



Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias en Inteligencia Artificial.

DETECCIÓN INTELIGENTE DE PATRONES EN IMÁGENES PARA LA
RECONSTRUCCIÓN DE UN MAPA DE PUNTOS Y POSICIÓN DE UNA CÁMARA
EN MOVIMIENTO.

Tesis

Que como parte de los requisitos para obtener el Grado de
Maestro en Ciencias en Inteligencia Artificial

Presenta

I.C. Luis Rogelio Román Rivera.

Dirigido por:

Dr. Jesús Carlos Pedraza Ortega

Dr. Jesús Carlos Pedraza Ortega
Presidente

Dr. Juan Manuel Ramos Arreguín
Secretario

Dr. Saúl Tovar Arriaga
Vocal

Dr. Efrén Gorrostieta Hurtado
Suplente

Dr. Marco Antonio Aceves Fernandez
Suplente

Centro Universitario Querétaro, Qro.
Junio de 2019
México



Dirección General de Bibliotecas y Servicios Digitales
de Información



DETECCIÓN INTELIGENTE DE PATRONES EN
IMÁGENES PARA LA RECONSTRUCCIÓN DE UN
MAPA DE PUNTOS Y POSICIÓN DE UNA CÁMARA EN
MOVIMIENTO

por

Luis Rogelio Román Rivera

se distribuye bajo una [Licencia Creative Commons
Atribución-NoComercial-SinDerivadas 4.0
Internacional](https://creativecommons.org/licenses/by-nc-nd/4.0/).

Clave RI: IGMAN-116006

RESUMEN

El trabajo de tesis tiene como finalidad obtener una nube de puntos a partir de un diseño básico de un proceso SLAM (Simultaneous Localization And Mapping) utilizando un método directo el cuál utiliza el error fotométrico como elemento básico en la detección de patrones y correspondencias en imágenes consecutivas adquiridas en el tiempo, las imágenes son capturadas por una sola cámara (monocular), a partir de los patrones detectados y las correspondencias se calcula la posición de la cámara y se realiza la proyección de los puntos en un modelo en 3D como nube de puntos. El proceso de funcionamiento SLAM requiere de al menos dos imágenes tomadas de manera incremental, una de las imágenes se toma como cuadro clave a partir del cual los patrones detectados se corresponden en la siguiente imagen. La inicialización y selección de cuadros clave se realiza por medio del flujo óptico calculado a partir de una función costo usando campos de Markov aleatorios cuya minimización es realizada usando gradiente descendente estocástico [1], y la construcción de puntos en 3D mediante matriz esencial, el presente trabajo toma como base la guía para construcción de un proceso SLAM de [2].

(Palabras clave: Campos Aleatorios de Markov, Gradiente Descendiente Estocástico, SLAM, Nube de puntos 3D)

SUMMARY

The goal of this Thesis is to get a cloudpoint from a basic SLAM (Simultaneous Localization And Mapping) process design using a direct method which used photometric error as a basic element in pattern and correspondences detection on consecutive images taken in time, these images are acquired using only one camera (monocular), from the detected patterns and correspondences a camera positions is computed and a point projection is performed in a 3D model as a cloudpoint. A basic SLAM process requires at least two images taken incrementally in time, one of these images is considered as a keyframe from which patterns are computed looking for correspondences in the second image. The initialization and keyframe selection is made by an optical flow which is computed in a cost function based in Markov Random Fields that is minimized using Stochastic Gradient Descent [1], the 3D cloudpoint construction used an essential matrix. This Thesis work is heavily based in a guide to design SLAM processes [2].

(Key words: Markov Random Fields, Stochastic Gradient Descent, SLAM, 3D Cloud point)

A mi familia

AGRADECIMIENTOS

Agradezco enormemente a todos mis profesores que brindaron su amable tiempo y paciencia en todas las asignaturas y que tuvieron mente abierta y brindaron su confianza en mis condiciones de no ser un estudiante regular en el sentido de estar completamente enfocado al estudio, agradezco la confianza que me dieron y por creer en mi capacidad al realizar las actividades que la maestría requería.

Agradezco especialmente a mi Asesor el Dr. Jesús Carlos Pedraza Ortega, por que con su gran experiencia logró sacar lo mejor de mí y hacer que creciera en habilidades y conocimiento.

Especialmente agradezco mucho a mi familia, mi esposa e hijos que también tuvieron que donar de su tiempo pues estuve estudiando cuando pude haber estado jugando o divirtiendome con mis hijos y a mi esposa agradezco porque hubo muchos fines de semana que no pude planear salir en familia debido a que tenía que entregar trabajos y tareas.

Sin duda alguna volver a la vida estudiantil ha sido revitalizante para mí y el día de hoy considero que soy una mejor persona gracias a la Universidad Autónoma de Querétaro y a la Facultad de Ingeniería, gracias a todos por mantener y fomentar este espacio tan libre y enriquecedor para todos quienes estudiamos aquí.

Finalmente y con todo respeto para quien difiere en creencias, agradezco a Dios que siempre está conmigo y quién me dió fuerzas para seguir adelante cuando el cuerpo ya rogaba descanso.

ÍNDICE GENERAL

Resumen	1
Summary	3
Dedicatoria	5
Agradecimientos	7
1. INTRODUCCIÓN	15
1.1. Inteligencia artificial	15
1.2. Odómetro	17
1.3. Fotogrametría	17
1.4. Estado del arte.	18
1.5. Importancia del tema	18
1.6. Justificación	19
1.7. Descripción del problema	19
1.8. Hipótesis	20
1.9. Objetivos	20
1.9.1. Objetivo general	20
1.9.2. Objetivos específicos	20
2. MARCO TEÓRICO	21
2.1. Reconstrucción 3D	21
2.2. Estructura desde movimiento.- (SFM Structure from motion)	23
2.3. Odometría visual.- (VO Visual odometry)	24
2.4. Localización y mapeo simultáneos SLAM (Simultaneous localization and mapping)	26
2.5. Campos aleatorios de markov	30
2.5.1. Función de energía	31
2.5.2. Modelos aleatorios de markov por pares	31
2.5.3. Modelos aleatorios de markov de alto orden	31
2.6. Gradiente descendiente	31
2.6.1. Gradiente descendiente por lotes	31
2.6.2. Gradiente descendiente estocástico	32
2.7. Materiales	33
2.7.1. Dispositivos ARM	33
2.7.2. Dispositivos x86_x64	34

2.7.3.	Cámaras digitales	34
2.7.4.	Cámara de profundidad intel realsense d435	35
3.	METODOLOGÍA	37
3.1.	Descripción general	37
3.2.	Captura de imagen	38
3.3.	Conversión a grises, preprocesamiento, 1er y 2do cuadro	38
3.4.	Estimación del flujo óptico	39
3.5.	Función costo ó de energía	39
3.5.1.	Factor de intensidad	39
3.5.2.	Factor de distancia	40
3.5.3.	Factor de vecindad	40
3.5.4.	Energía total	40
3.5.5.	Minimización de la función de energía.	40
3.5.6.	Correspondencia de puntos clave.	42
3.6.	Triangulación y cálculo de la posición de la cámara.	43
4.	RESULTADOS	45
4.0.1.	Software y hardware utilizado	45
4.0.2.	Validación del gradiente descendiente estocástico	49
4.0.3.	Comparación contra camara realsense D435	56
4.0.4.	Nubes de puntos 3D	57
5.	CONCLUSIONES Y TRABAJOS FUTUROS	63
	Apéndices	69
.1.	Artículo	69
.2.	Requisito manejo de la lengua inglés	82
.3.	Código C++	84
.3.1.	Gradiente de un pixel	84
.3.2.	Energía de un pixel	85

ÍNDICE DE TABLAS

2.1. Características cámara intel realsense d435	35
4.1. Error y valores de Gradiente Descendiente Estocástico	54
4.2. Error y valores de Gradiente Descendiente Estocástico	55
4.3. Resolución y Puntos sin preprocesamiento	56
4.4. Resolución y Puntos usando CannyEdge	56
4.5. Resolución y Puntos usando SLIC superpixel	57
4.6. Resolución y Puntos usando SLIC superpixel	57

ÍNDICE DE FIGURAS

1.1. Entrevista de Barack Obama con Joi Ito (MIT) [3].	15
1.2. Árbol de la Inteligencia Artificial [4].	16
1.3. Odómetro [5]	17
1.4. Odómetro Romano [5]	17
1.5. Fotogrametría aérea I [6]	17
1.6. Fotogrametría aérea II [6]	17
2.1. Taxonomía de las principales metodologías utilizadas en la adquisición de imágenes en 3D [7].	21
2.2. Adquisición de datos de Contacto [8]	22
2.3. Resonancia Magnética Nuclear (RMN) [9]	22
2.4. Características detectadas en un proceso SLAM [10]	22
2.5. Resultado del proceso SLAM (nube de puntos) [10]	22
2.6. Resultado del proceso SLAM Denso [11]	23
2.7. Características detectadas [11]	23
2.8. El problema SFM [12].	23
2.9. Odometría Visual [12].	24
2.10. Diagrama de bloque proceso de Odometría Visual [12].	25
2.11. El problema SLAM [13].	26
2.12. Densidad de puntos variable [14].	27
2.13. Proceso SLAM	28
2.14. Modelo 3D resultante	29
2.15. Vecinos y Cliques [15].	30
2.16. Raspberrypi 3b.	33
2.17. Beaglebone black.	33
2.18. Cámara Celular.	34
2.19. Cámara USB.	34
2.20. Cámara de profundidad Intel Realsense d435 [16].	35
3.1. Método propuesto basado en la etapa inicial SLAM [2].	37
3.2. Figura ejemplo cuadro clave.	38
3.3. Figura ejemplo 2do. cuadro	38
4.1. Cuadro clave en gris.	45
4.2. Ciclos de enfriamiento Gradiente Descendiente imagen completa	45
4.3. Cuadro Clave Preprocesado con el algoritmo <i>canny edge detection</i>	46

4.4. ciclos de enfriamiento Gradiente Descendiente imagen preprocesada con el algoritmo <i>canny edge detection</i>	46
4.5. Nube 3D Figura 3.2 y 3.3 vista 1.	46
4.6. Nube 3D Figura 3.2 y 3.3 vista 2.	46
4.7. Nube 3D Figuras 3.2 y 3.3 puntos correctos vista 1.	47
4.8. Nube 3D Figuras 3.2 y 3.3 puntos correctos vista 2.	47
4.9. Nube 3D Figuras 3.2 y 3.3 aplicando <i>canny edge detection</i> vista 1.	47
4.10. Nube 3D Figuras 3.2 y 3.3 <i>canny edge detection</i> vista 2.	47
4.11. Cocina 1.	47
4.12. Cocina 2.	47
4.13. Cocina nube de puntos 1.	48
4.14. Cocina nube de puntos 2.	48
4.15. Camino 1.	48
4.16. Camino 2.	48
4.17. Nube 3D Figuras 4.15 y 4.16 vista 1.	48
4.18. Nube 3D Figuras 4.15 y 4.16 vista 2.	48
4.19. SGD epochs Laptop webcam	49
4.20. SGD epochs webcam c170	50
4.21. SGD epochs Mobile camera	51
4.22. SGD epochs Canon T3i camera	52
4.23. Slope values from a pixel SGD process.	52
4.24. Pixel Intensity variation during GD.	53
4.25. Pixel position variation during GD.	53
4.26. Salida PLY, Cámara Laptop.	58
4.27. Salida PLY, Cámara Laptop usando CannyEdge.	58
4.28. Salida PLY, Cámara Laptop usando SLIC superpixel.	59
4.29. Salida PLY, Cámara Web USB.	59
4.30. Salida PLY, Cámara Web USB usando Canny Edge	60
4.31. Salida PLY, Cámara Web USB usando SLIC superpixel.	60
4.32. Salida PLY Cámara Integrada Celular.	61
4.33. Salida PLY Cámara Intel Realsense D435.	62

1. INTRODUCCIÓN

1.1. Inteligencia artificial

¿Que es Inteligencia Artificial? En el periodo de tiempo en el que se está trabajando en la presente tesis, este es un tema de moda, por ejemplo en las agendas gubernamentales de países de primer mundo se encuentra como prioridad el desarrollo de la Inteligencia Artificial, una referencia importante del impacto de este tema es la entrevista del expresidente de los Estados Unidos de América, Barack Obama Fig. 1.1 [3], temas como automóviles auto guiados, reconocimiento de voz, clasificación de imágenes, diagnósticos médicos automatizados son algunos temas cada vez mas populares, de la misma manera existe mucha desinformación y es común escuchar gracias a la ciencia ficción opiniones de un futuro fatalista.



Figura 1.1: Entrevista de Barack Obama con Joi Ito (MIT) [3].

Algoritmos capaces de aprender y adaptarse a situaciones y ambientes nuevos es una característica de los programas con Inteligencia Artificial [17]. En la Fig. 1.2 se observa una clasificación de alto nivel de la Inteligencia Artificial [4].

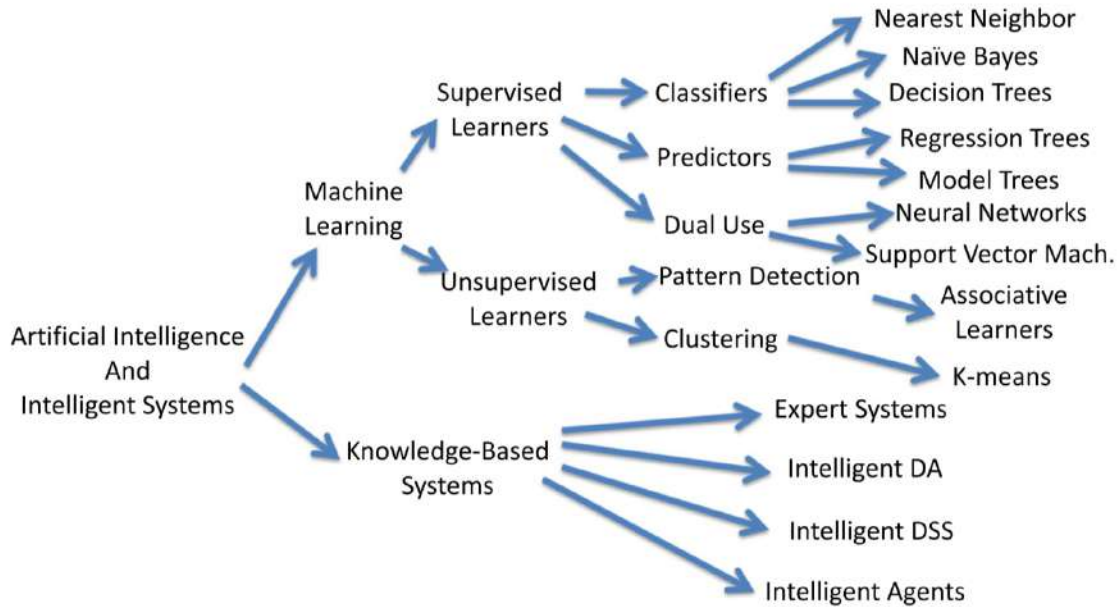


Figura 1.2: Árbol de la Inteligencia Artificial [4].

El presente trabajo se enfoca en métodos de reconocimiento de patrones (pattern recognition) ubicado debajo de aprendizaje máquina (machine learning) en la Fig. 1.2.

1.2. Odómetro

La medición de rutas o caminos mediante el uso de un odómetro Fig. 1.3 (“hodometron” hodos= camino, metron = medida) es una actividad que se remonta a la dinastía china HAN [5].

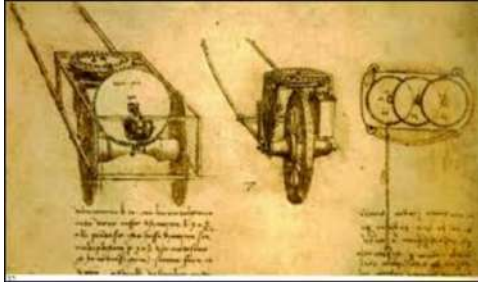


Figura 1.3: Odómetro [5]

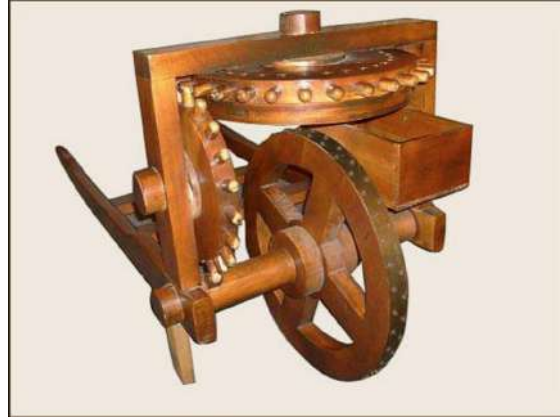


Figura 1.4: Odómetro Romano [5]

El odómetro brindaba una manera de medir la distancia recorrida mediante una cuenta de piedras que iban cayendo a un recipiente. Conforme se iba avanzando, el odómetro una rueda dentada la cuál accionaba un mecanismo de engrane que a su vez giraba una segunda rueda donde se almacenaban las piedras Fig. 1.4.

1.3. Fotogrametría

Es una técnica para medir y procesar distancias y ángulos en fotografías con el objetivo de obtener el tamaño, forma y posición de los objetos en fotografías Fig. 1.5 . La fotogrametría se usaba regularmente para la medición de predios y mapeo de rutas mediante la toma de fotografías aéreas Fig. 1.6, las cuales posteriormente se procesaban de manera manual para obtener información [6].

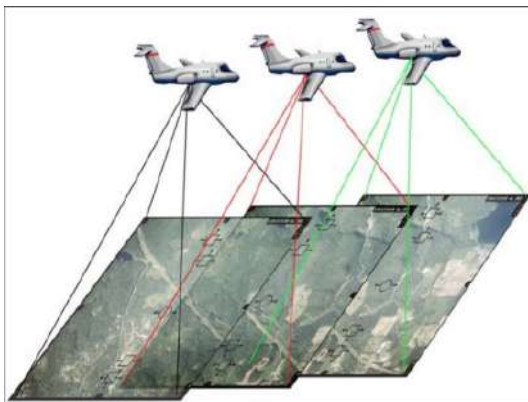


Figura 1.5: Fotogrametría aérea I [6]

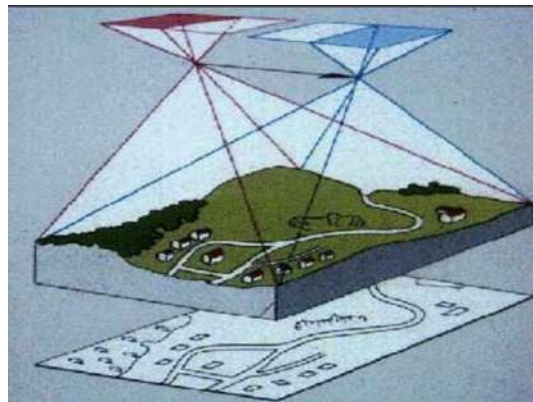


Figura 1.6: Fotogrametría aérea II [6]

1.4. Estado del arte.

Los métodos directos y semi directos se vuelven populares debido a que proveen mayor información comparados contra los métodos basados en características [18] y [19], los métodos semidirectos disminuyen la cantidad de puntos que se corresponden entre dos imágenes con la finalidad de disminuir el tiempo de cómputo, existen diversos esfuerzos para reducir la cantidad de puntos, por ejemplo medir la calidad de los puntos correspondidos y seleccionar unicamente los mejores basados en una función costo [14], el uso de segmentación es otra opción para reducir la complejidad de la imagen y enfocar el tiempo de computo unicamente en áreas de interés [20] y [21]. La investigación en métodos de flujo óptico también continua, generando resultados de mayor calidad en una mayor amplitud de casos [22] y [23], también se observan métodos probabilísticos para aumentar la precisión disminuyendo el tiempo de computo en los procesos de reconstrucción en 3D [24].

