Universidad Zaragoza
1542

Trabajo Fin de Grado

Cálculo de profundidad con una cámara de ojo de pez estéreo

Depth computation from a stereo fisheye camara

Autor

César Sáenz Rodríguez

Director/es

José Jesús Guerrero Campo

Samuel Bruno Berenguel Baeta

Escuela de Ingeniería y Arquitectura
2023

RESUMEN

En el presente trabajo de fin de grado, se va a diseñar un algoritmo que calcule un mapa de profundidad a partir de un par estéreo de imágenes. Para ello se va a utilizar el modelo proyectivo de Kannala-Brandt, así como la geometría epipolar, y la búsqueda de correspondencias mediante un método fotométrico.

Este algoritmo se va a realizar en el lenguaje de programación Python y se probará la robustez de dicho algoritmo mediante diferentes ejemplos de prueba, así como su funcionamiento cambiando ciertos parámetro clave.

ABSTRACT

In this final degree work, an algorithm will be designed to calculate a depth map from a stereo pair of images. Using the Kannala-Brandt projective model, as well as the epipolar geometry, and the search of matchings by photometric method.

This algorithm is going to be performed in the Python programming language and the robustness of this algorithm will be tested by different examples, the performance of this algorithm will be study as well changing some key parameters.

Índice

1. Introducción

Al realizar una fotografía del entrono se pierde información, ya que estamos transformando un espacio 3D en otro 2D. En el contexto de la visión por computador, recuperar esta información es de gran ayuda ya que nos permite hacer otras tareas de forma más segura y con una mayor repetibilidad.

La profundidad, que es la dimensión eliminada en el proceso, se reconstruye a partir de la triangulación de rayos pertenecientes a un mismo punto tridimensional capturado desde diferentes puntos de vista. Este proceso tiene la misma base que la percepción 3D de la visión humana.

En este trabajo se va a realizar el cálculo de un mapa de profundidad a través de una cámara ojo de pez estéreo, estas cámaras destacan por proporcionan un campo de visión (FOV) mucho más amplio, desde 100º hasta 280º que las cámaras convencionales, donde los campos de visión se encuentran entre 40º y 60º. Además este sistema de visión está formado por dos objetivos de ojo de pez dispuestos en una configuración estereoscópica [1].

El programa desarrollado para el cálculo de profundidad tiene aplicaciones en diferentes campos, como la navegación autónoma de vehículos, la robótica y la creación de mapas 3D a partir de imágenes de satélite.

2. Objetivos y alcance

El objetivo del presente TFG es desarrollar un algoritmo que calcule un mapa de profundidad semidenso a partir de un par estéreo de imágenes de ojo de pez.

Para esto hay que tener en cuenta la distorsión implicitica de este tipo de cámaras y que no se puede aplicar el modelo de proyección de una cámara pinhole, sino que se debe aplicar modelos más complejos como el modelo de proyección de Kannala-Brandt.

Por otra parte también se debe destacar que nuestro problema está planteando en una geometría de dos visitas, lo que nos va a permitir aplicar la teoría de la geometría epipolar.

Por último, se evaluará la precisión y robustez del algoritmo desarrollado mediante pruebas con imágenes reales y su funcionamiento cambiando ciertos parámetros de diseño del algoritmo.

Dicho algoritmo se desarrollará en Python con librerías específicas para la visión por compuatdor, el análisis matemático y reconstrucción tridimensional (OpenCV, Numpy y Open3D).

Este algoritmo se evaluará con imágenes reales de una cámara de ojo de pez estéreo (en concreto la Inter RealSense T265).

Hay que destacar que el alcance del trabajo no incluye la implementación del algoritmo en un sistema de visión en tiempo real. Además el algoritmo está limitado para obtener como máximo profundidades de hasta 5 metros, ya que aumentar la distancia máxima de profundidad implica un coste computacional elevado.

Por último también se debe mencionar que la cámara no realiza la captura de imágenes en el mismo instante de tiempo para cada objetivo. Para la aplicación y caso de estudio en este trabajo, esta diferencia temporal es despreciable y no se tiene en cuenta.

