MINISTERIO DE TRANSPORTE," 6 Diciembre 2012. [Online]. Available: http://web.mintransporte.gov.co/rnat/app/ayudas/Resolucion_0011268_2012.pdf.

[5] Newsonjapan, "Japan Real," 19 mayo 2015. [Online]. Available: http://www.japanrealm.com/japanese-drones-examine/.

[6] EL ESPECTADOR, "El ESPECTADOR," 2 MARZO 2015. [Online]. Available: http://www.elespectador.com/tecnologia/drones-al-servicio-de-investigacion-de-accidentes-ecuad-articulo-547055.

[7] CESVI, "El Tiempo," Martes Enero 2011. [Online]. Available: http://www.eltiempo.com/archivo/documento/CMS-8789352.

[8] Prosic, 2010. [Online]. Available: http://www.prosic.ucr.ac.cr/sites/default/files/documentos/capitulo_06_4.pdf.

[9] Instituto Superior de Formacion y Recuersos en Red para el Profesorado, "ITE," 2008. [Online]. Available: http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0103.html.

[10] WHMCompleteSolution, "gospelidea," 2015. [Online]. Available: http://secure.gospelidea.com/clients/knowledgebase/147/-iComo-funciona-el-Streaming.html.

[11] google, "Android," 2015. [Online].

[c. torres, "androidsis," 19 junio 2014. [Online]. Available:

12] http://www.androidsis.com/la-verdadera-historia-de-android-llega-android-4-2-jelly-bean-2012/.

[13] J. Martinez, "Andro4all," 28 enero 2014. [Online].

[14] Beatriz, "uptodown," [Online]. Available: http:android-studio.uptodown.com/.

[15] Android Development, "Android Development," 2010. [Online]. Available: http://developer.android.com/tools/studio/index.html.

[16] aertec solutions, "aertec solutions," 27 enero 2015. [Online]. Available: http://www.aertecsolutions.com/download/infographics/infografia-UAS-ES-v02.pdf.

[17] flying machines, "flying machines," 1914. [Online]. Available: http://flyingmachines.ru.

[18] Crown Copyright, "the aviationist," 29 Julio 2012. [Online]. Available: http://theaviationist.com/tag/general-atomics-mq-1-predator/page/5/.

[19] Ar Drone, "ardrone 2.0," 2015. [Online]. Available: http://ardrone-2.es/especificaciones-ar-drone-2/.

[20] DJI, "LA NACION," 09 abril 2015. [Online]. Available: http://www.lanacion.com.ar/1782675-dji-actualiza-su-linea-de-drones-con-el-modelo-phantom-3.

[21] B. digital, "Biblioteca digital," [Online]. Available: http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm.

[22] D. L. Aristizábal Ramírez and C. A. Ramírez Martínez , "Conceptos Básicos del Procesamiento Digital de Imágenes Usando OrquideaJAI," 2006, pp. 6-10.

[23] A. e. I. i. Dpt. Electronica and C. Platero, "Procesamiento digital de Imagenes," pp. 92-95.

[24] N. &. A. M. Vladimirovich, "Situation control of unmanned Aerial Vehicles for Road Traffic Monitoring," *Modern Applied Science,* no. 5, p. 9.

[25] Puri, "A survey of unmanned aerial vehicles (UAV) for traffic surveillance.," 2005.

[Kanistras, Martins, Rutherford and Valavanis, "Survey of unmanned aerial

26] vehicles (UAVs) for traffic monitoring," in *Handbook of Unmaneed Aerial Vehicles*, K. P. Valavanis & G. j. Vachtsevanos, 2015, pp. 2643-2666.

[27] Bethke, Baumgartner and Gabele, Airborne Road Traffic Monitoring with Radar, 2007.

[28] Fang, Lu, Tian and Miao, "An Improved Object Tracking Method in UAV Videos," in *Procedia Engineering*, 2011, pp. 634-638.

[29] S. Lederer, C. Müller and C. Timmerer, "Dynamic Adaptative Streaming over HTTP dataset," in *Proceedings of the ACM Multimedia System Conference* , New York, 2012.

[30] N. Apellido, "Titulo del articulo en conferencia," in *Nombre de la conferencia*, Ciudad, 2011.

[31] Bussiness Wire, "Research and Markets: Industrial Wireless Sensor Network Sensor & Technology Market Trends & Forecasts to 2013-2020 for the Oil & Gas, Energy & Power, Automotive, & Food & Beverage Industries," 15 Enero 2015. [Online]. Available: http://www.businesswire.com/news/home/20150108005601/en/Research-Markets-Industrial-Wireless-Sensor-Network-Sensor. [Accessed 13 Octubre 2015].

[32] global security, [Online]. Available: www.globalsecurity.org.

[33] P. M. Institute, Proyect Management Body Of Knowledge, Estados Unidos: Project Management Institute, 2012.

[34] "The UAV," [Online]. Available: http://www.theuav.com/.

[35] F. A., F. J., V. E and L. G.S., "UAV obstacle avoidance using image processing techniques," in *Tecnologies for practical robot applications*, Woburn, MA, 2012.

[36] B. Coifman, M. McCrod, R. Mishalani, M. Iswalt and Y. ji, "Roadway traffic monitoring from an unmanned aerial vehicle," *IET,* pp. 11-20, 206.

[37] R. Deepthi, M. C. f. I. Sci. and Sankaraiah, "Implementation of mobile platform using Qt and OpenCV for image processing applications," in *Open Systems (ICOS), 2011 IEEE*, Langkawi, 2011.

[M. S. Ruiz and J. V. Jorge, "Funtes de Infromacion Primarias, Secundarios y

38] Terciarias," ENERO 2008. [Online]. Available: http://ponce.inter.edu/cai/manuales/FUENTES-PRIMARIA.pdf.

[39] s. a. zavala and F. Dolly Smith, "Implementacion de una metodologia para el acceso, registro, analisis y trazabilidad de informacion para el informe policial de accidentes de transito (IPAT); dirigido a las necesidades de la direccion de transito y transporte de bucaramanga," Bucaramanga, 2015.

[40] Z. Zhang, Y. Hou, Y. Wang and J. Qin, "A traffic flow detection system combining optical flow and shadow removal," in *Intelligent Visual Surveillance*, Beijing, 2011.