1.5. Importancia del tema

Se observa en el estado del arte recopilado que existen áreas de mejora en un proceso SFM como:

- Extracción de puntos clave y correspondencia de referencias.- Hay opciones variadas para la extracción de puntos clave que por su alto costo computacional lo hacen prohibitivo para cierto tipo de aplicaciones y algunas otras técnicas relativamente eficientes manifiestan perdidas significativas en precisión. Técnicas de mayor eficiencia son de gran valor [12].
- La densidad de puntos extraídos.- Se pueden extraer diferentes densidades de puntos en los cuadros clave, la densidad afecta directamente en la construcción del mapa de puntos 3D y en el desempeño del proceso SLAM. Mayor eficiencia para seleccionar la densidad adecuada es tema de investigación [14].

El trabajo propuesto pretende implementar la detección de patrones en imágenes basados en técnicas de Inteligencia Artificial para la reconstrucción de un mapa de puntos y posición de un agente en movimiento, usando imágenes recopiladas mediante cámaras digitales como la de un teléfono celular con cámara, o una cámara usb en un dispositivo como el raspberry pi los cuales son dispositivos de bajo costo y comunes hoy en día o en una PC.

- Movilidad. Un dispositivo embebido y/o un teléfono celular nos permite capturar información mediante una cámara digital, la capacidad de ejecución de un algoritmo de reconstrucción en 3D en un dispositivo móvil, permite obtener resultados in situ sin necesidad de un procesamiento posterior o remoto en otra plataforma.
- Reducción de costos. Las cámaras digitales en dispositivos móviles se han vuelto comunes hoy en día es difícil encontrar un teléfono celular de nueva o reciente generación que no incluya al menos una cámara digital, el poder aprovechar esta disponibilidad sin la necesidad de invertir en un equipo adicional reduce costos en la obtención de modelos 3D. De la misma manera las cámaras USB han reducido su costo y hay una gran variedad disponibles en el mercado que permiten su uso en dispositivos embebidos.

- **Facilidad y Disponibilidad de la Herramienta.** La impresión en 3D empieza a convertirse en un servicio común como lo es el fotocopiado, más aún existen opciones de compra para usuario final cada vez a un menor costo, sin embargo las herramientas para reconstrucción en 3D no se encuentran disponibles en la misma proporción.

1.6. Justificación

Existen alternativas a la reconstrucción 3D de tipo Óptico, sin embargo la mayoría depende de equipo profesional como cámaras especiales, proyectores de franjas y puntos, LIDAR, y para obtener resultados inmediatos o en un corto plazo se requiere de equipos robustos [12].

La reconstrucción 3D mediante el proceso SLAM sigue avanzando en los últimos años, con una tendencia a ofrecer soluciones capaces de ejecutarse y generar resultados en dispositivos móviles [25]. Sin embargo aún existen muchas limitantes y variaciones que se requieren seguir investigando en el diseño del proceso SLAM [2].

La construcción y análisis de modelos 3D es cada vez más común y algunos ejemplos donde se utiliza son: área de la salud, en el reconocimiento de identidad, videojuegos, películas, realidad virtual, diseño de productos, modelado y simulación.

El desarrollo y pruebas que involucran el desarrollo del proceso SLAM no requiere hacer pruebas con seres vivos (personas o animales) por lo que no representa un peligro para la salud.

1.7. Descripción del problema

Existe una dependencia de procesamiento en un equipo robusto para la reconstrucción 3D a partir de imágenes y video donde se generan resultados de alta calidad en tiempos aceptables, basado en [12].

El uso de dispositivos móviles, dispositivos embebidos que se pueden encontrar en vehículos no tripulados UAV (Unmanned Aerial Vehicle) ó drones o en vehículos tripulados para la recolección de información por medio de cámaras de video en lugares poco accesibles, es una tendencia creciente, sin embargo gran parte de este tipo de equipos solo capturan la información, esta es transmitida y procesada en una central de computo robusto como puede ser desde una laptop, una pc, workstation y posterior al procesamiento y generación de resultados se realizan desiciones las cuales vuelven a enviarse a los equipos móviles para realizar una acción, basado en [2].

Resultados en corto plazo.- Existen situaciones donde es común que se cuente con dispositivos móviles pero no de computadoras de escritorio o una laptop, por ejemplo en caso de un temblor, o una emergencia la reconstrucción 3D en un dispositivo móvil y/o embebido permite compartir un modelo del sitio o zona del siniestro en un corto plazo ayudando a la logística de los rescatistas. En caso por ejemplo de alguna fractura o lesión la reconstrucción 3D permite brindar mayor información y detalle de manera remota. En vehículos no tripulados en ambientes aislados es importante generar resultados que permitan tener acciones sin depender de una central de procesamiento remoto, basado en [2]. El mapeo en ambientes

donde la tecnología GPS (Global Position System) es limitada. En un edificio con mala recepción, la reconstrucción 3D permite generar mapas de la estructura interna de un edificio o ubicación interna en alguna construcción, basado en [19].

1.8. Hipótesis

Es posible diseñar un proceso SLAM para reconstrucción 3D que utilice la información capturada por una cámara en base de imágenes digitales (video, fotografías) y que pueda ser implementado en sistemas embebidos y móviles con arquitectura ARM para la obtención de un mapa de puntos.

1.9. Objetivos

1.9.1 Objetivo general

Desarrollar, implementar y evaluar un proceso SLAM para la Detección de patrones en imágenes basados en técnicas de Inteligencia Artificial para la reconstrucción de un mapa de puntos y posición de una cámara en movimiento. Al implementar un pre procesamiento de imagen se pretende obtener mejores resultados con respecto a las métricas por definir, para poder ser ejecutado en un dispositivo con arquitectura ARM y x86_64

1.9.2 Objetivos específicos

- Diseñar un proceso SLAM que pueda ser ejecutado en una pc y en un dispositivo ARM
- Generar una ruta 3D de desplazamiento de la cámara que capturó las imágenes o video.
- Generar una nube de puntos por medio del proceso SLAM diseñado.
- Implementar pruebas de desempeño del proceso SLAM en sistemas embebidos y dispositivos móviles.
- Realizar pruebas para medir los resultados del proceso SLAM diseñado en una computadora personal.
- Realizar pruebas para medir los resultados del proceso SLAM diseñado en sistemas embebidos y dispositivos móviles.
- Realizar comparativa entre resultados obtenidos por la computadora personal, contra los obtenidos por los resultados obtenidos por los sistemas embebidos y dispositivos móviles.

2. MARCO TEÓRICO

2.1. Reconstrucción 3D

Existen diversas maneras de adquirir datos en un proceso de reconstrucción 3D, podemos observar en Fig. 2.1 la siguiente clasificación basada en [7]. Un ejemplo de ad-

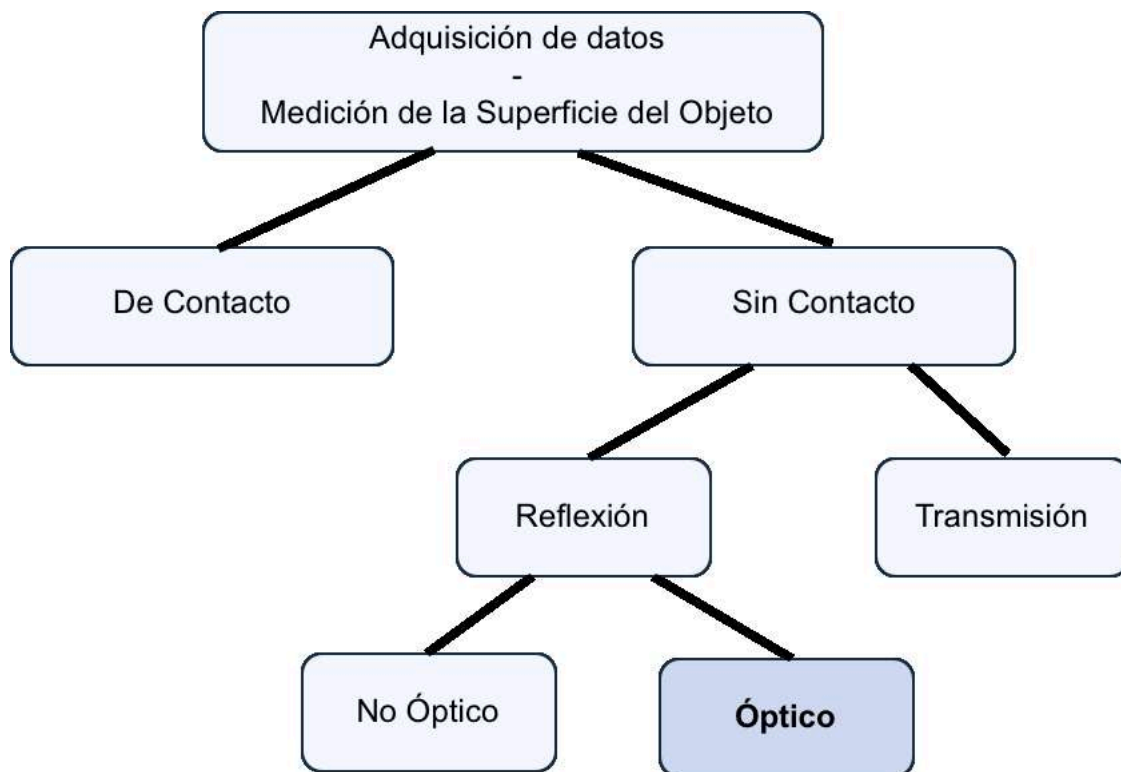


Figura 2.1: Taxonomía de las principales metodologías utilizadas en la adquisición de imágenes en 3D [7].

quisición de datos sin contacto es un brazo de medición como el que se muestra en Fig. 2.2 donde se requiere tomar mediciones mediante el contacto de la punta del brazo con la pieza a escanear en diferentes puntos. Por el contrario se pueden adquirir datos sin contacto, la resonancia electromagnética ver Fig. 2.3, se clasifica como transmisión ya que se genera un campo electromagnético para escanear un objeto de estudio [8] y [9].



Figura 2.2: Adquisición de datos de Contacto [8]



Figura 2.3: Resonancia Magnética Nuclear (RMN) [9]

En el presente trabajo de tesis se considera la adquisición de datos usando imágenes 2D por lo que la clasificación del método de adquisición es óptica, específicamente el trabajo se enfocará en un proceso SLAM similar al mostrado usado en [10] pero usando una sola cámara ó como en [11].



Figura 2.4: Características detectadas en un proceso SLAM [10]



Figura 2.5: Resultado del proceso SLAM (nube de puntos) [10]



Figura 2.6: Resultado del proceso SLAM Denso [11]

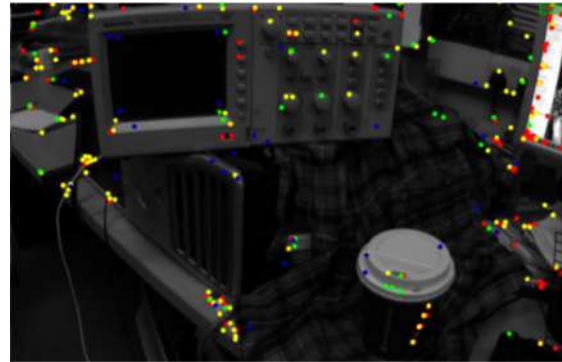


Figura 2.7: Características detectadas [11]

2.2. Estructura desde movimiento.- (SFM Structure from motion)

Es la recuperación de la estructura en 3 dimensiones desde un conjunto de imágenes en 2 dimensiones donde se consideran 3 etapas principales [12].

1. Extracción de patrones en Imágenes (puntos de interés, líneas, esquinas) desde distintas imágenes relacionando estos patrones entre las imágenes.
2. Estimación del movimiento de la cámara (se estima a partir de la extracción de patrones).
3. Recuperación de la estructura en 3D utilizando el movimiento estimado de la cámara y los patrones extraídos.

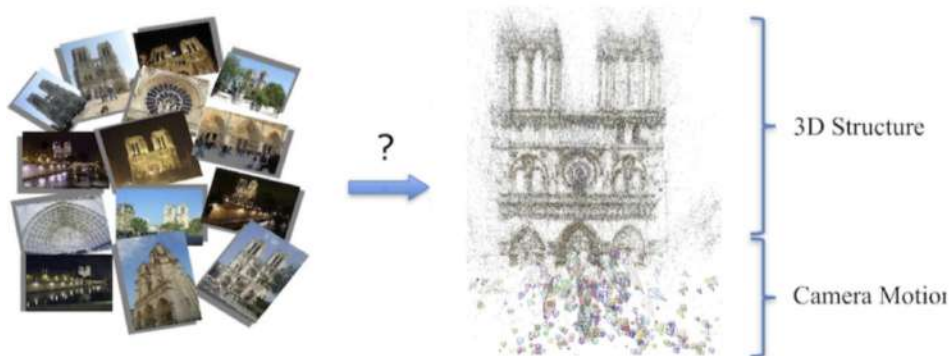


Figura 2.8: El problema SFM [12].

Estructura desde Movimiento abarca varios métodos como son Odometría Visual y SLAM (Localización y mapeo simultáneos). Se tiene en común la utilización de imágenes para la extracción de información para el cálculo de la posición de la cámara y cálculo de puntos en 3D para la creación de mapas.

2.3. Odometría visual.- (VO Visual odometry)

Es el proceso de estimación del movimiento propio de un agente (vehículos, robot) usando únicamente la entrada de una o múltiples cámaras sujetas al agente.

VO opera de manera incremental estimando la posición analizando los cambios de movimiento inducidos en las imágenes de la(s) cámaras. Se requiere que exista una sobreposición de la escena Fig. 2.9.

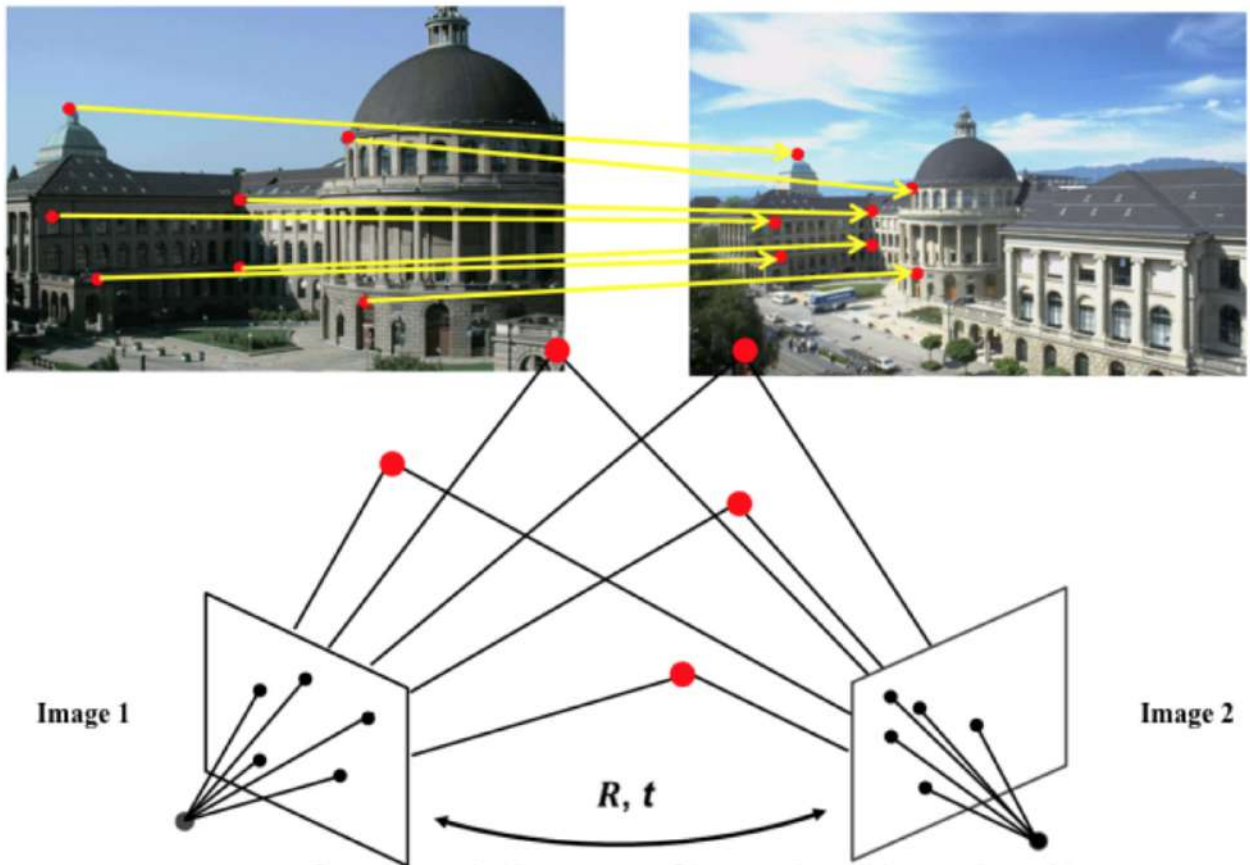


Figura 2.9: Odometría Visual [12].

VO puede obtener trayectorias con un error de entre 1 % y 2 % en la posición relativa, convirtiendo al proceso VO en un suplemento a sistemas de posicionamiento global ó GPS. En ambientes donde no se dispone de GPS, VO es de gran importancia [25]. El principio del funcionamiento de VO (fig. 2.11) requiere de al menos dos imágenes tomadas de manera incremental, en las cuales se detectan puntos de interés (feature detection) los cuáles se tratan como puntos clave ó referencias (landmarks), se corresponden los puntos clave idénticos en las imágenes (tracking) a partir de estos datos se realiza la estimación de movimiento Fig. 2.10 [25].

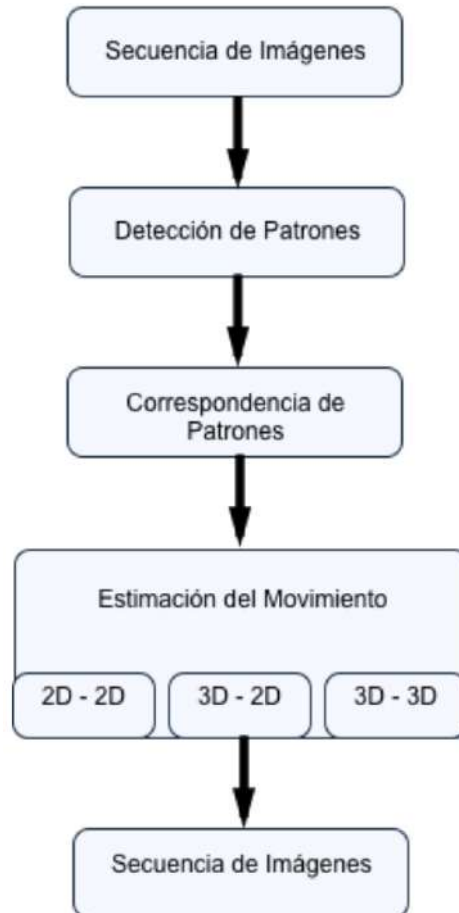


Figura 2.10: Diagrama de bloque proceso de Odometría Visual [12].

2.4. Localización y mapeo simultáneos SLAM (Simultaneous localization and mapping)

SLAM es un proceso por el cual un robot móvil puede construir un mapa del ambiente representando referencias (landmarks) y simultáneamente utilizar el mapa para deducir su posición, es importante notar que en el proceso SLAM no se requiere un conocimiento previo de la ubicación [13]. En la Fig. 2.11 se representan los siguientes puntos de un proceso SLAM [13].