3. Cámaras con lente de ojo de pez

En los últimos años el uso de cámaras con lente de ojo de pez para aplicaciones en robótica a aumentando, esto se debe a la gran cantidad de ventajas que tienen frente a las cámaras convencionales. Como son, un mayor campo de visión, la captura de imágenes panorámicas o un menor requerimiento de alineamiento.[2]

Aun así estas cámaras también presentan una serie de inconvenientes, como la distorsión implícita provocada por estos objetivos, requerimientos de procesamientos más altos y limitación en la detección de objetos.[2]

Tradicionalmente, a las cámaras se le aplica el modelo de proyección de la cámara pinhole junto con una corrección de la distorsión. Esto no se puede aplicar a las lentes de ojo de pez ya que están diseñadas para captar todo el campo hemisférico en frente de la cámara, pudiendo capturar hasta 280º para algunas cámaras. Es más es imposible proyectar el campo hemisférico de visión en un plano finito con la proyección de perspectiva, por esta razón las lentes de ojos de pez no pueden ser consideradas como una desviación del modelo de cámara convencional.

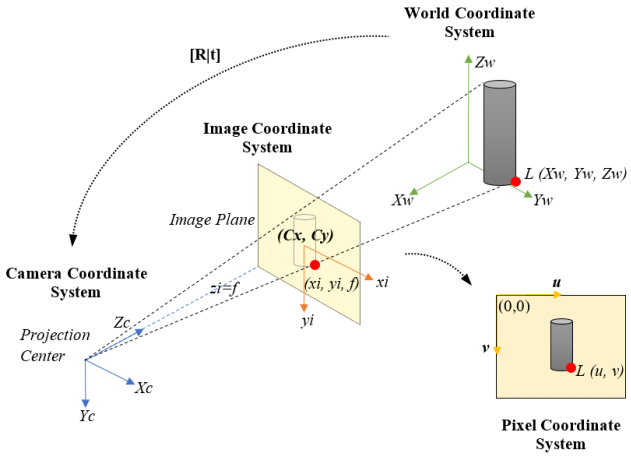


Figura 1.Modelo proyectivo de un objeto 3D a una imagen 2D de la cámara pinhole.[3]

Con esta idea en 2006 Juho Kannala y Sami S. Brandt [4] desarrollan un modelo matemático que permite describir la proyección de imágenes captadas por una lente de ojo de pez. De forma general, este modelo define la distorsión de la imagen como:

$$r(\theta) = k_1\theta + k_2\theta^3 + k_3\theta^5 + k_4\theta^7 + k_5\theta^9 + \cdots \qquad (1)$$

Para computación debemos limitar el número de términos, y se ha comprobado experimentalmente que si llegamos a la novena potencia tenemos suficientes grados de libertad para realizar una buena aproximación de la curva. Una vez obtenido el rayo podemos calcular la proyección del punto en las coordenadas normalizadas de la imagen como:

$$\begin{pmatrix} x \\ y \end{pmatrix} = r(\theta) \begin{pmatrix} cos\varphi \\ sin\varphi \end{pmatrix} \quad (2)$$

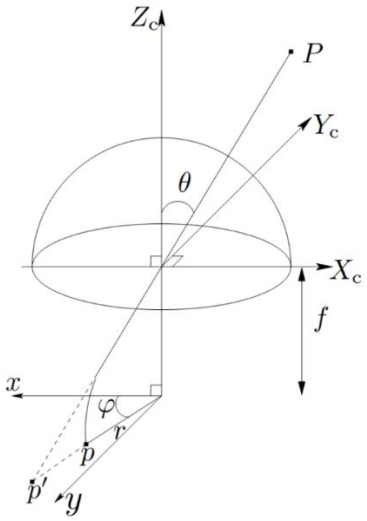


Figura 2. Modelo proyectivo de las cámaras con lente de ojo de pez.[4]

Las lentes reales se desvían de este modelo de proyección radial simétrico, y para completarlo debemos añadirle una parte antisimétrica. Para ello se añaden dos términos de distorsión, uno en la dirección radial:

$$\Delta_r(\theta, \varphi) = (l_1\theta + l_2\theta^3 + l_3\theta^5)(i_1 cos\varphi + i_2 sin\varphi + i_3 cos2\varphi + i_4 sin2\varphi) \quad (3)$$

Y otro en la dirección tangencial:

$$\Delta_t(\theta, \varphi) = (m_1\theta + m_2\theta^3 + m_3\theta^5)(j_1 cos\varphi + j_2 sin\varphi + j_3 cos2\varphi + j_4 sin2\varphi) \quad (4)$$

Añadiendo estos términos a la ecuación (2), obtenemos las coordenadas distorsionadas:

$$\boldsymbol{x_d} = r(\theta)\boldsymbol{u}_r(\varphi) + \Delta_r(\theta, \varphi)\boldsymbol{u}_r(\varphi) + \Delta_t(\theta, \varphi)\boldsymbol{u}_t(\varphi) \quad (5)$$

Donde $\boldsymbol{u}_r(\varphi)$ y $\boldsymbol{u}_t(\varphi)$ son vectores unitarios en las direcciones radial y tangencial. Por último para conseguir el modelo completo debemos transformar las coordenadas distorsionadas en píxeles de la imagen, obteniendo:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} m_u & 0 \\ 0 & m_v \end{bmatrix} \begin{pmatrix} x_d \\ y_d \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \quad (6)$$

Donde $(u_0, v_0)^T$ son las coordenadas en píxeles del centro óptico de la cámara y m_u y m_v dan el número de píxeles por unidad de distancia en las direcciones horizontal y vertical. De esta forma obtenemos el modelo directo de la cámara, que describe cómo se proyectan los puntos 3D del mundo real en coordenadas de una imagen 2D.

De igual forma que existe un modelo directo también existe un modelo inverso, que nos permite obtener los rayos 3D del mundo real que atraviesan el objetivo y generan el plano de la imagen 2D. Tanto el modelo directo como el modelo inverso van a ser utilizados para resolver el problema planteado en este trabajo de fin de grado.

Por último hay que mencionar que este modelo proyectivo utiliza los parámetros intrínsecos y extrínsecos de los objetivos, así como los vectores de parámetros de distorsión.

4. Geometría Epipolar

Con el modelo de proyección vamos a poder transformar los píxeles de la imagen 2D en rayos 3D del mundo real como bien se ha explicado en el apartado 3, pero seguimos sin conocer cuál es la profundidad del punto que nos genera el píxel en la imagen.

Para lograr esto debemos tener en cuenta la geometría que generan las cámaras estéreo, la cual es conocida como geometría epipolar.[5]

La geometría epipolar se obtiene cuando dos cámaras captan el mismo espacio 3D desde diferentes puntos de vista, obteniendo el siguiente diagrama:

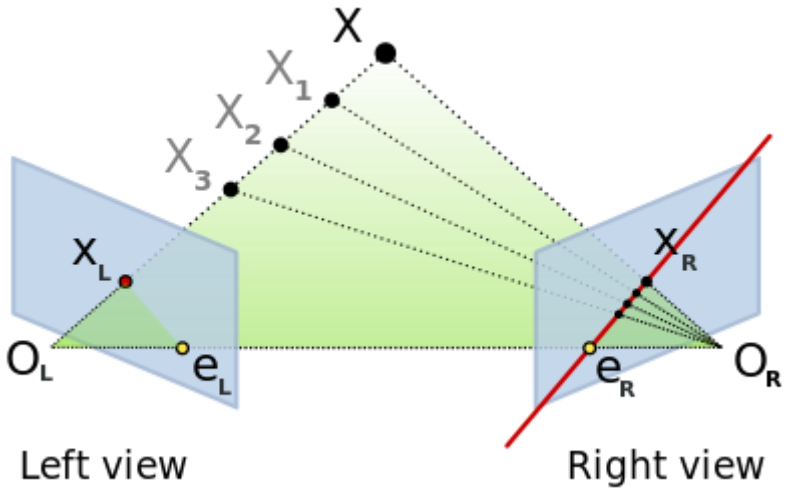


Figura 3. Geometría epipolar.[6]

En la Figura 3 observamos los puntos O_L y O_R los cuales son los centros ópticos de las dos lentes de la cámara, el punto tridimensional $\boldsymbol{X} = (x, y, z)$ y los puntos $\boldsymbol{X}_l$ y $\boldsymbol{X}_R$ que son las proyecciones del punto $\boldsymbol{X}$ en los planos de la imagen.

La proyección del centro óptico izquierdo sobre el plano derecho genera el epipolo derecho en la Figura 3 representado como $\boldsymbol{e}_R$. De igual forma el centro óptico derecho genera el epipolo izquierdo $\boldsymbol{e}_R$. Todos estos puntos tanto los centros ópticos, como los epipolos quedan unidos por la línea base. También, tenemos el plano epipolar el cual está formado por la línea base, y la unión de los centros ópticos con el punto $\boldsymbol{X}$. Por último las rectas epipolares son las rectas que unen los centros ópticos con el punto tridimensional proyectadas sobre el plano opuesto.