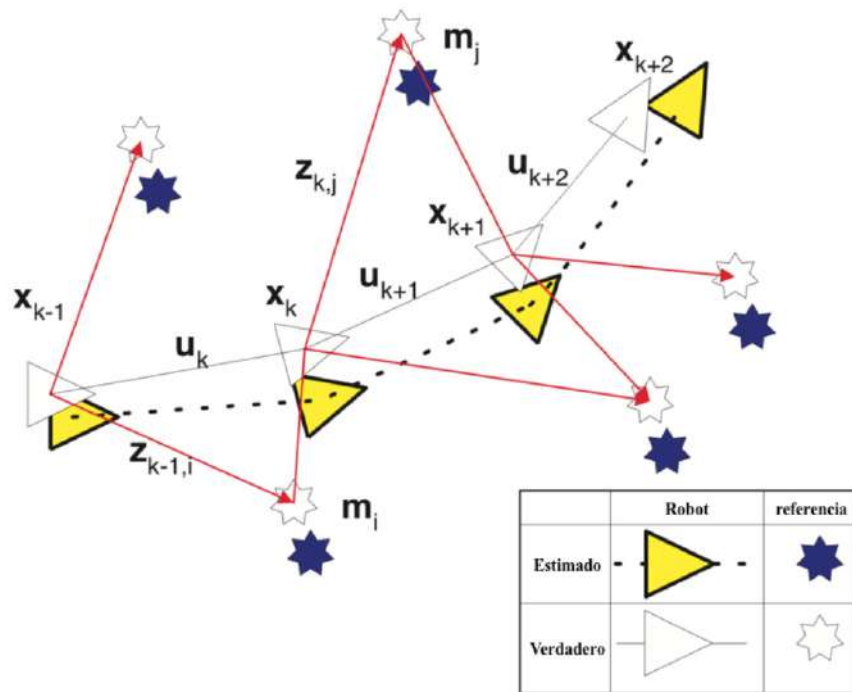


Figura 2.11: El problema SLAM [13].

- x_k : Vector estado que describe la orientación y ubicación del robot-vehículo.
- u_k : Vector control, aplicado en un tiempo $k-1$ el cuál cambia el robot-vehículo al estado x_k en el tiempo k
- m_i : Vector que describe la ubicación de la i -ésima referencia cuya ubicación se considera invariante el el tiempo.
- z_{ik} : Observación realizada desde el robot-vehículo de la referencia i en un tiempo k

El proceso SLAM considera los siguientes puntos [2].

1. Inicialización .- Se realiza un cálculo de la posición de la cámara y el cálculo de los puntos en un espacio 3D a colocar en un mapa por primera vez desde un cuadro clave (keyframe).
2. Selección de los cuadro clave (keyframes).- Se selecciona un segundo cuadro desde el cuál se calcula la posición en el espacio 3D de las referencias respecto al primer cuadro clave, el segundo cuadro clave pasa a tomar el lugar del primero hasta encontrar un nuevo cuadro clave.
3. Asociación de datos
4. Cálculo de profundidad de los puntos en las referencias y posición de la cámara. Se realiza el cálculo de profundidad en las referencias y mediante el mapa de referencia se calcula la posición de la cámara.
5. Actualización de mapa de puntos.- Se agregan nuevos puntos 3D al mapa de referencia.
6. Mantenimiento del mapa de referencia .- Se realizan ajustes al mapa de referencia según se vaya recopilando información adicional que proporcione mejoras en el mapa, por ejemplo si se detectan rutas con circuitos cerrados (loops) se realiza un ajuste global del mapa de puntos o un ajuste local a la ruta con circuito cerrado.

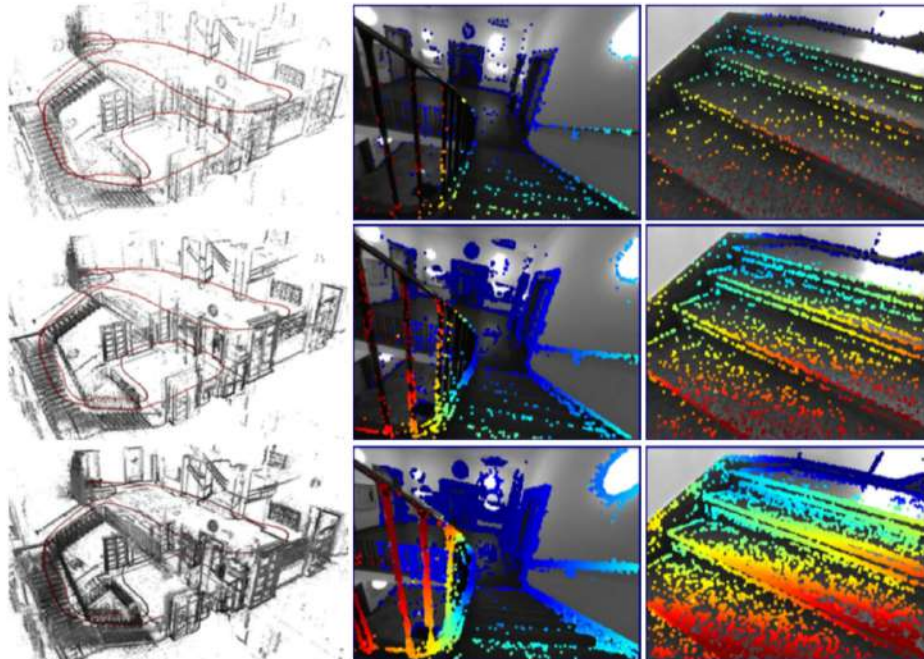


Figura 2.12: Densidad de puntos variable [14].

La reconstrucción de objetos en 3D mediante un proceso SLAM requiere de diversos pasos los cuáles pueden variar en requerimiento de computo sin embargo los cálculos requeridos al ser aplicados en imágenes digitales 2D, sub-imágenes y hasta nivel de pixel por cada imagen procesada se convierte en un proceso que se beneficia de la disponibilidad de recurso de procesamiento y memoria (M. P. Christian Forster, Davide Scaramuzza, 2014). El diseño del proceso SLAM puede variar en sus componentes y en el resultado, un reto actual es el diseño de un proceso SLAM es que se ajuste a dispositivos móviles y/o embebidos.

Podemos observar en la Fig. 2.12 diferentes densidades de puntos para la misma imagen en tres ejemplos, cada uno con un mapa de puntos con diferente densidad sin perder información clave para la interpretación del mapa [14].

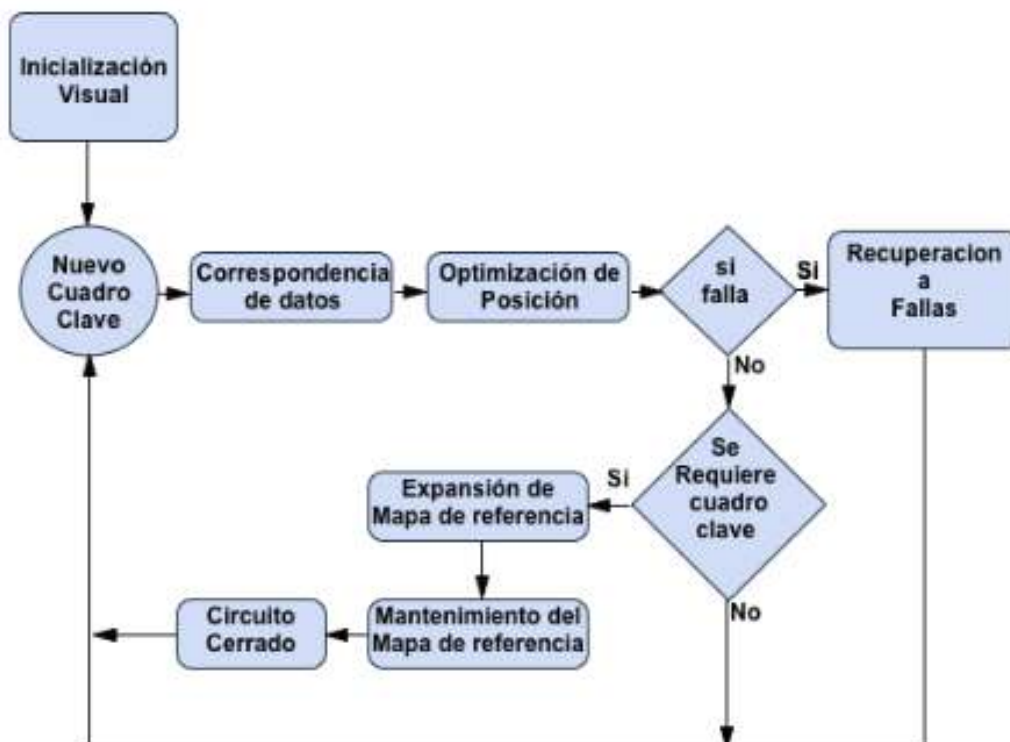


Figura 2.13: Proceso SLAM

En la Fig. 2.13 se observan los bloques contenidos en un proceso SLAM basado en cuadros clave.

En Fig. 2.14, se presenta un modelo 3D generado con una cámara intel realsense d435 de la Fig. 2.20 como ejemplo de resultado esperado.

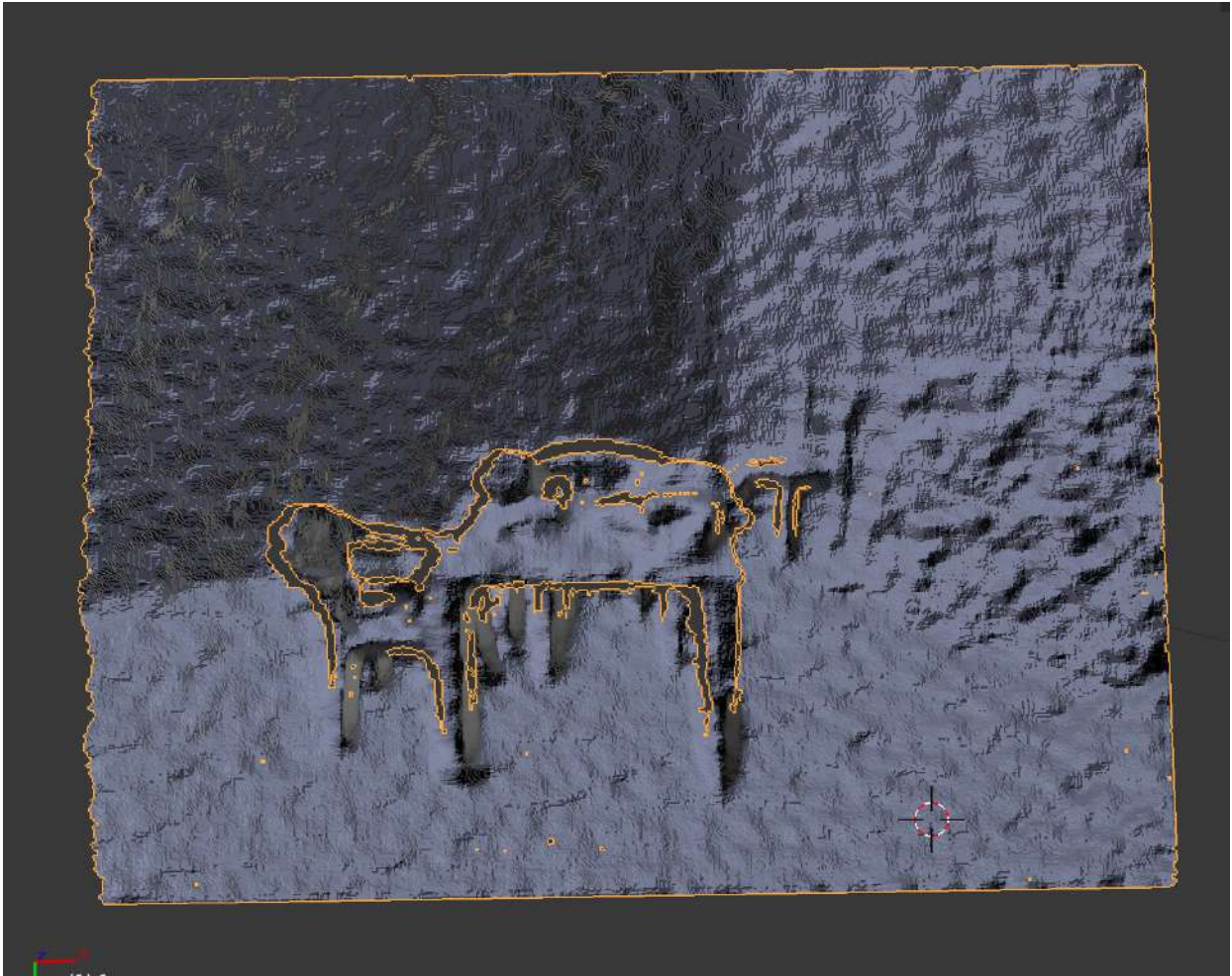


Figura 2.14: Modelo 3D resultante

2.5. Campos aleatorios de markov

La teoría de los campos aleatorios de Markov es una rama de la teoría de probabilidad que analiza dependencia contextuales o espaciales de algún fenómeno, es usada en el etiquetado visual para establecer distribuciones probabilísticas de etiquetas que interactúan entre sí [15].

Se definen sitios en un dominio S y se relacionan unos con otros mediante un sistema de vecindades definido en la Eq. 2.1.

$$\mathcal{N} = \{\mathcal{N}_i | \forall i \in S\} \quad (2.1)$$

Donde \mathcal{N}_i es el conjunto de sitios vecinos i , las relaciones de vecindad tienen las siguientes propiedades:

1. Un sitio no es vecino de si mismo.
2. Una relación de vecindad es mutua.

En una retícula regular de sitios S como la que se muestra en la Fig. 2.15 se observan ejemplos de vecinos considerados en a), b) y c) y cliques en d), e), f) y h) [15].

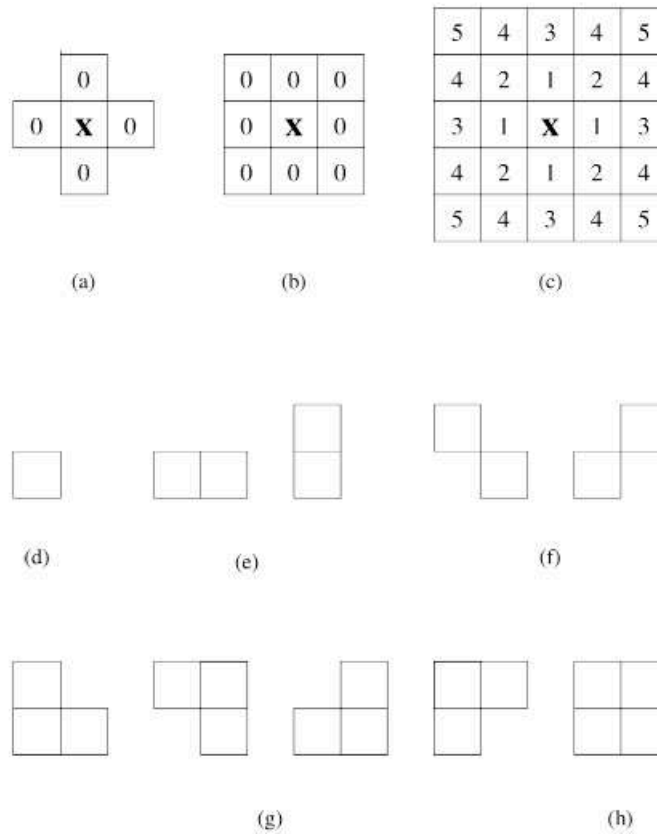


Figura 2.15: Vecinos y Cliques [15].

2.5.1 Función de energía

El rol de una función de energía en visión basada en minimización tiene dos características [15]:

1. Es la medida cuantitativa de la calidad global de la solución.
2. Es una guía para la búsqueda de la solución mínima.

2.5.2 Modelos aleatorios de markov por pares

El tipo mas común de Modelos Aleatorios de Markov es por pares [26], donde la suma de energía asociada se factoriza en una suma potencial de funciones definidas en los cliques de orden menor que tres [26].

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \theta_i(x_i) + \sum_{\{i,j\} \in \mathcal{E}} \theta_{ij}(x_{ij}) \quad (2.2)$$

2.5.3 Modelos aleatorios de markov de alto orden

Los Modelos Aleatorios de Markov de alto orden consideran funciones potenciales que están definidas en cliques de más de dos nodos y que no pueden ser descompuestas a menor nivel [26].

2.6. Gradiente descendiente

Las redes neuronales y el aprendizaje profundo (deep learning) son un tema en voga al momento de escribir esta tesis, uno de los algoritmos más usados para minimización de errores en ellas es el Gradiente Descendiente.

El gradiente descendiente es un método de minimización para una función objetivo $J(\theta)$ cuyos parámetros son actualizados en dirección opuesta al gradiente de la función objetivo $\nabla_{\theta} J(\theta)$ con respecto a los parámetros, el radio η de aprendizaje determina el tamaño de los pasos para alcanzar un mínimo local [1].

2.6.1 Gradiente descendiente por lotes

El gradiente descendiente por lotes (batch gradient descent) Eq. 2.3 calcula el gradiente de la función costo con respecto a sus parámetros para el dataset de entrenamiento completo [1], representa un procesamiento lento debido a la necesidad de procesar una gran cantidad de información comparado contra otras alternativas.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (2.3)$$

2.6.2 Gradiente descendiente estocástico

El método gradiente descendiente estocástico actualiza los parámetros de entrenamiento en cada iteración para optimizar funciones costo, las actualizaciones pueden tener una alta varianza causando que los resultados de la función costo fluctuen de manera marcada [1].

$$\theta = \theta - \eta \nabla J(\theta; x^{(i)}; y^{(i)}) \quad (2.4)$$

La fluctuación del gradiente descendiente estocástico permite saltar a potenciales mejores mínimos locales, sin embargo esto puede complicar el converger al mínimo local correcto pero se ha mostrado que si se decrementa lo suficiente mente despacio el radio de aprendizaje, se puede converger con un comportamiento similar al gradiente descendiente por Lotes para mínimos globales y locales en minimizaciones no convexas y convexas.

Los pesos son ajustados usando las siguientes Eqs. 2.5 y 2.6 basadas en gradiente descendiente estocástico [1].

$$Wx = Wx + Learningrate \times GradientX \quad (2.5)$$

$$Wy = Wy + Learningrate \times GradientY \quad (2.6)$$

2.7. Materiales

2.7.1 Dispositivos ARM

Actualmente los sistemas embebidos y móviles se encuentran dominados por la arquitectura ARM (Advanced Risc Machines) la cuál es una arquitectura RISC (Reduced Instruction Set Computer), esta arquitectura es licenciada para su uso y desarrollada por ARM. Dispositivos móviles y embebidos como consolas de juegos Nintendo 3DS, Nintendo Switch, Cámaras Fotográficas, Televisores, Teléfonos celulares Android (Google, Samsung, HTC, Huawei, etc) y iOS (Apple), Raspberry Pi (raspberrypi-3-model-b,"2017), BeagleBone black ("BeagleBone Black,"2017)entre otros, utilizan esta arquitectura debido a sus ventajas de desempeño y bajo consumo de energía. ARM está soportado en una gran variedad de sistemas operativos como lo son: Linux, Android, iOS, FreeRTOS, FreeBSD, OpenSolaris etc. Es claro que la integración de sistemas con procesadores ARM será cada vez más común, hoy en día incluso los podemos encontrar en lavadoras, refrigeradores y en nuestros automóviles [27].



Figura 2.16: Raspberrypi 3b.

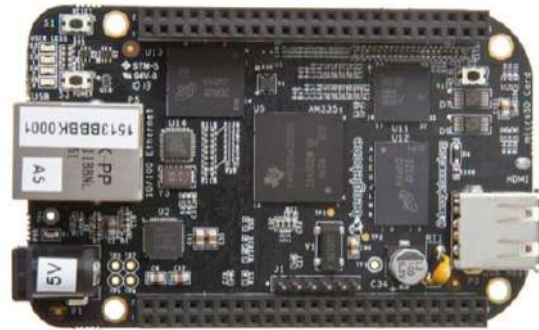


Figura 2.17: Beaglebone black.

2.7.2 Dispositivos x86_x64

La arquitectura utilizada en dispositivos como workstations y servidores y es la x86_64 mismo caso para laptops, donde podemos encontrar procesadores (AMD e Intel) en distintas configuraciones de velocidades de reloj, memoria cache y número de núcleos de procesamiento (cores) [27].

2.7.3 Cámaras digitales

Actualmente es común encontrar al menos una cámara digital en los dispositivos móviles, en casos de los sistemas embebidos las opciones de cámaras usb es abundante. Mediante el uso de cámaras montadas en celulares o sistemas embebidos es posible capturar video y fotografías en cualquier lugar al que se pueda acceder, las imágenes y el video son cada vez de mayor calidad.



Figura 2.18: Cámara Celular.



Figura 2.19: Cámara USB.