Como se ha visto anteriormente, las cámaras con lentes de ojo de pez no se adaptan a los modelos geométricos convencionales y al igual que con el modelo de proyección, hay que realizar ciertas distinciones para la geometría epipolar [7].

En el caso de las cámaras convencionales, los planos epipolares intersecan el plano de la imagen con líneas que atraviesan un único epipolo.

En cambió en los objetivos de lente de ojo de pez los planos epipolares intersecan una retina esférica, lo que genera una forma circular, que al ser proyectada en el plano de la imagen genera una curva que cruza dos epipolos. El campo de visión de la cámara define el número de epipolos que se ven en la imagen pudiendo ser desde cero hasta dos.

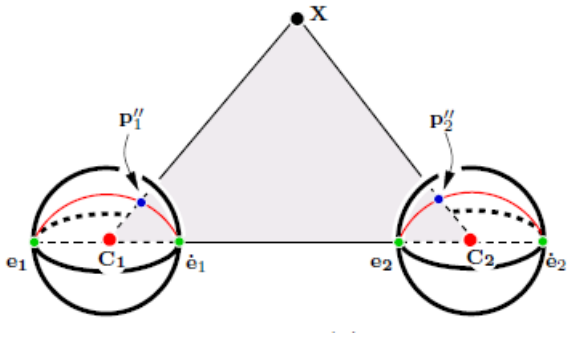


Figura 4. Geometría epipolar de cámaras con lente de ojo de pez.[7]

En la Figura 4 se observa como el plano epipolar interseca los objetivos esféricos en dos puntos, generando dos pares de epipolos por cada objetivo. Los puntos p_1'' y p_2'' son las proyecciones del punto X en el plano de la imagen.

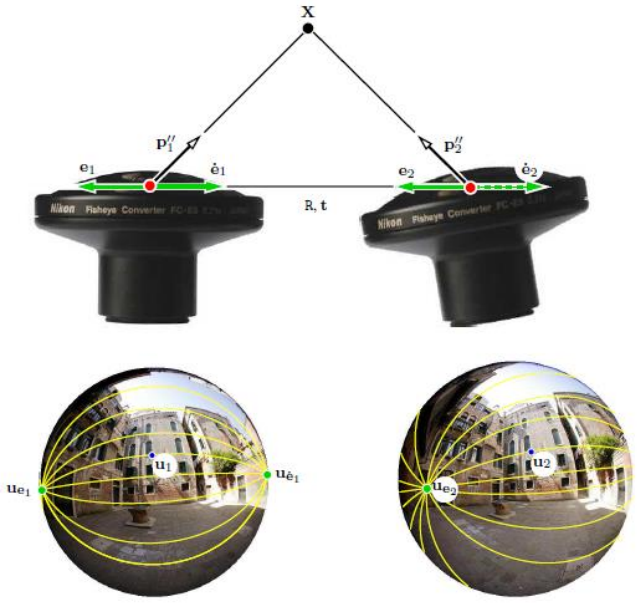


Figura 5. Geometría epiplar de un par de lentes de ojo de pez. Lineas epipolares proyectadas sobre el plano de la iamgen. [7]

En la Figura 5 se puede observar cómo las formas circulares que generan las proyecciones de los planos epipolares se ve proyectada en la imagen.

Conocer esta geometría nos va a permitir, tanto triangular los puntos para poder formar el mapa de profundidad, como obtener el matching (obtener píxeles que corresponden al mismo punto 3D) de diferentes puntos.

5. Resolución y resultados

Una vez expuestos los fundamentos teóricos que rigen nuestro problema pasamos a explicar cómo se ha obtenido la solución de forma técnica.

Como se ha nombrado en los Objetivos y alcance de este TFG, este problema se va a resolver mediante el lenguaje de programación de Python con el uso de ciertas librerías. Para observar el funcionamiento del algoritmo se van a utilizar 5 pares de imágenes a lo largo del desarrollo de la resolución:

Figura 6. Fotografía ejemplo 1 cámara izq

Figura 7. Fotografía ejemplo 1 cámara dcha

Figura 8. Fotografía ejemplo 2 cámara izq

Figura 9. Fotografía ejemplo 2 cámara dcha

Figura 10. Fotografía ejemplo 3 cámara izq

Figura 11. Fotografía ejemplo 3 cámara dcha

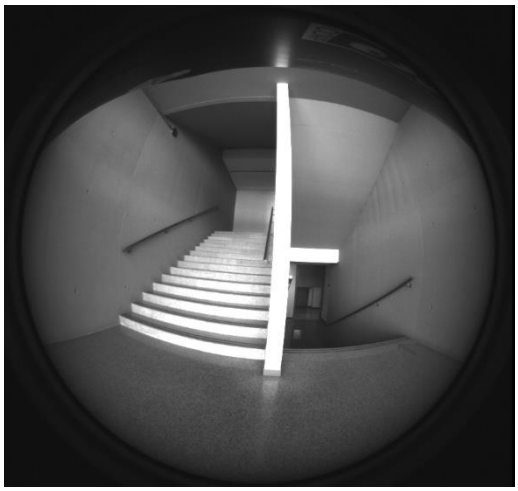


Figura 12. Fotografía ejemplo 4 cámara izq

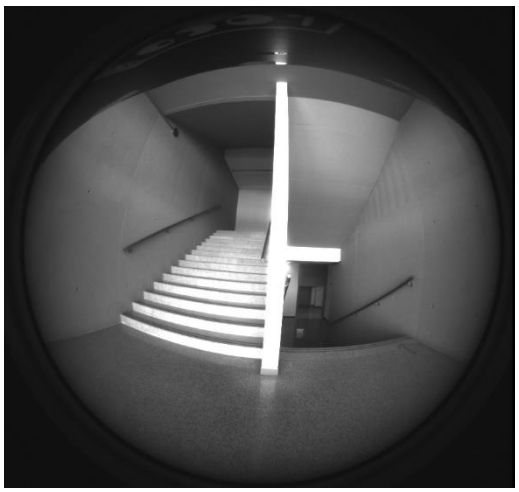


Figura 13. Fotografía ejemplo 4 cámara dcha

Figura 14. Fotografía ejemplo 5 cámara izq

Figura 15. Fotografía ejemplo 5 cámara dcha

En primer lugar, identificamos los puntos clave que contienen cada una de las imágenes mediante los algoritmos SIFT(Anexo B. Scale Invariant Feature Transform (SIFT)) y Canny (Anexo C. Canny Edge Detector). Estos algoritmos son dos funciones que se encuentra en la librería OpenCV y que nos permite detectar y describir píxeles característicos.

Aplicamos dichos algoritmos a la imagen izquierda, y así obtenemos un vector que contienen la posición en dimensiones de píxeles de la imagen (X,Y) de todos los puntos clave que se ha logrado detectar con dichos algoritmos.

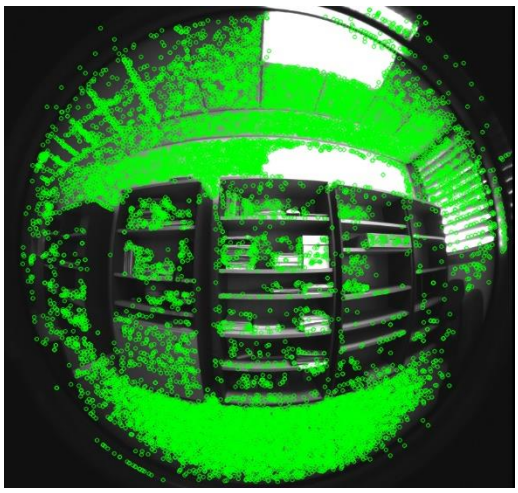


Figura 16. SIFT Feature Detection (1)

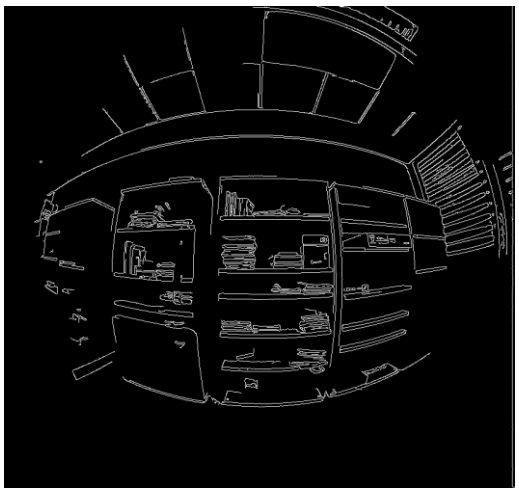


Figura 17. Canny Edge Detection (1)