Tabla 2.1: Características cámara intel realsense d435

Dispositivo	Intel Realsende D435
Tecnología	Estereoscópico Activo
Rango	0.2 - 4.5 m
Resolución	1280 x 720 pix
Frecuencia de cuadros	30, 60, 90 fps
Campo visual	85,2° x 58°

2.7.4 Cámara de profundidad intel realsense d435

Se utiliza una cámara de profundidad intel realsense d435 Fig. 2.20 basada en tecnología de luz estructurada por medio de un proyecto laser infrarojo [16].



Figura 2.20: Cámara de profundidad Intel Realsense d435 [16].

Se resumen las propiedades de la cámara en la Tabla 2.1.

La profundidad es calculada por medio de un circuito ASIC, los algoritmos están basados en correlaciones de correspondencias [16], un ejemplo de la nube de puntos 3D lograda en este dispositivo puede observarse en la Fig. 2.14.

3. METODOLOGÍA

3.1. Descripción general

Se propone un procedimiento, utilizando el cálculo del flujo óptico a partir de una función basada en campos aleatorios de Markov y minimizada con el método de gradiente descendente estocástico como se muestra en la Fig. 3.1 en la etapa de inicialización de un proceso SLAM, etapa donde se requiere tomar una imagen como cuadro clave y generar mediante una segunda imagen, correspondencias, por medio de las cuales se calculan las posiciones de las cámaras en las primera y segunda imagen y se triangulan las correspondencias proyectando una nube de puntos 3D [2].

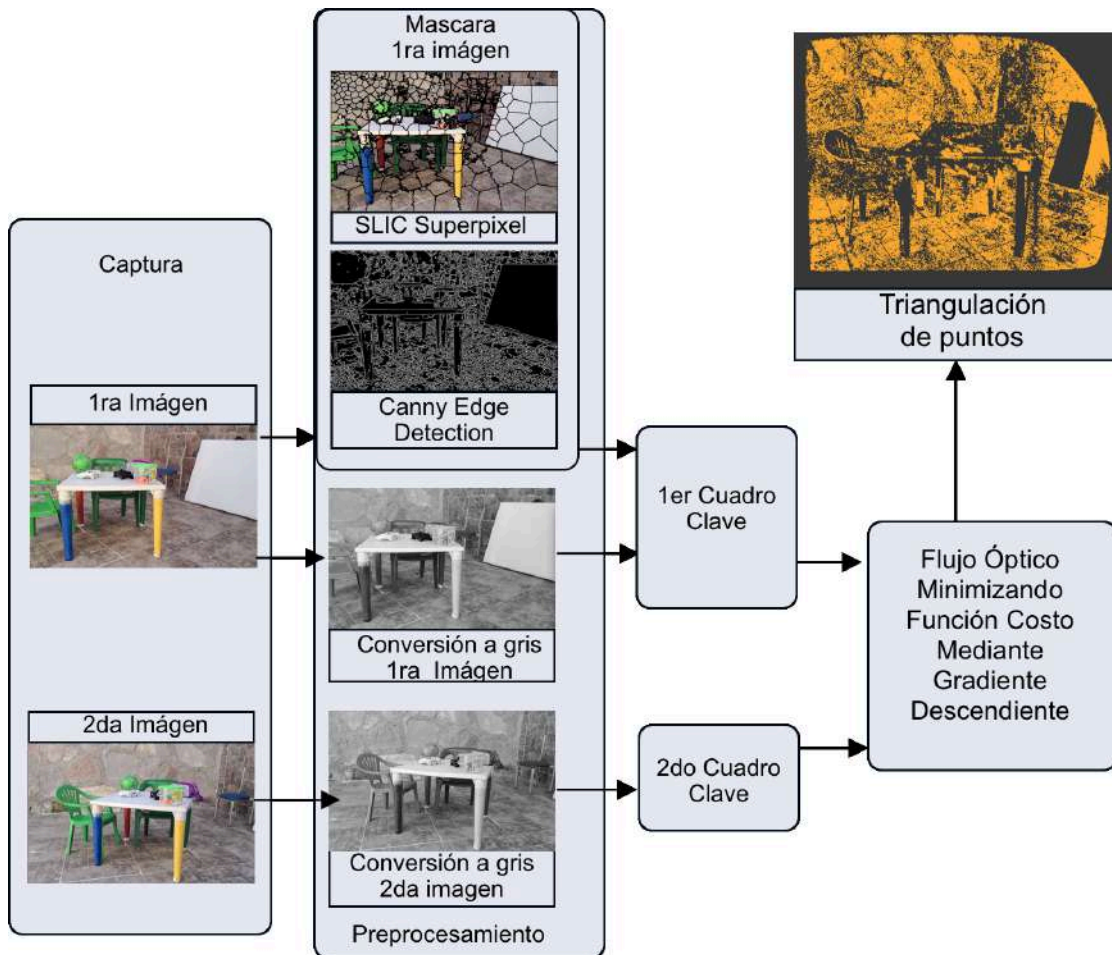


Figura 3.1: Método propuesto basado en la etapa inicial SLAM [2].

3.2. Captura de imagen

En el presente trabajo se realiza la captura de las imágenes como se muestra en la Fig. 3.1 mediante la cámara web integrada de una laptop y mediante la cámara integrada de un dispositivo móvil. Las imágenes se almacenan y procesan en una laptop Macbook.

3.3. Conversión a grises, preprocesamiento, 1er y 2do cuadro



Figura 3.2: Figura ejemplo cuadro clave.



Figura 3.3: Figura ejemplo 2do. cuadro

Se selecciona el 1er y 2do cuadro Figs. 3.2 y 3.3 de las imágenes capturadas, los cuadros pueden ser preprocesados, en el presente trabajo se utilizan los siguientes algoritmos *canny edge detection* [28] y *SLIC superpixel* [20], [29], [21], la primera imagen preprocesada se puede usar como máscara para reducir el número de píxeles procesados contra la segunda imagen y como consecuencia reducir el tiempo de cómputo, adicionalmente las imágenes se convierten a escala de grises para utilizar un solo canal de intensidad de color (gris), reduciendo opcionalmente el tiempo de cómputo al ciclo de enfriamiento de la función de energía. Se considera que *SLIC superpixels* trabaja las imágenes en un espacio de color CIELAB, clustereando áreas utilizando k-means con una función de distancia basada en su espacio de color y un k arbitrario de número de clusters, este método muestra un tiempo de cómputo lineal.

3.4. Estimación del flujo óptico

El flujo óptico se refiere al movimiento aparente del campo de visión 2D entre imágenes consecutivas en el tiempo. Es diferente al movimiento de los objetos en la escena, en el caso extremo en el movimiento en el eje de la cámara, no hay flujo óptico, por el contrario rotando la cámara hay flujo óptico.

Existen 2 métodos base para estimar el flujo óptico:

- 1) Lucas - Kanade. El cual genera un flujo de vectores disperso asumiendo un movimiento local constante y un brillo de cada pixel constante [30].
- 2) Horn Schunck. Genera un campo de flujo denso asumiendo un flujo de campo suave espacialmente [31].

Se obtiene el flujo óptico a nivel de pixel a partir de dos imágenes tomadas de una escena de manera consecutiva en el tiempo, se genera una lista de correspondencias de un conjunto de pixeles en la primera imagen a la segunda imagen. Se utilizan las Eqs. 3.1, 3.2 y 3.3 para el computo del flujo óptico a nivel de pixel.

El cálculo del flujo óptico y la triangulación de las correspondencias requieren que la cámara utilizada se calibre, evitando este tipo de distorsiones antes de procesar las imágenes. Se utilizan un proceso de calibración basado en [32].

3.5. Función costo ó de energía

Se utiliza una función de energía ó función costo basada ampliamente en [33], a continuación se detalla los componentes de la función costo.

Se definen los pixeles correspondientes a una imagen de tamaño $h \times w$ donde p_{ij} corresponde al i ésimo renglón y a la j ésima columna, y el flujo óptico correspondiente a este pixel P_{ij} $F = F_{ij}$, con $i = 1, \dots, h$ y $j = 1, \dots, w$ se definen los siguientes factores en un *Markov Random Field* ó campo aleatorio de Markov:

3.5.1 Factor de intensidad

$\phi_I : \phi_I(F_{ij}) = \exp(-E_I(F_{ij}))$ donde $E_I(F_{ij})$ es la intensidad de Energía del flujo óptico, se define para cada pixel como:

$$E_I(F_{ij}) = \frac{(I(p_{ij}) - I'(p_{ij} + F_{ij}))^2}{\alpha^2} \quad (3.1)$$

Donde I es la intensidad de pixel en la primer imagen e I' es la intensidad de pixel en la segunda imagen, el parámetro α es la varianza en la intensidad de la energía para controlar cuanto se compara el movimiento con la misma intensidad con movimiento con cambio en intensidad.

3.5.2 Factor de distancia

$\phi_D : \phi_D(F_{ij}) = \exp(-E_D(F_{ij}))$ donde $E_D(F_{ij})$ es la distancia Energetica del flujo óptico, se define para cada pixel como:

$$E_D(F_{ij}) = \frac{|F_{ij}|^2}{\beta^2} \quad (3.2)$$

Donde $|F_{ij}|^2$ es la distancia del vector de flujo F_{ij} y el parámetro β controla la varianza de la distancia de la energía, de esta manera se favorecen vectores de flujo óptico con una energía menor, por lo tanto movimientos cortos.

3.5.3 Factor de vecindad

$\phi_N : \phi_N(F_{ij}) = \exp(-E_N(F_{ij}, Fn_{ij}))$ donde $E_N(F_{ij}, Fn_{ij})$ es la energía en la vecindad definida en cada par de pixels adyacentes $(p1_{ij}, p2_{ij}) \in E$:

$$E_N(F_{ij}, Fn_{ij}) = \frac{|F_{ij} - Fn_{ij}|^2}{\lambda^2} \quad (3.3)$$

Donde $|F_{ij} - Fn_{ij}|^2$ es la distancia del vector de diferencia F_{ij} y Fn_{ij} y λ , cuando la energía de las vecindades es similares, la energía resultante es baja, permitiendo a los pixeles de un mismo objeto moverse juntos y suavizando transiciones de flujo en contornos.

3.5.4 Energía total

La energía total del modelo probabilístico $E(\mathbf{F})$ se define como:

$$\mathbf{E}(\mathbf{F}) = \sum_{(i,j) \in \mathbf{E}} E_I(F_{ij}) + \sum_{(i,j) \in \mathbf{E}} E_D(F_{ij}) + \sum_{(i,j) \in \mathbf{E}} E_N(F_{ij}, Fn_{ij}) \quad (3.4)$$

$$\mathbf{E}(\mathbf{F}) = \sum_{(i,j) \in \mathbf{E}} \frac{(I(p_{ij}) - I'(p_{ij} + F_{ij}))^2}{\alpha^2} + \sum_{(i,j) \in \mathbf{E}} \frac{|F_{ij}|^2}{\beta^2} + \sum_{(i,j) \in \mathbf{E}} \frac{|F_{ij} - Fn_{ij}|^2}{\lambda^2} \quad (3.5)$$

A energías bajas mayor certidumbre del flujo óptico del elemento evaluado.

3.5.5 Minimización de la función de energía.

El uso de una función de energía o función costo requiere minimizar el error o energía resultante, por lo tanto cuando la energía es mínima, la certidumbre es alta sobre la validez del flujo óptico. Si solo se encuentran energías altas o errores grandes el flujo óptico se puede considerar inválido.

En dos imágenes consecutivas, si estas son capturadas con un intervalo de tiempo pequeño considerando una velocidad constante, se asume que el flujo óptico en un pixel de manera general se encuentra localizado en una subárea alrededor de la posición de dicho pixel en la primera imagen, esta subarea puede cambiar de tamaño dependiendo del movimiento de la escena y del movimiento del objeto donde el elemento comparado se encuentre en la imagen, se asume por tal motivo una búsqueda de un mínimo local.

Si existen oclusiones por ejemplo, aún con una búsqueda global de energías en la imagen, el error se espera sea grande en todo momento.

Se selecciona el método de Gradiente Descendiente Estocástico considerando lo anterior para minimizar la función de energía.

En el presente trabajo el método de gradiente descendiente estocástico [1] utiliza únicamente las derivativas del factor de intensidad Eq. 3.1, y del factor de distancia Eq. 3.2 disminuyendo el orden del campo aleatorio de markov [26] como una estrategia para simplificar la minimización de la función costo, obteniendo como resultado una intensidad y una dirección en el eje X y en el eje Y en las cuales se busca iterar buscando minimizar el error o energía obtenidos en la función costo, hasta alcanzar un criterio de paro.

Las derivativas se muestran en las Eqs. 3.6,3.7 y en las Eqs. 3.8,3.9:

$$\frac{dFI}{du} = \frac{(-2C + 2(x + u))}{\lambda^2} \quad (3.6)$$

$$\frac{dFI}{dv} = \frac{(-2C + 2(y + v))}{\lambda^2} \quad (3.7)$$

$$\frac{dFD}{du} = \frac{(4u^3 + 4uv^2)}{\beta^2} \quad (3.8)$$

$$\frac{dFD}{dv} = \frac{(4v^3 + 4vu^2)}{\beta^2} \quad (3.9)$$

Por cada correspondencia de los puntos de interés de la primera imagen a la segunda, se crea un ciclo de enfriamiento para la función de energía 3.4 donde el criterio de paro es lo que se cumpla primero, un error menor a $e \leq 0,0001$ ó la cantidad de ciclos de enfriamiento límite de 70, en donde la función de energía converge se considera una correspondencia encontrada. Los valores mostrados se eligieron por medio de una heurística.

3.5.6 Correspondencia de puntos clave.

El algoritmo para corresponder los puntos usando la función costo y minimizandola usando gradiente descendente estocástico se muestra en el Algoritmo 1.

Algoritmo 1 Correspondencia de puntos

```
/* Iniciar Variables */
1: Iniciar todos los pixels  $Energy \leftarrow \infty$ 
2: Iniciar para todo pixel x,y  $optical\ flow \leftarrow 0$ 
3: Iniciar  $error \leftarrow 0,0001$ 
4: Iniciar  $epochslimit \leftarrow 100$ 
5: Iniciar  $\alpha \leftarrow 0,99$ 
6: Iniciar  $\beta \leftarrow 0,9$ 
7: Iniciar  $\lambda \leftarrow 0,9$ 
8: Seleccionar primera Imagen
9: Seleccionar segunda Imagen
10: /* Optional */ Utilizar una mascara para procesar pixeles de interes.
11: for cada pixel seleccionado do Minimizar Función Costo
12:     for 1 hasta  $epochslimit$  do Obtener gradientes en ejes X y Y.
13:         Calcular Gradiente Descendiente Estocástico.
14:         Registrar nuevo valor  $optical\ flow$ .
15:         Calcular la energía de cada pixel.
16:         if Energía de Pixel <  $error$  then
17:             Continuar con el siguiente pixel
18:         end if
19:     end for
20: end for
```

3.6. Triangulación y cálculo de la posición de la cámara.

La triangulación de puntos 3D está basada en un proceso documentado en [34], donde las correspondencias de dos imágenes consecutivas en una escena con baja planaridad y con los intrínsecos de la cámara disponibles son usados para obtener la posición de las cámaras y generar una matriz esencial la cuál es usada para triangular puntos previamente de distorsionados en una nube de puntos 3D, la nube de puntos se representa en escala debido a la incertidumbre en la línea base del movimiento de la cámara.

El proceso de Triangulación consiste en los siguientes pasos:

- a) Generar por medio de los parámetros intrínsecos de la cámara la matrix esencial e indexar puntos inusuales de todos los puntos mediante la función *findEssentialMat* de OpenCV.
- b) Generar la matriz de rotación y traslación de la cámara en la segunda imagen mediante la función *recoverPose* de OpenCV
- c) Corregir la distorsión de los puntos que convergieron mediante la función *undistort* de OpenCV.
- d) Triangular las correspondencias en el espacio.

La triangulación se realiza seleccionando 4 puntos sobre una base *affine* y despues calculando los puntos restantes sobre la misma base mediante *Singular Value Decomposition* [35].

Los puntos se almacenan en formato PLY para poderlo visualizar mediante el programa Blender [36].

4. RESULTADOS

4.0.1 Software y hardware utilizado

El experimento se lleva a cabo en una Macbook con un procesador de 1.1 Ghz Intel Core M3 y 8 GB de memoria RAM. El proceso se programa en C++ y se utiliza OpenCV para la captura de las imágenes y para el almacenamiento de las mismas. La triangulación se realiza con funciones y procedimientos disponibles en OpenCV.

Se toman dos imágenes consecutivas, mediante la cámara web de la computadora Macbook, en este ejemplo las de la Fig. 3.2 y la Fig. 3.3 de 480x720 píxeles. En la Fig.



Figura 4.1: Cuadro clave en gris.

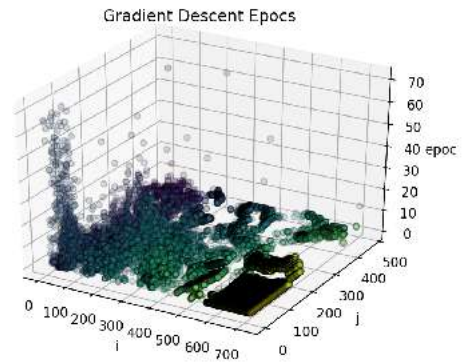


Figura 4.2: Ciclos de enfriamiento Gradiente Descendiente imagen completa

4.1 se muestra una imagen preprocesada y convertida a todos de gris en la que se realiza la minimización de la función de energía.

En la Fig. 3.3 se muestran la cantidad de ciclos de enfriamiento por cada correspondencia seleccionada que logró minimizar su temperatura.

En la Fig. 4.3 se muestra una imagen preprocesada con el algoritmo *canny edge detection* [28]. Solo los píxeles en blanco de la Fig. 4.3 son entrada de la función de energía 3.4 y en la Fig. 4.4 se muestran la cantidad de ciclos de enfriamiento por cada correspondencia seleccionada que logró minimizar su temperatura.

Se encuentran las correspondencias minimizando la función de energía Eq. 3.5 tanto en el eje x como en el eje y para el factor de intensidad y factor de distancia en la función de energía Eqs. 3.1, 3.2 para obtener el flujo óptico de dos imágenes en donde se converge.

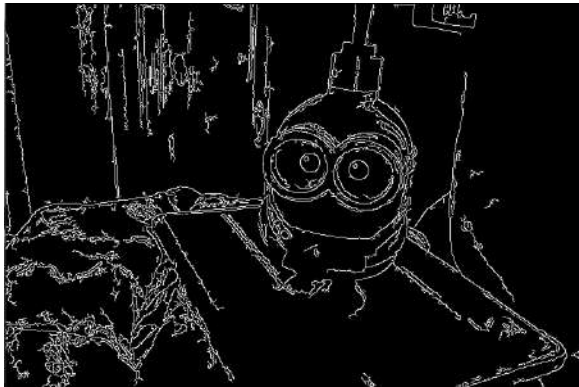


Figura 4.3: Cuadro Clave Preprocesado con el algoritmo *canny edge detection*.

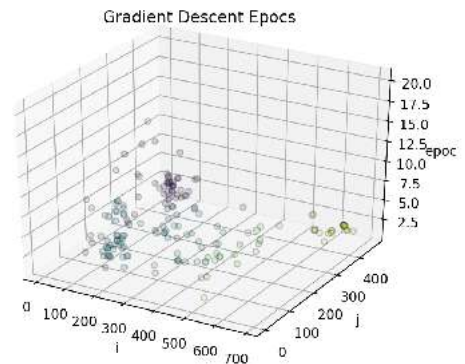


Figura 4.4: ciclos de enfriamiento Gradiente Descendiente imagen preprocesada con el algoritmo *canny edge detection*.

Después del proceso convergen 80412 en 47.9282 segundos puntos de los cuales se obtiene la siguiente triangulación de la escena:

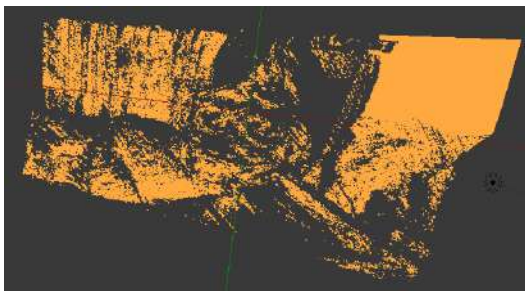


Figura 4.5: Nube 3D Figura 3.2 y 3.3 vista 1.

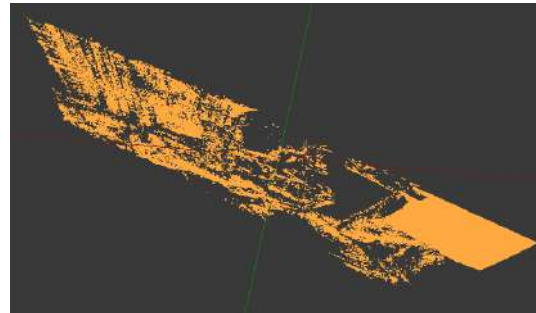


Figura 4.6: Nube 3D Figura 3.2 y 3.3 vista 2.

Si en el proceso de triangulación solo se toman en cuenta los puntos correctos que arroja el proceso de construcción de la matriz esencial, los puntos triangulados se reducen a 58435, el resultado es el siguiente:

Se realiza el proceso incluyendo un preprocesamiento adicional, detectando bordes con el método *canny edge detection* y solo se calculan aquellos pixeles que correspondan a la máscara generada por la detección de bordes sobre la imagen clave. El tiempo de procesamiento se reduce a 4.14785 segundos y los puntos que convergen son solamente 386.

Se toman dos imágenes consecutivas de una escena de una cocina, mediante la cámara web de la computadora Macbook, en este ejemplo las de la Fig. 4.11 y la Fig. 4.12 de 480x720 pixeles.

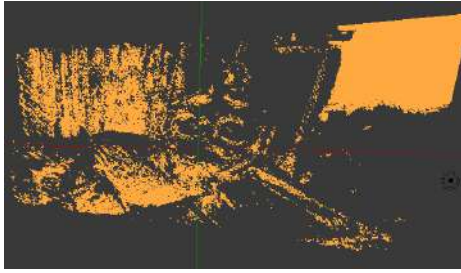


Figura 4.7: Nube 3D Figuras 3.2 y 3.3 puntos correctos vista 1.

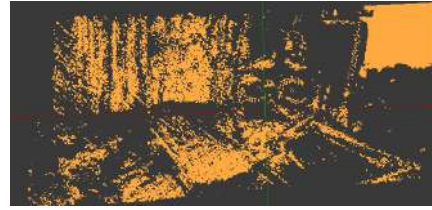


Figura 4.8: Nube 3D Figuras 3.2 y 3.3 puntos correctos vista 2.

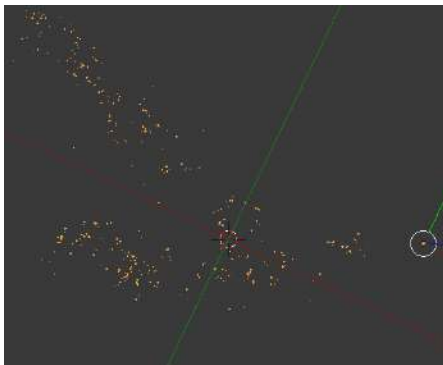


Figura 4.9: Nube 3D Figuras 3.2 y 3.3 aplicando *canny edge detection* vista 1.

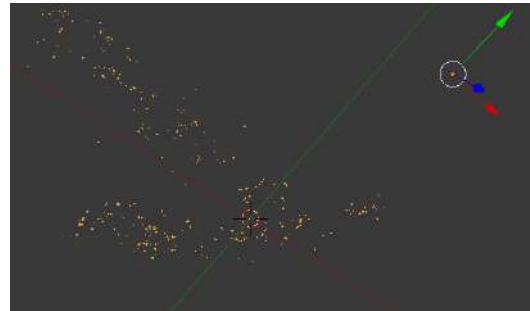


Figura 4.10: Nube 3D Figuras 3.2 y 3.3 *canny edge detection* vista 2.



Figura 4.11: Cocina 1.



Figura 4.12: Cocina 2.

La escena muestra objetos con distinta profundidad, existen planos como las puertas de la cocina integral, sin embargo, son negras y no convergen o casi no convergen los pixeles que las componen como muestra en las Figs 4.11, 4.12, 4.13, 4.14

Se toman dos imagenes consecutivas de una escena de un camino, mediante la camara de un celular, en este ejemplo las de la Fig. 4.11 y la Fig. 4.12 de 1536x2048 pixeles, se generan 1312633 puntos y el tiempo de convergencia es de 363.361 segundos como se muestra en las Figs 4.17 y 4.18.

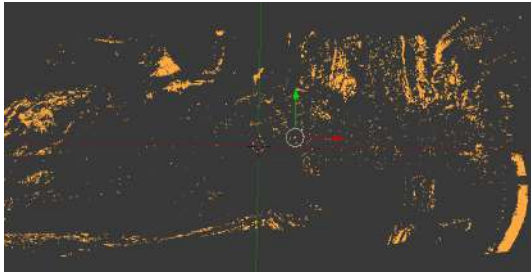


Figura 4.13: Cocina nube de puntos 1.

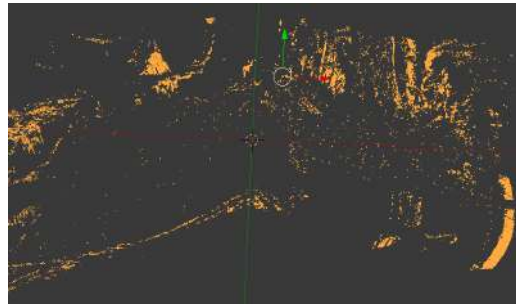


Figura 4.14: Cocina nube de puntos 2.



Figura 4.15: Camino 1.



Figura 4.16: Camino 2.

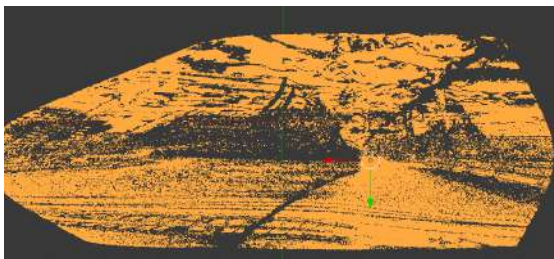


Figura 4.17: Nube 3D Figuras 4.15 y 4.16 vista 1.

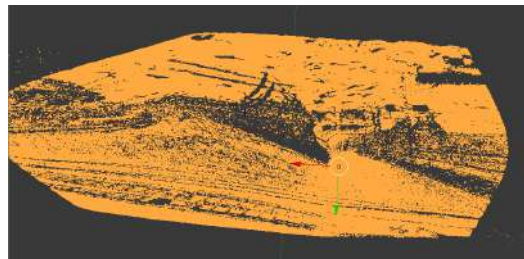


Figura 4.18: Nube 3D Figuras 4.15 y 4.16 vista 2.

4.0.2 Validación del gradiente descendiente estocástico

En el proceso del gradiente descendiente estocástico se utiliza una heurística para definir el límite de *epochs*, en donde 100 iteraciones muestran un balance aceptable de la cantidad de puntos que convergen exitosamente contra los que no. En las siguientes figuras 4.19, 4.20, 4.21, 4.22, se muestran los epochs para todos los píxeles (i, j) donde el Gradiente Descendiente Estocástico converge.

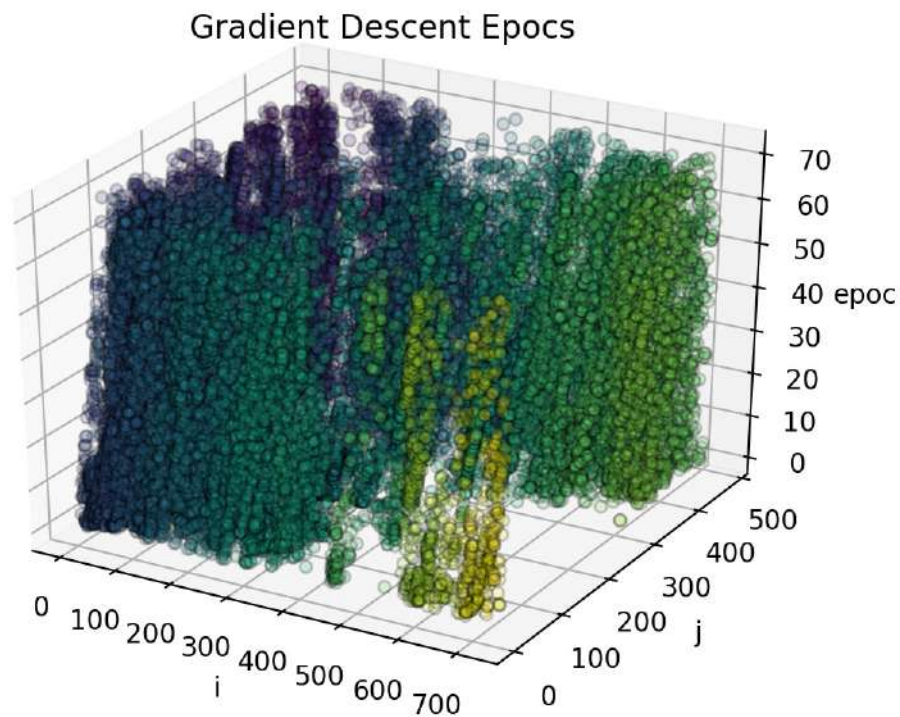


Figura 4.19: SGD epochs Laptop webcam

En el ejemplo que se muestra en la “Fig. 4.23” un valor de flujo óptico es obtenido, gradientes en el eje X y en el eje Y cambian abruptamente durante el proceso de valores positivos a negativos de pendiente hasta que el método converge en la iteración 191.

La intensidad del pixel candidato al flujo óptico en cada iteración se muestra en la “Fig. 4.24” cruzando el valor esperado en varias ocasiones, este comportamiento es esperado debido a que la función de energía en realidad no converge en estos cruces sino cuando el Modelo Aleatorio de Markov disminuye su energía considerando las diferencias en las vecindades y en el factor distancia además de la intensidad del pixel que es donde vemos cruces. Únicamente cuando todos los factores suman un error mínimo la convergencia se alcanza [33].

La posición de los píxeles candidatos al flujo óptico también cambia como se muestra en la “Fig. 4.25”, se observa como el Gradiente Descendiente Estocástico explora un área

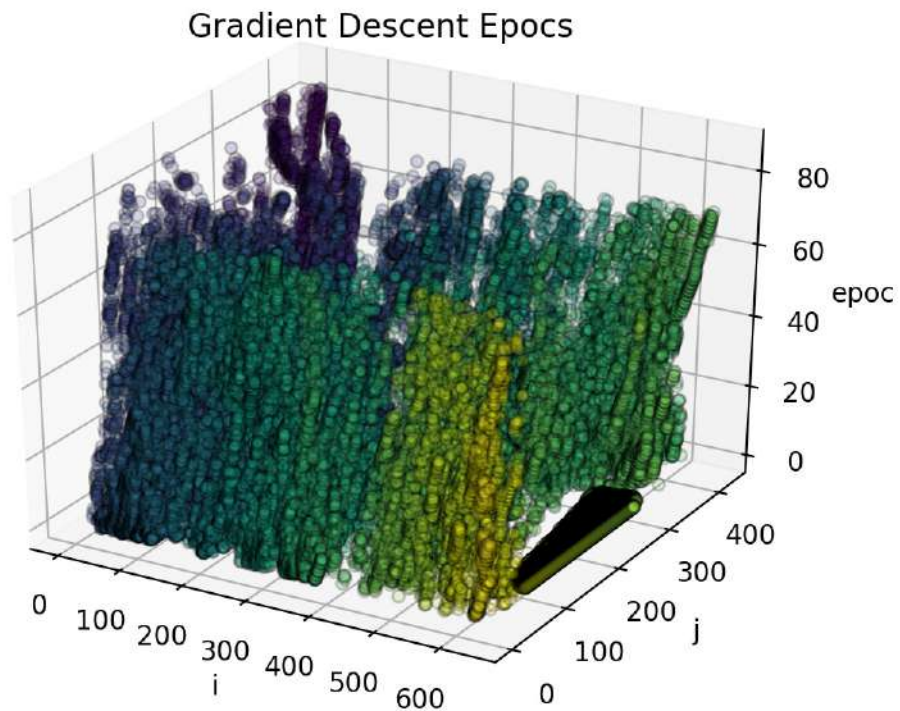


Figura 4.20: SGD epochs webcam c170

localizada a la posición original del pixel de la primera imagen en el área de la segunda imagen hasta que la condición de convergencia es alcanzada o un valor limite de iteraciones se agota. Este comportamiento es deseado y soporta diferentes valores de flujo óptico para diferentes objetos en una escena de una imagen a otra.

Mientras más grande el límite de iteraciones mayor área permitida a explorar en la búsqueda de valores de flujo óptico válido sin embargo el costo computacional aumenta de manera acumulada en la imagen completa. El valor inicial del flujo óptico se configura en cero, debido a que se asume que el movimiento de la escena es pequeño y/o en muchos casos casi nulo o nulo debido a elementos estáticos en la escena. Se asumen movimientos pequeños los cuales arrojan un mayor radio de convergencia, los pesos muestran un buen comportamiento iniciando en un valor de 3.0.

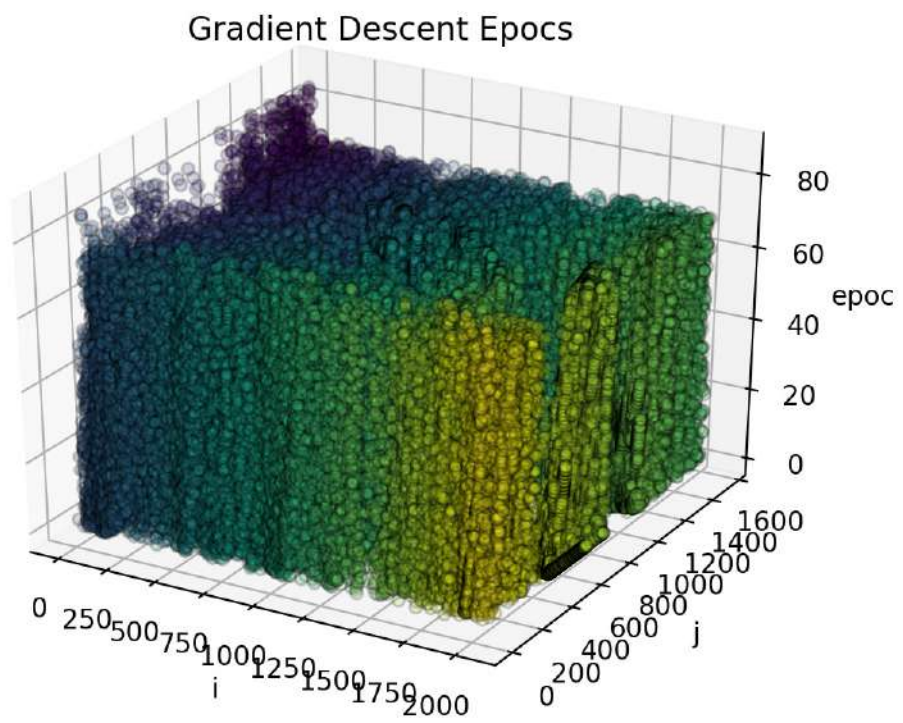


Figura 4.21: SGD epochs Mobile camera

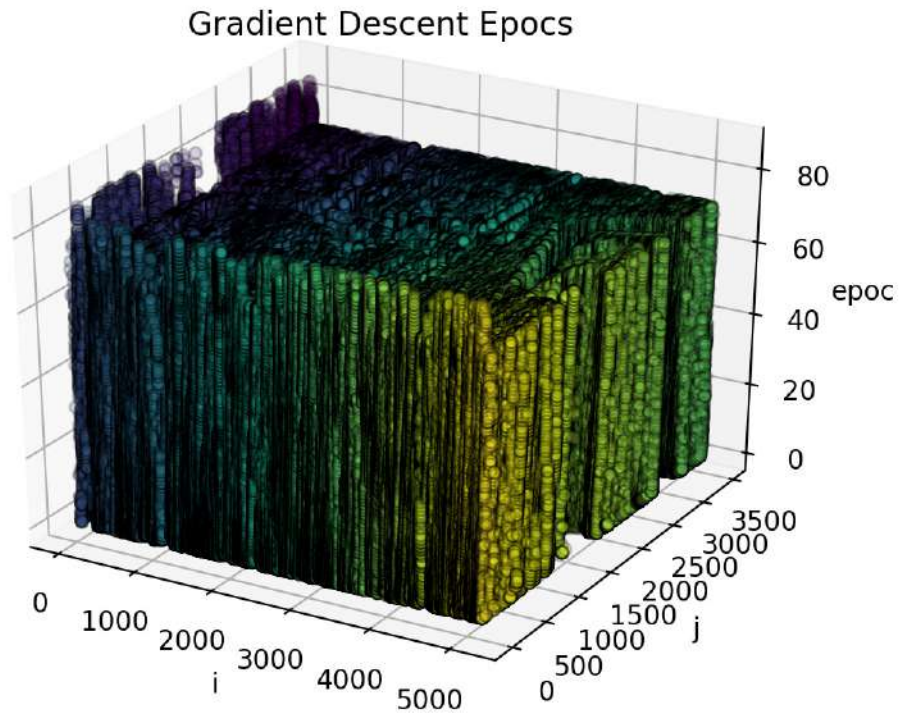


Figura 4.22: SGD epochs Canon T3i camera

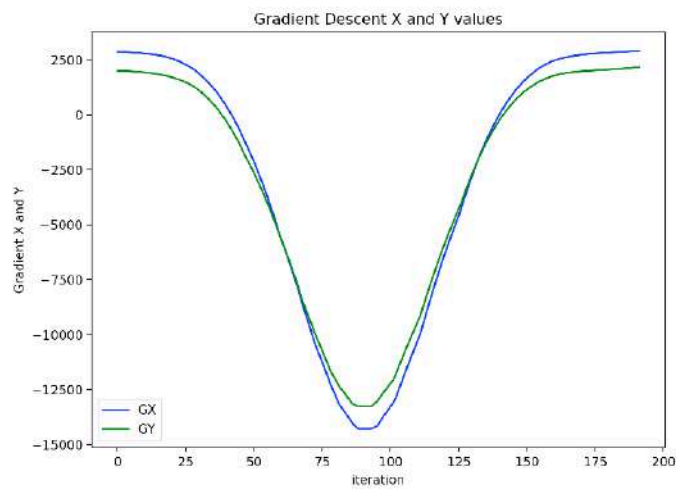


Figura 4.23: Slope values from a pixel SGD process.

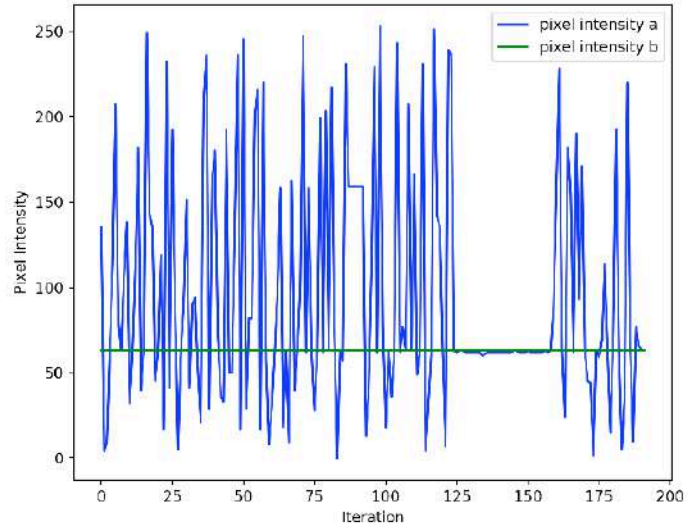


Figure 4.24: Pixel Intensity variation during GD.