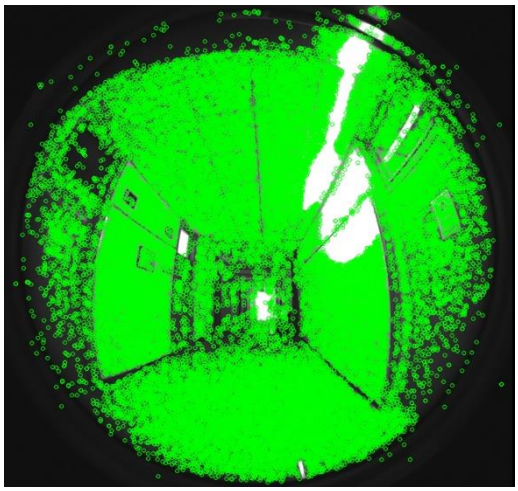


Figura 18. SIFT Feature Detection (2)

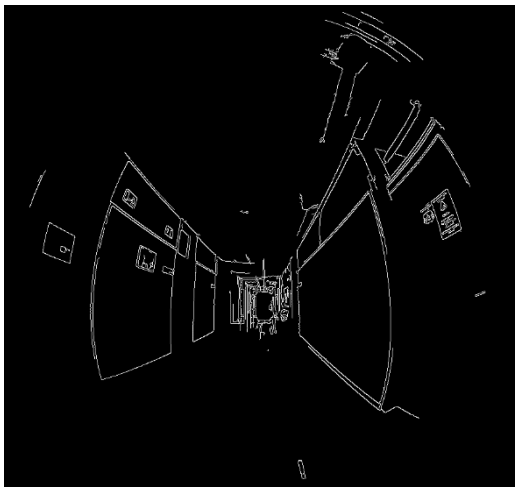


Figura 19. Canny Edge Detection (2)

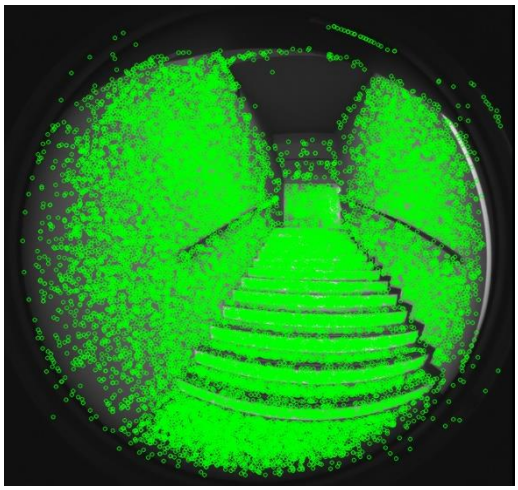


Figura 20. SIFT Feature Detection (3)

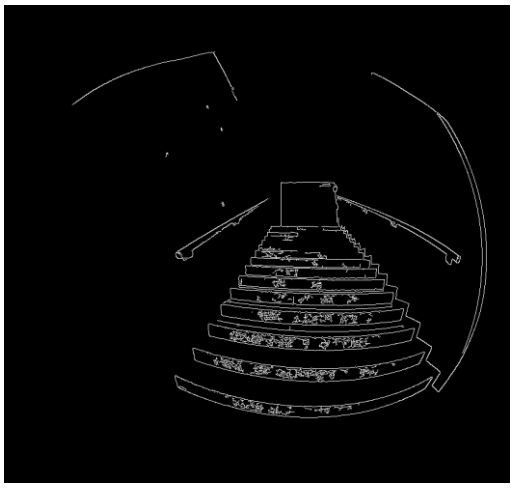


Figura 21. Canny Edge Detection (3)

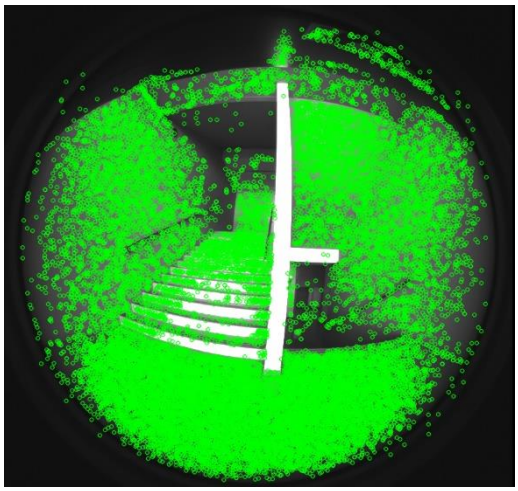


Figura 22. SIFT Feature Detection (4)