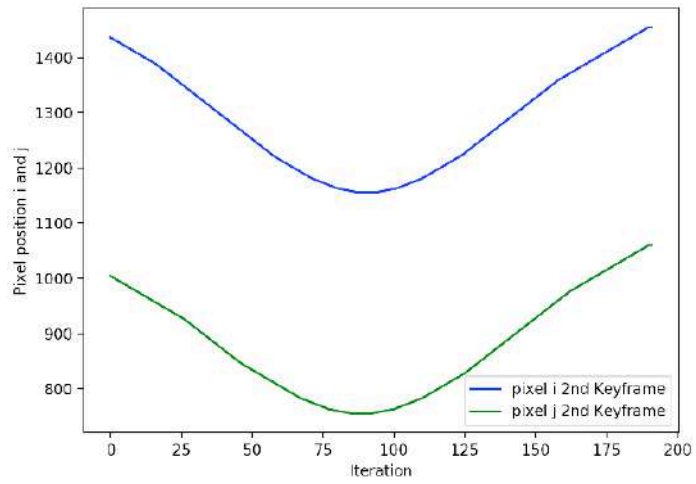


Figure 4.25: Pixel position variation during GD.

Tabla 4.1: Error y valores de Gradiente Descendiente Estocástico

Error	pixel i	pixel j	pixel b	GX	GY
⋮	⋮	⋮	⋮	⋮	⋮
0.126738	1227	828	62	-4586.95	-4275.91
0.123357	1231	832	63	-4206.88	-3968.04
0.19955	1235	836	63	-3810.39	-3613.25
0.0564706	1239	840	62	-3429.33	-3273.02
0.0942802	1243	844	62	-3063.39	-2947.06
0.0438666	1247	848	62	-2712.27	-2635.05
0.131402	1251	852	62	-2375.64	-2336.67
0.0247729	1255	856	62	-2053.2	-2051.61
0.0589751	1259	860	62	-1744.62	-1779.56
0.065421	1263	864	60	-1449.6	-1520.21
0.0829703	1267	868	61	-1167.83	-1273.23
0.0302587	1271	872	62	-898.98	-1038.33
0.0224551	1275	876	62	-642.749	-815.174
0.0703246	1279	880	62	-398.821	-603.46
0.0250124	1283	884	62	-166.881	-402.874
0.170653	1287	888	62	53.3829	-213.102
0.101031	1291	892	62	262.285	-33.8296
0.0972307	1295	896	62	460.138	135.256
0.178283	1299	900	62	647.256	294.468
0.118486	1303	904	62	823.952	444.121
0.0425441	1307	908	63	990.54	584.527
0.152675	1311	912	63	1147.33	716
0.118484	1315	916	62	1294.64	838.854
0.140419	1319	920	62	1432.79	953.401
0.14222	1323	924	62	1562.08	1059.96
0.0605921	1327	928	63	1682.82	1158.83
0.142473	1331	932	62	1795.34	1250.34
0.0459845	1335	936	62	1899.95	1334.8
0.0453013	1339	940	62	1996.96	1412.52
0.0571736	1343	944	62	2086.67	1483.81
0.150516	1347	948	62	2169.42	1548.99
0.0194914	1351	952	63	2245.5	1608.37
0.0855715	1355	956	62	2315.23	1662.27
⋮	⋮	⋮	⋮	⋮	⋮

Tabla 4.2: Error y valores de Gradiente Descendiente Estocástico

Error	pixel i	pixel j	pixel b	GX	GY
⋮	⋮	⋮	⋮	⋮	⋮
0.0385896	1359	960	63	2378.94	1711
0.0238919	1362	964	88	2436.92	1754.86
0.0432361	1365	968	158	2479.57	1791.5
0.288766	1368	972	228	2518.68	1824.63
0.0168067	1371	976	63	2554.43	1854.49
0.00308672	1374	979	24	2587	1881.34
0.0913165	1377	982	182	2615.12	1900.76
0.0493524	1380	985	155	2640.92	1918.4
0.174355	1383	988	62	2664.53	1934.4
0.196674	1386	991	190	2686.08	1948.88
0.00855275	1389	994	93	2705.7	1961.99
0.0716889	1392	997	171	2723.54	1973.84
0.0333718	1395	1000	63	2739.71	1984.58
0.0950052	1398	1003	45	2754.35	1994.33
0.0751557	1401	1006	44	2767.6	2003.23
0.0625891	1404	1009	1	2779.59	2011.41
0.0233493	1407	1012	63	2790.44	2019
0.106704	1410	1015	59	2800.3	2026.14
0.0147416	1413	1018	69	2809.29	2032.95
0.0314143	1416	1021	114	2817.54	2039.58
0.0040424	1419	1024	63	2825.19	2046.14
0.252958	1422	1027	15	2832.38	2052.78
0.0145746	1425	1030	73	2839.22	2059.62
0.249155	1428	1033	192	2845.86	2066.81
0.014713	1431	1036	63	2852.42	2074.46
0.508817	1434	1039	5	2859.05	2082.71
0.122568	1437	1042	36	2865.86	2091.7
0.22693	1440	1045	220	2873	2101.56
0.000184544	1443	1048	63	2880.59	2112.41
0.20825	1446	1051	10	2888.77	2124.4
0.00936123	1449	1054	77	2897.67	2137.65
0.0867732	1452	1057	66	2907.42	2152.29
0.00005931	1455	1060	63	2918.16	2168.46

Tabla 4.3: Resolución y Puntos sin preprocesamiento

Método	Camara	Resolución	Puntos
MRF	Camara Laptop	480x720	36,204
MRF	Webcam	426x640	82,526
MRF	Camara Móvil	1536x2048	207,969
MRF	Canon T3i	3456x5184	6,585,689
Stereo	Realsense D345	1280x720	296,473

Tabla 4.4: Resolución y Puntos usando CannyEdge

Método	Cámara	Resolución	Puntos
MRF	Camara Laptop	480x720	4,423
MRF	Webcam	426x640	1,828
MRF	Cámara Móvil	1536x2048	33,759
MRF	Canon T3i	3456x5184	29,651

Las Tablas 4.1, 4.2 muestran las últimas 66 iteraciones desde la 125 hasta la 191, se incluyen los valores generados en la búsqueda de un valor de flujo óptico valido para el pixel $i: 1439, j: 1007$ en la primera imagen. La intensidad del pixel de la primera imagen no se muestra debido a que es un valor que no cambia durante el proceso su intensidad es de 63, el número de iteracion no se muestra sin embargo la interacion final es 191.

Las dos imagenes utilizadas tienen una resolución de 1536×2048 , 3145728 pixeles en total cada una, se colocó un límite de iteración de 1000 y un error de 0.0001, solamente 370877 pixeles convergieron esto es 11.789%. Debido a que el radio de convergencia dependen de la composición de las imágenes, el límite de iteraciones, el error establecido y la aleatoriedad del proceso, este porcentaje no es un radio fijo.

4.0.3 Comparación contra camara realsense D435

En la tabla 4.3 se muestra una comparación directa de la cantidad de puntos generados con el método propuesto y la cámara Realsense (disparidad de imagen y proyección de Laser Infrarojo) [37]. En seguida en las tablas 4.4 y 4.5 se muestran resultados usando imágenes preprocesadas aplicando los métodos *canny edge detection* y *SLIC superpixels*.

Tiempo de cómputo. Los experimentos fueron ejecutados en una laptop MacBook con las siguientes características Procesador 1.1Ghz Intel Core M3, 8GB RAM, Intel HD Graphics 515 1536 MB, el método propuesto se implementó utilizando en lenguaje de programación C++ y la librería de OpenCV. El tiempo de creación del archivo PLY no es tomado en cuenta, para obtener una salida PLY a un archivo con la cámara Realsense D435 se utilizó python mediante el SDK Intel Realsense disponible [38], el tiempo de computo se muestra en la tabla 4.6:

Tabla 4.5: Resolución y Puntos usando SLIC superpixel

Método	Camara	Resolución	Puntos
MRF	Cámara Laptop	480x720	2,182
MRF	Webcam	426x640	4,716
MRF	Cámara Móvil	1536x2048	24,097
MRF	Canon T3i	3456x5184	29,651

Tabla 4.6: Resolución y Puntos usando SLIC superpixel

Método	Cámara	Resolución	Tiempo
MRF	Cámara Laptop	480x720	72.5237 seg.
MRF	Webcam	426x640	52.3465 seg.
MRF	Cámara Móvil	1536x2048	1001.14
MRF	Canon T3i	3456x5184	3148.91 seg.
Stereo	Realsense D345	1280x720	RealTime

4.0.4 Nubes de puntos 3D

En las siguientes figuras 4.26, 4.27, 4.28 se utilizó la cámara integrada de una laptop para obtener las imágenes para generar la nube de puntos en 3D.

En las figuras. 4.29, 4.30, 4.31 , se utilizó la cámara web USB Logitech C120 para obtener las imágenes para generar la nube de puntos en 3D.

En la figura. 4.32, se utilizó la cámara integrada de un celular Iphone XS max para obtener las imágenes para generar la nube de puntos en 3D.

En la figura. 4.33, se observa la nube de puntos 3D generada por la cámara stereo Intel Realsense D435.

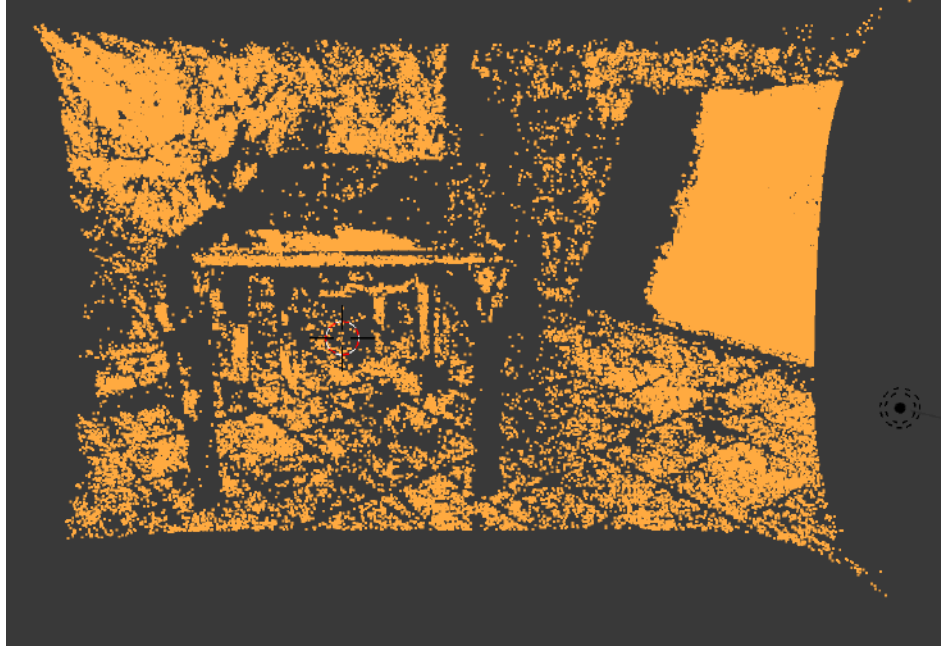


Figura 4.26: Salida PLY, Cámara Laptop.

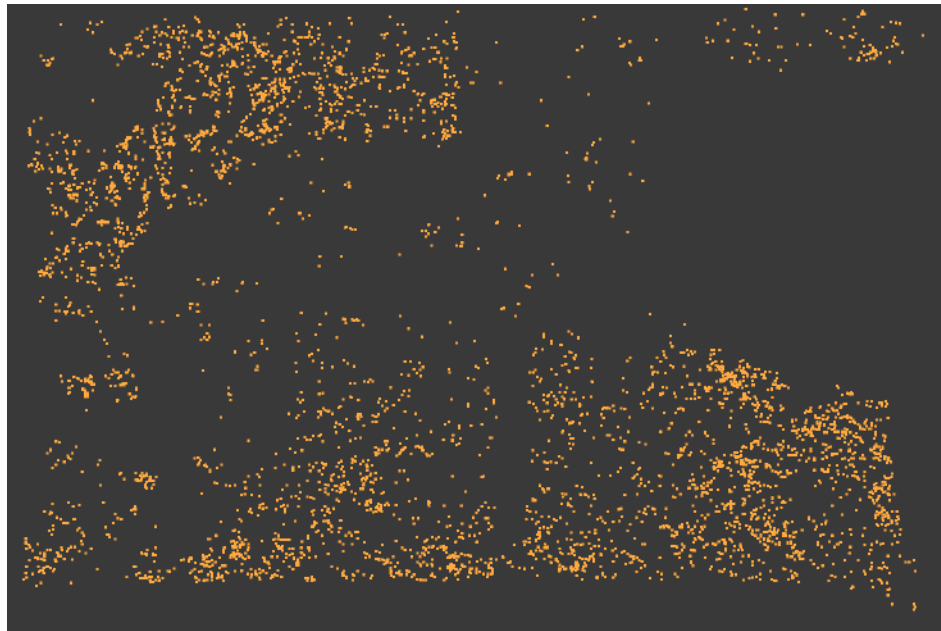


Figura 4.27: Salida PLY, Cámara Laptop usando CannyEdge.

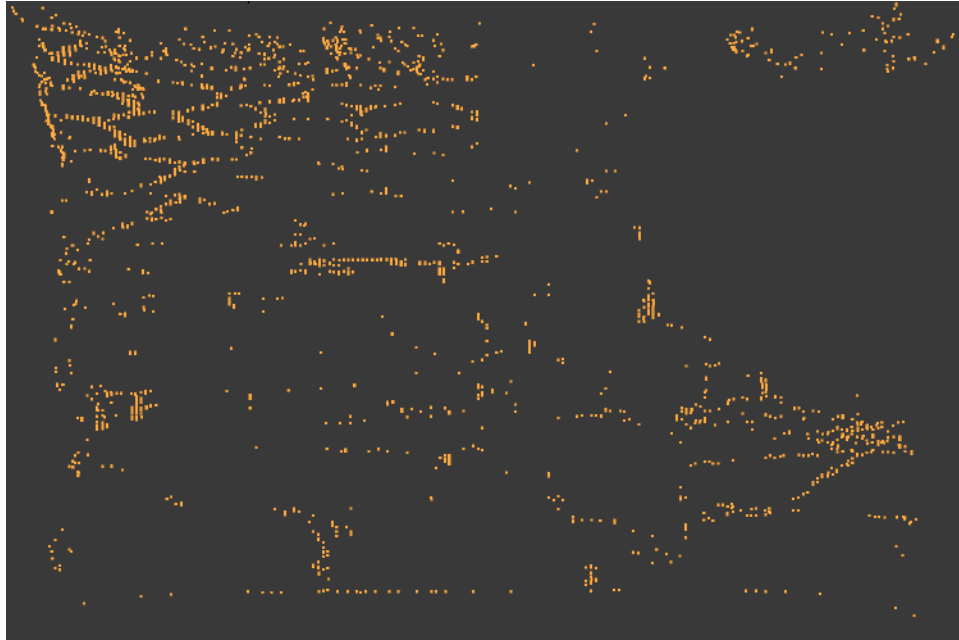


Figura 4.28: Salida PLY, Cámara Laptop usando SLIC superpixel.

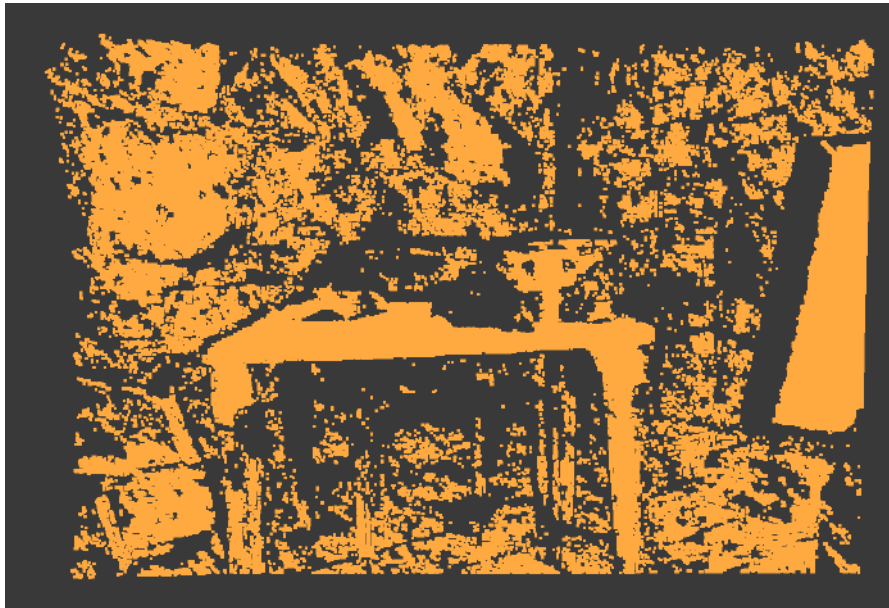


Figura 4.29: Salida PLY, Cámara Web USB.

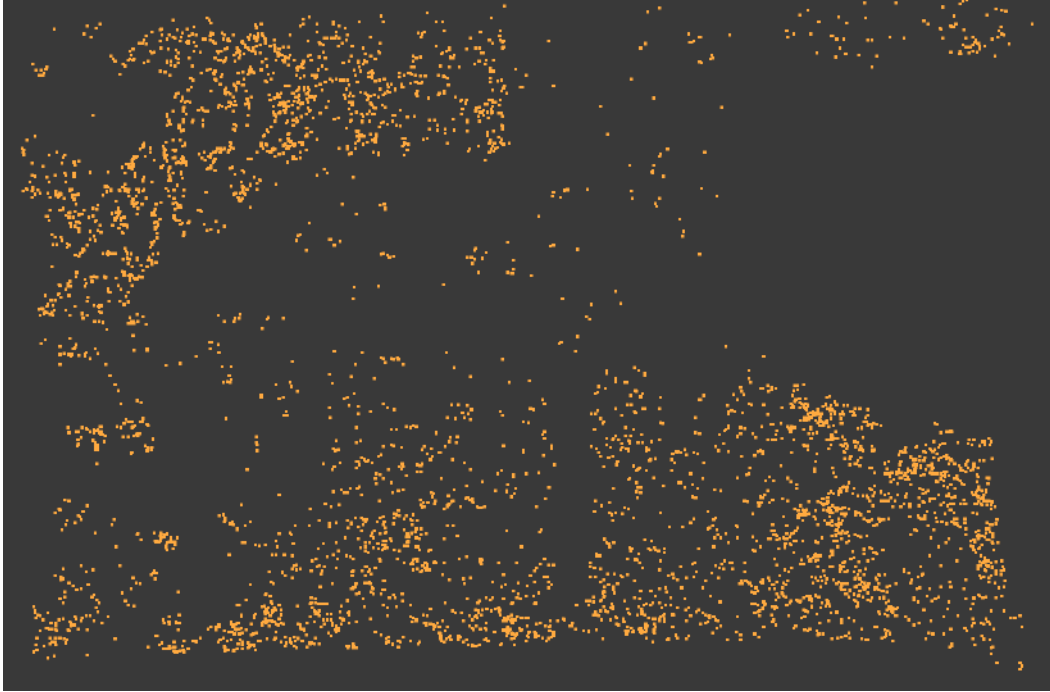


Figura 4.30: Salida PLY, Cámara Web USB usando Canny Edge .

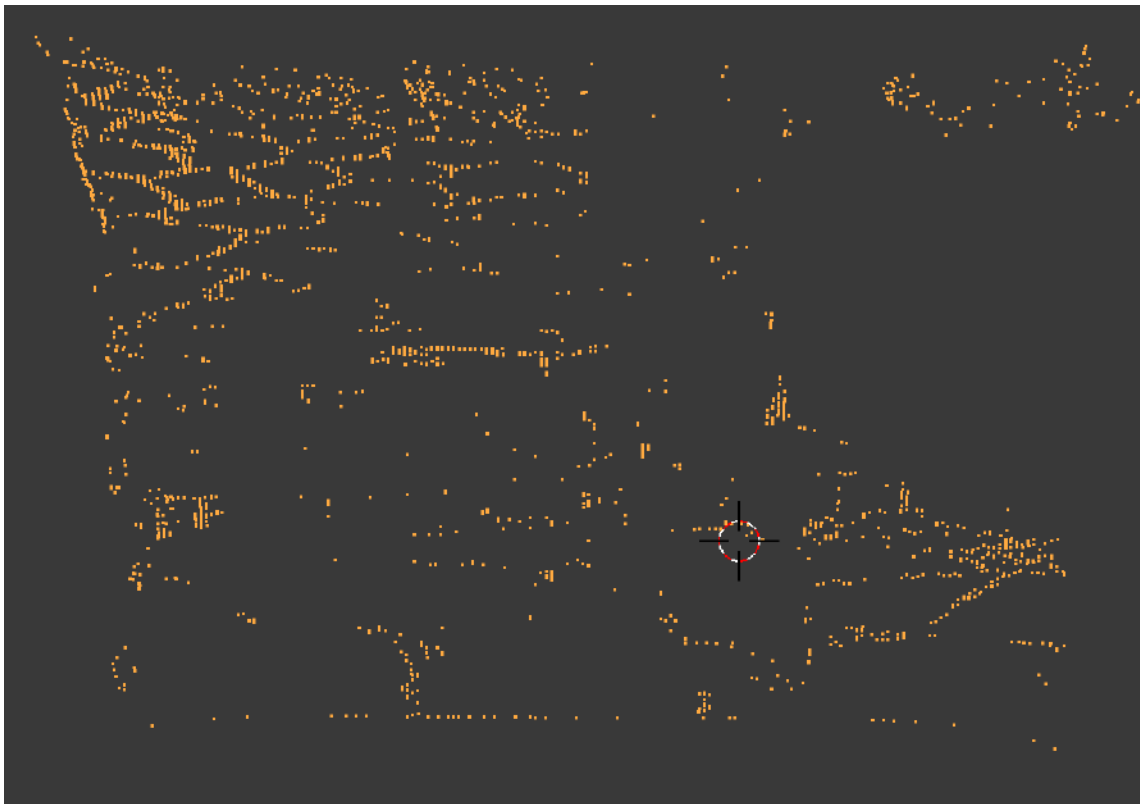


Figura 4.31: Salida PLY, Cámara Web USB usando SLIC superpixel.



Figura 4.32: Salida PLY Cámara Integrada Celular.



Figura 4.33: Salida PLY Cámara Intel Realsense D435.

5. CONCLUSIONES Y TRABAJOS FUTUROS

La etapa de inicialización de un proceso SLAM actualmente tiene áreas de oportunidad de mejora en la robustez en escenas dinámicas y con cambios de iluminación, de igual manera hay posibilidad de mejora al buscar operar sin asumir características de una escena inicial [2].

1. Resultados iniciales muestran que el tiempo de computo puede mejorar con desplazamientos de correspondencias menores debido a que la minimización es local y no global.
2. El factor de distancia en el Modelo de Campo Aleatorio de Markov y el límite de iteraciones definen el área de búsqueda para minimizar la función costo.
3. Las áreas regulares con pixeles muy similares en intensidad tienden a converger sin embargo el valor de flujo óptico no es lo suficiente preciso.
4. Los pixeles que no convergen usualmente es debido a que se encuentran en oclusiones en la segunda imagen, la textura es regular, o cuando el movimiento del objeto que contiene el pixel es más grande que el área de búsqueda del Gradiente Descendente Estocástico.
5. La función de energía Eq. 3.4 basada en [33] permite obtener correspondencias entre dos imágenes consecutivas de manera aceptable bajo tolerancias a cambios de iluminación y deformaciones de objetos, observese la diferencia de iluminación en las figuras 3.2 y 3.3 de las cuales fue posible generar correspondencias y triangular una nube de puntos 3D Fig. 4.9.
6. El método de Gradiente Descendente es útil para minimizar este tipo de funciones energéticas debido a la naturaleza localizada del problema donde no se requiere una solución global de minimización.
7. El procedimiento propuesto es susceptible a mejorarse por medio de etapas de preprocesamiento, la nube generada cuando se preprocesa la imagen con detección de bordes es mucho menos densa 386 puntos contra 80412 sin preprocesamiento, esta característica se puede utilizar para generar una escena en un proceso SLAM, al generar la escena e ir agregando puntos la escena puede ir creciendo en puntos acorde a necesidades variadas de densidad.
8. Se observa una mejor visualización de los puntos en escenas donde hay un pocos planos, esto es debido a que se utiliza la matriz esencial la cuál asume una escena no plana y con elementos en perspectiva en diferentes distancias.
9. La correspondencia de los puntos se puede utilizar de la misma manera en etapas posteriores de un proceso SLAM.

BIBLIOGRAFÍA

- [1] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [2] Georges Younes, Daniel Asmar, Elie Shamma, and John Zelek. Keyframe-based monocular slam: design, survey, and future directions. *Robotics and Autonomous Systems*, 98:67–88, 2017.
- [3] Scott Dadich. The president in conversation with mit’s joi ito and wired’s scott dadich. <https://www.wired.com/2016/10/president-obama-mit-joi-ito-interview/>, 2016. Accesado: 2018-08-14.
- [4] Steve G Sutton, Matthew Holt, and Vicky Arnold. “the reports of my death are greatly exaggerated”—artificial intelligence research in accounting. *International Journal of Accounting Information Systems*, 22:60–73, 2016.
- [5] A. Carretón. ¿cuál es el origen del odómetro? <http://queaprendemos hoy.com/cual-es-el-origen-del-odometro/>, 2014. Accesado: 2018-08-14.
- [6] N. Haq. Introduction to photogrammetry. <http://queaprendemos hoy.com/cual-es-el-origen-del-odometro/>, 2013. Accesado: 2018-08-14.
- [7] César Téllez Zamora, Daniel Cisneros Muñoz, Jesús Carlos Pedraza Ortega, Saúl Tovar Arriaga, Efrén Gorrostieta Hurtado, and Sandra Luz Canchola Magdaleno. Sistema para digitalización y modelado 3d de objetos, mediante proyección laser utilizando hardware de arquitectura abierta. In *de IX Congreso Internacional sobre Innovación y Desarrollo Tecnológico, Cuernavaca, México*, 2011.
- [8] FARO® GAGE. Faro technologies. <https://www.faro.com/es-mx/productos/factory-metrology/faro-gage/>, 2017. Accesado: 2018-08-14.
- [9] S. Monge. fmri – resonancia magnética funcional. <http://neuromarca.com/neuromarketing/fmri/>, 2009. Accesado: 2018-08-14.
- [10] Ruben Gomez-Ojeda, David Zuñiga-Noël, Francisco-Angel Moreno, Davide Scaramuzza, and Javier Gonzalez-Jimenez. Pl-slam: a stereo slam system through the combination of points and line segments. *arXiv preprint arXiv:1705.09479*, 2017.

- [11] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.
- [12] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion*. *Acta Numerica*, 26:305–364, 2017.
- [13] H Durrant Whyte. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.
- [14] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2018.
- [15] Stan Z Li. Markov random field modeling in image analysis.
- [16] Silvio Giancola, Matteo Valenti, and Remo Sala. *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscapy Technologies*. Springer, 2018.
- [17] Michael Kosta Loukides and Ben Lorica. What is artificial intelligence? 2016.
- [18] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.
- [19] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.
- [20] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: an evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018.
- [21] Murong Wang, Xiabi Liu, Yixuan Gao, Xiao Ma, and Nouman Q Soomro. Superpixel segmentation: a benchmark. *Signal Processing: Image Communication*, 56:28–39, 2017.
- [22] Stefan Roth, Victor Lempitsky, and Carsten Rother. Discrete-continuous optimization for optical flow estimation. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 1–22. Springer, 2009.
- [23] Chen Liu, Hang Yan, Pushmeet Kohli, and Yasutaka Furukawa. Multi-way particle swarm fusion. *arXiv preprint arXiv:1612.01234*, 2016.
- [24] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2609–2616. IEEE, 2014.
- [25] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.


- [26] Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. *Computer Vision and Image Understanding*, 117(11):1610–1627, 2013.
- [27] G. Sims. Arm vs x86 – key differences explained! <https://www.androidauthority.com/arm-vs-x86-key-differences-explained-568718/>, 2014. Accessed: 2018-08-14.
- [28] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.
- [29] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, Sabine Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.
- [30] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.
- [31] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.
- [32] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.
- [33] Dongzhen Piao, Prahlad G Menon, and Ole J Mengshoel. Computing probabilistic optical flow using markov random fields. In *International Symposium Computational Modeling of Objects Represented in Images*, pages 241–247. Springer, 2014.
- [34] Robert Laganière. *OpenCV Computer Vision Application Programming Cookbook Second Edition*. Packt Publishing Ltd, 2014.
- [35] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [36] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam,
- [37] Realsense stereo depth. <https://realsense.intel.com/stereo/>. Accessed: 2018-12-13.
- [38] Realsense stereo depth sdk. <https://software.intel.com/en-us/realsense/sdk>. Accessed: 2018-12-13.


APÉNDICES

0.1. Artículo



Triangulación de puntos usando campos aleatorios de Markov con 2 imágenes consecutivas

Román Rivera Luis Rogelio , Pedraza Ortega Jesús Carlos , Aceves Fernandez Marco Antonio, Gorrostieta Hurtado Efrén, Ramos Arreguín Juan Manuel

Universidad Autónoma de Querétaro. Facultad de Ingeniería.
 r3roger@gmail.com

Resumen

El objetivo de este trabajo es obtener la triangulación de puntos en una nube 3D para ser usada en un proceso SLAM (Simultaneous Localization And Mapping) usando un método directo el cuál utiliza la información a nivel pixel como base en la detección de puntos de interes en imágenes consecutivas en el tiempo capturadas por una sola cámara fotográfica o de video creando al final del proceso una nube de puntos en 3 dimensiones. El diseño y arquitectura del proceso requiere seleccionar entre diferentes opciones basandose en las características deseadas del proceso SLAM como tal, este diseño se basa en [1] donde se proporciona a detalle una guía y un resumen del estado del arte y áreas de oportunidad de investigación y desarrollo para los procesos SLAM. En la etapa inicial, para obtener un cuadro clave se usan campos aleatorios de Markov para calcular una función costo generando como resultado un flujo óptico de dos imágenes consecutivas, la función costo esta basada en [2], la cuál es minimizada por medio de gradiente descendente estocástico [3]. A partir del flujo óptico se calculan las posiciones de la camara y las proyecciones en 3D son trianguladas usando la matriz esencial [4] con los puntos ajustados mediante los parámetros intrínsecos de la cámara utilizada. Los métodos tradicionales no realizan el proceso de correspondencia de puntos como se propone en el presente trabajo.

Palabras clave: campo aleatorio de Markov, gradiente descendente, triangulación

Abstract

The goal of this work is a triangulated 3D cloud points usefull in a SLAM process (Simultaneous Localization And Mapping) via a direct method wich use pixel intensities as a base to detect interest points in consecutive frames in time captured by only one video or picture camera. The design and architecture of the process requires to select from different options based in the desired capabilities of the SLAM process, based in [1] where a detailed survey of the state of the art, a guide about the design of slam processes and its coponents, and oportunity areas are provided. In the initialization stage in order to get a key frame a Markov random field is used to compute a energy fuction wich applied in two consecutive frames generates an optical flow, the energy function is based in [2], this energy function is minimized using Stochastic Gradient Descent [3]. From the optical flow camera positions are computed and 3D proyections are triangulated using the esencial matrix [4], where the adjusted points with the camera intrinsics are used. Traditional methods do not process correspondences as it is proposed in this work.



keywords: Markov random field, gradient descent, triangulation

1. Introducción

El uso de cámaras digitales se ha generalizado, se pueden encontrar cámaras en dispositivos móviles, tabletas. Aprovechar la información generada por estos dispositivos para construir un modelo en 3D de la escena puede aportar información útil para diversos escenarios como navegación, mapeo, reconocimiento de objetos, reconocimiento facial, entre otros.

En un proceso SLAM basado en cuadros claves, la asociación de datos puede ser directa, basado en características y/o en una combinación de ambos. Los métodos directos se caracterizan por explotar la información de todos los píxeles, un método semi-denso solo utiliza la información de un subconjunto de píxeles. Los métodos basados en características se enfocan en regiones con gradientes sobresalientes. Los métodos híbridos combinan los métodos descritos [1].

En la etapa de inicialización de un proceso SLAM se puede utilizar una homografía asumiendo una escena plana, la matriz esencial, asumiendo una escena no plana, ó una asignación de profundidad aleatoria requiriendo procesar cuadros adicionales a fin de converger posteriormente la profundidad. SVO [5] utiliza una homografía, DT-SLAM [6] utiliza matriz esencial ambos métodos asociando características, DSO [7] y LSD-SLAM [8] utilizan valores de profundidad aleatorios. ORB SLAM [9] utiliza homografía ó matriz esencial seleccionando la opción mas adecuada mediante la penalización de un error. Se continua la investigación en métodos robustos en la asociación de los datos (correspondencias) ante cambios de iluminación, escenas y ambientes con dinámicos así como en una inicialización robusta que no requiera asumir una escena inicial [1]. El presente trabajo propone un método para la triangulación de puntos donde la adquisición de imágenes 2D consecutivas en el tiempo utiliza una sola cámara, y el proceso para obtener correspondencias, un método semi-directo utiliza un campo aleatorio de Markov basado en [2], el cuál es minimizado por medio de gradiente descendiente estocástico. El método propuesto es de utilidad en la etapa inicial de un proceso SLAM para la triangulación de una nube de puntos inicial, [3].



Figura 1: Características detectadas en un proceso SLAM [10]



Figura 2: Resultado del proceso SLAM (nube de puntos) [10]

En la Fig. 1 se muestra las características detectadas en la asociación de datos de un



Figura 3: Resultado del proceso SLAM Denso [11]

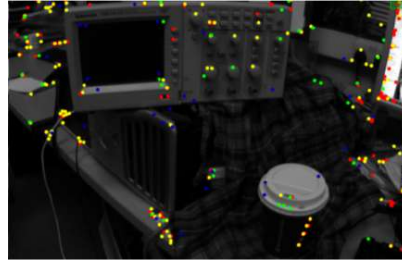


Figura 4: Características detectadas [11]

proceso SLAM, mediante las cuales se triangulan los puntos de la Fig.2 como resultado, en la Fig. 3 se muestra el resultado de la triangulación de una asociación de datos directa densa con algunas correspondencias mostradas en la Fig. 4.

1.1 Métodos y Materiales

Se propone un procedimiento, utilizando el cálculo del flujo óptico a partir de una función basada en campos aleatorios de Markov y minimizada con el método de gradiente descendiente estocástico como se muestra en la Fig. 5 en la etapa de inicialización de un proceso SLAM, etapa donde se requiere tomar una imagen como cuadro clave y generar mediante una segunda imagen, correspondencias, por medio de las cuales se calculan las posiciones de las cámaras en las primera y segunda imagen y se triangulan las correspondencias proyectando una nube de puntos 3D.

1.2 Captura de Imágen

En el presentetrabajo se realiza la captura de las imágenes como se muestra en la Fig. 5 mediante la cámara web integrada de una laptop y mediante la camara integrada de un dispositivo móvil. Las imágenes se almacenan y procesan en una laptop Macbook.

1.3 Conversión a Grises, Preprocesamiento, 1er y 2do Cuadro

Se selecciona el 1er y 2do cuadro Figs. 6 y 7 de las imágenes capturadas, los cuadros pueden ser preprocesados , se convierten a escala de grises, opcionalmente se aplica la detección de bordes por medio del algoritmo *canny edge detection* [12] el cuál permite reducir el tiempo de computo al ciclo de enfriamiento de la función de energía.

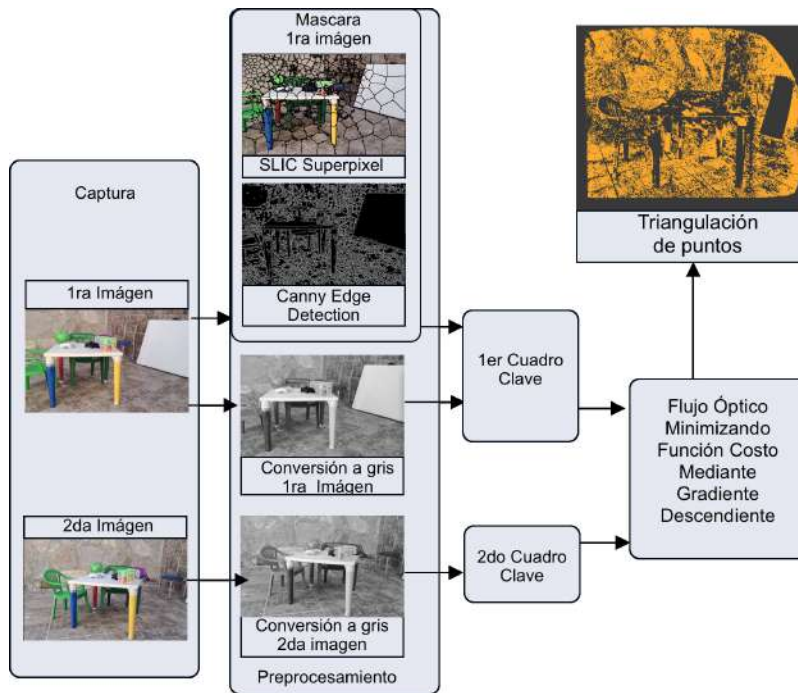


Figura 5: Método propuesto basado en la etapa inicial SLAM [1].



Figura 6: Figura ejemplo cuadro clave.



Figura 7: Figura ejemplo 2do. cuadro



1.4 Estimación del Flujo Óptico

El flujo óptico se refiere al movimiento aparente del campo de visión 2D entre imágenes consecutivas en el tiempo. Es diferente al movimiento de los objetos en la escena, en el caso extremo en el movimiento en el eje de la cámara, no hay flujo óptico, por el contrario rotando la cámara hay flujo óptico.

Existen 2 métodos base para estimar el flujo óptico:

- 1) Lucas - Kanade. El cuál genera un flujo de vectores disperso asumiendo un movimiento local constante y un brillo de cada pixel constante [13].
- 2) Horn Schunck. Genera un campo de flujo denso asumiendo un flujo de campo suave epialmente [14].

Se obtiene el flujo óptico a nivel de pixel a partir de dos imágenes tomadas de una escena de manera consecutiva en el tiempo, se genera una lista de correspondencias de un conjunto de pixeles en la primera imagen a la segunda imagen. Se utilizan las Eqs. 4 y 5 para el computo del flujo óptico a nivel de pixel.

El cálculo del flujo óptico y la triangulación de las correspondencias requieren que la cámara utilizada se calibre, evitando de este tipo de distorsiones antes de procesar las imágenes. Se utilizan un proceso de calibración basado en [15].

1.5 Función Costo ó de Energía

Se utiliza una función de energía ó función costo basada en [2].

Se definen los pixeles correspondientes a una imagen de tamaño $h \times w$ donde p_{ij} corresponde al i ésimo renglón y a la j ésima columna, y el flujo óptico correspondiente a este pixel $P_{ij} F = F_{ij}$, con $i = 1, \dots, h$ y $j = 1, \dots, w$ se definen los siguientes factores en un *Markov Random Field* ó campo aleatorio de Markov:

1.5.1 Factor de Intensidad

$\phi_I : \phi_I(F_{ij}) = \exp(-E_I(F_{ij}))$ donde $E_I(F_{ij})$ es la intensidad de Energía del flujo óptico, se define para cada pixel como:

$$E_I(F_{ij}) = \frac{(I(p_{ij}) - I'(p_{ij} + F_{ij}))^2}{\alpha^2} \quad (1)$$

Donde I es la intensidad de pixel en la primer imagen e I' es la intensidad de pixel en la segunda imagen, el parámetro α es la varianza en la intensidad de la energía para controlar cuanto se compara el movimiento con la misma intensidad con movimiento con cambio en intensidad.

1.5.2 Factor de Distancia

$\phi_D : \phi_D(F_{ij}) = \exp(-E_D(F_{ij}))$ donde $E_D(F_{ij})$ es la distancia Energetica del flujo óptico, se define para cada pixel como:



$$E_D(F_{ij}) = \frac{|F_{ij}|^2}{\beta^2} \quad (2)$$

Donde $|F_{ij}|^2$ es la distancia del vector de flujo F_{ij} y el parámetro β controla la varianza de la distancia de la energía, de esta manera se favorecen vectores de flujo óptico con una energía menor, por lo tanto movimientos cortos.

1.5.3 Factor de Vecindad

$\phi_N : \phi_N(F_{ij}) = \exp(-E_N(F_{ij}, Fn_{ij}))$ donde $E_N(F_{ij}, Fn_{ij})$ es la energía en la vecindad definida en cada par de pixels adyacentes $(p_{1ij}, p_{2ij}) \in E$:

$$E_N(F_{ij}, Fn_{ij}) = \frac{|F_{ij} - Fn_{ij}|^2}{\lambda^2} \quad (3)$$

Donde $|F_{ij} - Fn_{ij}|^2$ es la distancia del vector de diferencia F_{ij} y Fn_{ij} y λ , cuando la energía de las vecindades es similares, la energía resultante es baja, permitiendo a los pixeles de un mismo objeto moverse juntos y suavizando transiciones de flujo en contornos.

1.5.4 Energía Total

La energía total del modelo probabilístico $E(\mathbf{F})$ se define como:

$$\mathbf{E}(\mathbf{F}) = \sum_{(i,j) \in \mathbf{E}} E_I(F_{ij}) + \sum_{(i,j) \in \mathbf{E}} E_D(F_{ij}) + \sum_{(i,j) \in \mathbf{E}} E_N(F_{ij}, Fn_{ij}) \quad (4)$$

$$\mathbf{E}(\mathbf{F}) = \sum_{(i,j) \in \mathbf{E}} \frac{(I(p_{ij}) - I'(p_{ij} + F_{ij}))^2}{\alpha^2} + \sum_{(i,j) \in \mathbf{E}} \frac{|F_{ij}|^2}{\beta^2} + \sum_{(i,j) \in \mathbf{E}} \frac{|F_{ij} - Fn_{ij}|^2}{\lambda^2} \quad (5)$$

1.5.5 Minimización de la función de energía.

El uso de una función de energía o función costo requiere minimizar el error resultante. En dos imágenes consecutivas, si estas son capturadas con un intervalo de tiempo pequeño considerando una velocidad constante, se asume que el flujo óptico en un pixel de manera general se encuentra localizado en una subárea alrededor de la posición de dicho pixel en la primera imagen. Se selecciona el método de Gradiente Descendiente Estocástico considerando lo anterior para minimizar la función de energía.

El método Gradiente Descendiente Estocástico para optimizar funciones costo es utilizado comunmente en redes neuronales [3].

$$\theta = \theta - \eta \nabla J(\theta; x^{(i)}; y^{(i)}) \quad (6)$$

Por cada correspondencia de los puntos de interés de la primera imagen a la segunda, se crea un ciclo de enfriamiento para la función de energía 4 donde el criterio de paro es lo que se cumpla primero, un error menor a $e \leq 0,0001$ ó la cantidad de ciclos de enfriamiento



límite de 70, en donde la función de energía converge, se encuentra una correspondencia. Los valores mostrados se eligieron por medio de una heurística.

1.6 Triangulación y cálculo de la posición de la cámara.

El proceso de Triangulación consiste en los siguientes pasos:

- Generar por medio de los parámetros intrínsecos de la cámara la matrix esencial e indexar puntos inusuales de todos los puntos mediante la función *findEssentialMat* de OpenCV.
- Generar la matriz de rotación y traslación de la camara en la segunda imagen mediante la función *recoverPose* de OpenCV
- Corregir la distorsión de los puntos que convergieron mediante la función *undistort* de OpenCV.
- Triangular las correspondencias en el espacio.

La triangulación se realiza seleccionando 4 puntos sobre una base *affine* y despues calculando los puntos restantes sobre la misma base mediante *Singular Value Decomposition* [4].

Los puntos se almacenan en formato PLY para poderlo visualizar mediante el programa Blender.

2. Experimentación

El experimento se lleva a cabo en una Macbook con un procesador de 1.1 Ghz Intel Core M3 y 8 GB de memoria RAM. El proceso se programa en C++ y se utiliza OpenCV para la captura de las imágenes y para el almacenamiento de las mismas. La triangulación se realiza con funciones y procedimientos disponibles en OpenCV.

Se toman dos imagenes consecutivas, mediante la camara web de la computadora Macbook, en este ejemplo las de la Fig. 6 y la Fig. 7 de 480x720 pixeles.



Figura 8: Cuadro clave en gris.

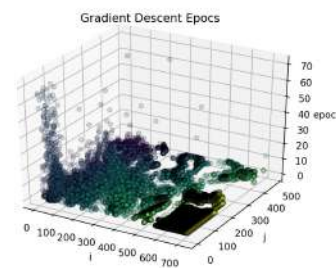


Figura 9: Ciclos de enfriamiento Gradiente Descendiente imagen completa



En la Fig. 8 se muestra una imagen en gris en la que se realiza la minimización de la función de energía. En la Fig. 7 se muestran la cantidad de ciclos de enfriamiento por cada correspondencia seleccionada que logró minimizar su temperatura.

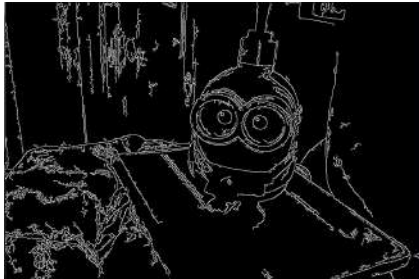


Figura 10: Cuadro Clave Preprocesado con el algoritmo *canny edge detection*.

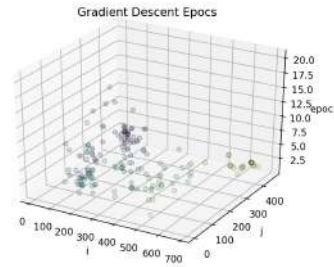


Figura 11: ciclos de enfriamiento Gradiente Descendiente imagen preprocesada con el algoritmo *canny edge detection*.

En la Fig. 10 se muestra una imagen preprocesada con el algoritmo *canny edge detection* [12]. Solo los pixeles en blanco de la Fig. 10 son entrada de la función de energía 4 y en la Fig. 11 se muestran la cantidad de ciclos de enfriamiento por cada correspondencia seleccionada que logró minimizar su temperatura.

Se encuentran las correspondencias minimizando la función de energía Eq. 5 tanto en el eje x como en el eje y para el factor de intensidad y factor de distancia en la función de energía Eqs. 1, 2 para obtener el flujo óptico de dos imágenes en donde se converge.

Después del proceso convergen 80412 en 47.9282 segundos puntos de los cuales se obtiene la siguiente triangulación de la escena:

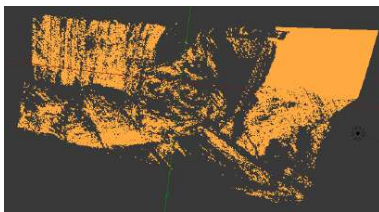


Figura 12: Nube 3D Figura 6 y 7 vista 1.

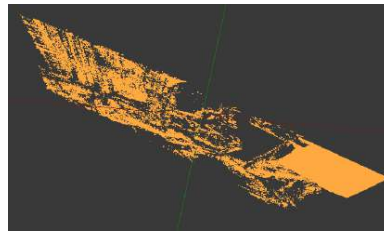


Figura 13: Nube 3D Figura 6 y 7 vista 2.

Si en el proceso de triangulación solo se toman en cuenta los puntos correctos que arroja el proceso de construcción de la matriz esencial, los puntos triangulados se reducen a 58435, el resultado es el siguiente:



Figura 14: Nube 3D Figuras 6 y 7 puntos correctos vista 1.



Figura 15: Nube 3D Figuras 6 y 7 puntos correctos vista 2.

Se realiza el proceso incluyendo un preprocesamiento adicional, detectando bordes con el método *canny edge detection* y solo se calculan aquellos pixeles que correspondan a la mascara generada por la detección de bordes sobre la imagen clave. El tiempo de procesamiento se reduce a 4.14785 segundos y los puntos que convergen son solamente 386.

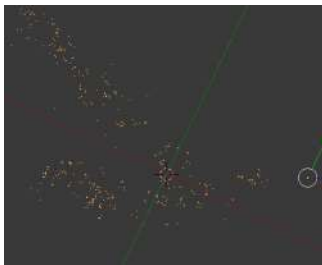


Figura 16: Nube 3D Figuras 6 y 7 aplicando *canny edge detection* vista 1.

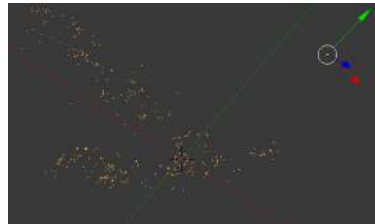


Figura 17: Nube 3D Figuras 6 y 7 *canny edge detection* vista 2.

Se toman dos imagenes consecutivas de una escena de una cocina, mediante la camara web de la computadora Macbook, en este ejemplo las de la Fig. 18 y la Fig. 19 de 480x720 pixeles.



Figura 18: Cocina 1.



Figura 19: Cocina 2.

La escena muestra objetos con distinta profundidad, existen planos como las puertas



de la cocina integral, sin embargo, son negras y no convergen o casi no convergen los pixeles que las componen como muestra en las Figs 18, 19, 20, 21

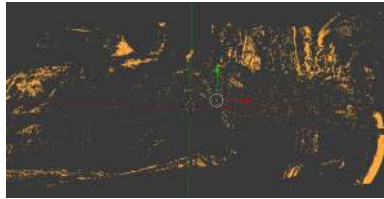


Figura 20: Cocina nube de puntos 1.



Figura 21: Cocina nube de puntos 2.

Se toman dos imagenes consecutivas de una escena de un camino, mediante la camara de un celular, en este ejemplo las de la Fig. 18 y la Fig. 19 de 1536x2048 pixeles, se generan 1312633 puntos y el tiempo de convergencia es de 363.361 segundos como se muestra en las Figs 24 y 25.



Figura 22: Camino 1.



Figura 23: Camino 2.

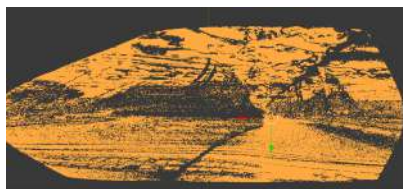


Figura 24: Nube 3D Figuras 22 y 23 vista 1.

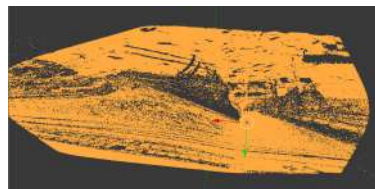


Figura 25: Nube 3D Figuras 22 y 23 vista 2.



3. Discusión y Conclusiones

La etapa de inicialización de un proceso SLAM actualmente tiene áreas de oportunidad de mejora en la robustez en escenas dinámicas y con cambios de iluminación, de igual manera hay posibilidad de mejora al buscar operar sin asumir características de una escena inicial [1].

1. La función de energía Eq. 4 basada en [2] permite obtener correspondencias entre dos imágenes consecutivas de manera aceptable bajo tolerancias a cambios de iluminación y deformaciones de objetos, observese la diferencia de iluminación en las figuras 6 y 7 de las cuales fue posible generar correspondencias y triangular una nube de puntos 3D Fig. 16.
2. El método de Gradiente Descendiente es útil para minimizar este tipo de funciones energéticas debido a la naturaleza localizada del problema donde no se requiere una solución global de minimización.
3. El procedimiento propuesto es susceptible a mejorarse por medio de etapas de preprocesamiento, la nube generada cuando se preprocesa la imagen con detección de bordes es mucho menos densa 386 puntos contra 80412 sin preprocesamiento, esta característica se puede utilizar para generar una escena en un proceso SLAM, al generar la escena e ir agregando puntos la escena puede ir creciendo en puntos acorde a necesidades variadas de densidad.
4. Se observa una mejor visualización de los puntos en escenas donde hay un pocos planos, esto es debido a que se utiliza la matriz esencial la cuál asume una escena no plana y con elementos en perspectiva en diferentes distancias.
5. La correspondencia de los puntos se puede utilizar de la misma manera en etapas posteriores de un proceso SLAM.

Referencias

- [1] Georges Younes, Daniel Asmar, Elie Shamma, and John Zelek. Keyframe-based monocular slam: design, survey, and future directions. *Robotics and Autonomous Systems*, 98:67–88, 2017.
- [2] Dongzhen Piao, Prahlad G Menon, and Ole J Mengshoel. Computing probabilistic optical flow using markov random fields. In *International Symposium Computational Modeling of Objects Represented in Images*, pages 241–247. Springer, 2014.
- [3] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
- [5] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.



- [6] C Daniel Herrera, Kihwan Kim, Juho Kannala, Kari Pulli, and Janne Heikkilä. Dt-slam: deferred triangulation for robust slam. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 609–616. IEEE, 2014.
- [7] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2018.
- [8] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.
- [9] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.
- [10] Ruben Gomez-Ojeda, David Zuñiga-Noël, Francisco-Angel Moreno, Davide Scaramuzza, and Javier Gonzalez-Jimenez. Pl-slam: a stereo slam system through the combination of points and line segments. *arXiv preprint arXiv:1705.09479*, 2017.
- [11] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.
- [12] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.
- [13] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.
- [14] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.
- [15] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

0.2. Requisito manejo de la lengua inglés



UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE
LENGUAS Y LETRAS



A QUIEN CORRESPONDA:

La que suscribe, Directora de la Facultad de Lenguas y Letras, hace **C O N S T A R** que

ROMAN RIVERA LUIS ROGELIO

Presentó el **Examen de Manejo de la Lengua** efectuado el día veintiséis de febrero de dos mil diecinueve, en el cual obtuvo la siguiente calificación:

8-

Se extiende la presente a petición de la parte interesada, para los fines escolares y legales que le convengan, en el Campus Aeropuerto de la Universidad Autónoma de Querétaro, el día cinco de marzo de dos mil diecinueve.



Atentamente,
"Enlazar Culturas por la Palabra"

LIC. LAURA PÉREZ TÉLLEZ



LPT/evm*CL*FLL-C.-116

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
Campus Aeropuerto, Anillo Vial Fray Junípero Serra S/N, Querétaro, Gro.
C.P. 76140, Tel. 01 (442) 192 12 00 Dirección Ext. 61010,
Secretaría Administrativa 61300, Posgrado 61140, Licenciatura 61070,
Centro de Lenguas 61050, Secretaría Académica 61100 y Planeación 61110

SOMOSUAQ
EDUCAR CRECER CONSOLIDAR

0.3. Código C++

0.3.1 Gradiente de un pixel

Código 1: gradiente de un pixel eje X y Y

```
1 void getGradiujv(cv::Mat &GX, cv::Mat &GY, cv::Mat &E,  
2             cv::Mat &U, cv::Mat &V, cv::Mat &a, cv::Mat &b,  
3             int i, int j, cv::Mat &UN, cv::Mat &VN){  
4     // Factor de Intensidad  
5     float alpha=0.99, beta=0.99, gamma=0.99, temp=0.0;  
6     float C = a.at<float>(j, i), CN = 0.0, ntemp = 0.0;  
7     float nytemp=0.0, num =0.0, num2=0.0;  
8     num= -2*C+(2*(i+(UN.at<int>(j, i))));  
9     if (num==0){num = 1;}  
10    GX.at<float>(j, i) = num / pow(alpha, 2);  
11    num= -2*C+(2*(j+(VN.at<int>(j, i))));  
12    if (num==0){num = 1;}  
13    GY.at<float>(j, i) = num / pow(alpha, 2);  
14  
15    //Factor de distancia  
16    num= 4*pow(UN.at<int>(j, i), 3);  
17    num2= 4*UN.at<int>(j, i)*pow(VN.at<int>(j, i), 2);  
18    if (num==0){num = 1;}  
19    GX.at<float>(j, i) += (num+num2) / pow(beta, 2);  
20    num= 4*pow(VN.at<int>(j, i), 3);  
21    num2= 4*VN.at<int>(j, i)*pow(UN.at<int>(j, i), 2);  
22    if (num==0){num = 1;}  
23    GY.at<float>(j, i) += (num+num2) / pow(beta, 2);  
24  
25 }
```

0.3.2 Energía de un pixel

Código 2: gradiente de un pixel eje X y Y

```
1 void getEiujv(cv::Mat &E, cv::Mat &U, cv::Mat &V,  
2             cv::Mat &a, cv::Mat &b, int i,  
3             int j, cv::Mat &UN, cv::Mat &VN){  
4     float alpha=0.999, beta=0.9, gamma=0.9;  
5     float temp=0.0, tempe = 0.0;  
6     int rows = a.rows;  
7     int cols = a.cols;  
8     int tempi= i+UN.at<int>(j, i);  
9     int tempj= j+VN.at<int>(j, i);  
10    // Factor de Intensidad  
11    if ((tempi < 0 || tempi > (cols - 1)) ||  
12        (tempj < 0 || tempj > (rows - 1)))  
13        temp = 1.0;  
14    else temp = abs(a.at<float>(j, i)  
15                  - b.at<float>(tempj, tempi));  
16    E.at<float>(j, i) = pow(temp, 2) / pow(alpha, 2);  
17    // Factor de distancia  
18    E.at<float>(j, i) += abs(pow(U.at<float>(j, i), 2)  
19                          + pow(V.at<float>(j, i), 2)) / pow(beta, 2);  
20    // Factor de vecindad  
21    tempe += (pow(a.at<float>(j, i) - a.at<float>(j+1, i), 2)  
22             - pow(b.at<float>(tempj, tempi)  
23                 - b.at<float>(tempj+1, tempi), 2));  
24    tempe += (pow(a.at<float>(j, i) - a.at<float>(j+1, i-1), 2)  
25             - pow(b.at<float>(tempj, tempi)  
26                 - b.at<float>(tempj+1, tempi-1), 2));  
27    tempe += (pow(a.at<float>(j, i) - a.at<float>(j-1, i+1), 2)  
28             - pow(b.at<float>(tempj, tempi)  
29                 - b.at<float>(tempj-1, tempi+1), 2));  
30    tempe += (pow(a.at<float>(j, i) - a.at<float>(j-1, i), 2)  
31             - pow(b.at<float>(tempj, tempi)  
32                 - b.at<float>(tempj-1, tempi), 2));  
33    tempe += (pow(a.at<float>(j, i) - a.at<float>(j-1, i-1), 2)  
34             - pow(b.at<float>(tempj, tempi)  
35                 - b.at<float>(tempj-1, tempi-1), 2));  
36    tempe += (pow(a.at<float>(j, i) - a.at<float>(j, i-1), 2)  
37             - pow(b.at<float>(tempj, tempi)  
38                 - b.at<float>(tempj, tempi-1), 2));  
39    tempe += (pow(a.at<float>(j, i) - a.at<float>(j+1, i+1), 2)  
40             - pow(b.at<float>(tempj, tempi)  
41                 - b.at<float>(tempj+1, tempi+1), 2));  
42    // tempe += (pow(a.at<float>(j, i) - a.at<float>(j+1, i), 2)
```

```
43     - pow(b.at<float>(j, i)-b.at<float>(j+1, i),2)) ;
44     // tempe += (pow(a.at<float>(j, i)-a.at<float>(j+1, i-1),2)
45     - pow(b.at<float>(j, i)-b.at<float>(j+1, i-1),2));
46     // tempe += (pow(a.at<float>(j, i)-a.at<float>(j-1, i+1),2)
47     - pow(b.at<float>(j, i)-b.at<float>(j-1, i+1),2));
48     // tempe += (pow(a.at<float>(j, i)-a.at<float>(j-1, i),2)
49     - pow(b.at<float>(j, i)-b.at<float>(j-1, i),2));
50     // tempe += (pow(a.at<float>(j, i)-a.at<float>(j-1, i-1),2)
51     - pow(b.at<float>(j, i)-b.at<float>(j-1, i-1),2));
52     // tempe += (pow(a.at<float>(j, i)-a.at<float>(j, i-1),2)
53     - pow(b.at<float>(j, i)-b.at<float>(j, i-1),2));
54     // tempe += (pow(a.at<float>(j, i)-a.at<float>(j+1, i+1),2)
55     - pow(b.at<float>(j, i)-b.at<float>(j+1, i+1),2));
56
57     tempe = tempe/(pow(gamma,2)*7);
58     E.at<float>(j, i) += abs(tempe);
59 }
```