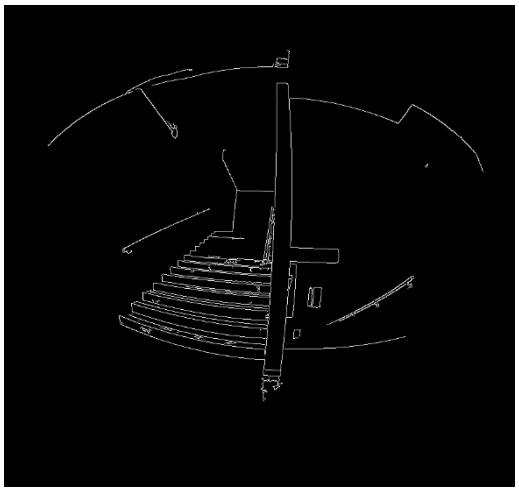


Figura 23. Canny Edge Detection (4)

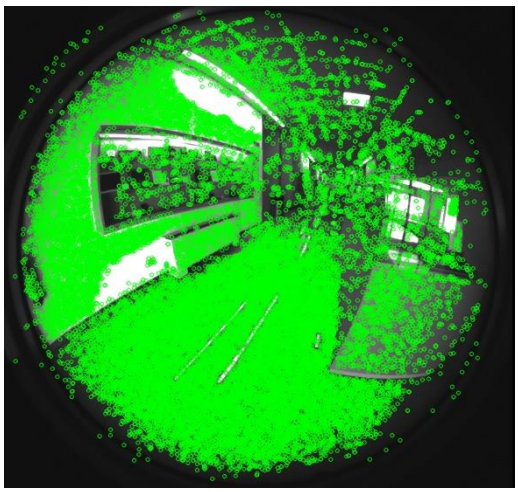


Figura 24. SIFT Feaature Detection (5)

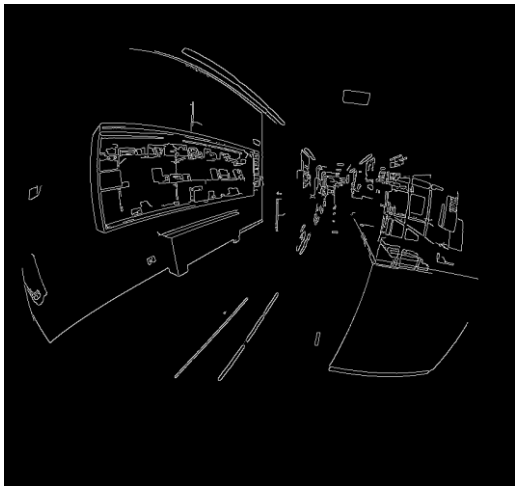


Figura 25. Canny Edge Detection (5)

Una vez obtenidos los puntos, se va a tratar de encontrar el máximo número de matchings posibles en la imagen derecha. Se entiende por matching, la actividad de encontrar el par de píxeles (tanto de la imagen derecha como el de la imagen izquierda) que representan al mismo punto tridimensional. Para ello se calcular el rayo 3D asociado a cada píxel a través del modelo inverso de Kannala-Brandt, con esto obtenemos un vector de tres componentes respecto a cada píxel encontrando mediante los métodos SIFT y Canny:

$$\boldsymbol{v}_{c1}^{i} = \begin{pmatrix} x_{c1}^{i} \\ y_{c1}^{i} \\ z_{c1}^{i} \end{pmatrix} \tag{7}$$

Donde $i \in [0, N]$ siendo N el número total de píxeles detectados.

A través del vector (7) obtenemos diferentes puntos 3D multiplicando por un escalar (el valor del escalar definirá la profundidad del punto 3D). Estos puntos 3D se trasladan a la referencia del segundo objetivo y mediante el modelo directo de Kannala-Brandt se proyectan sobre la segunda imagen.

Obteniendo así la línea epipolar proyectada en la imagen captada por el objetivo derecho:

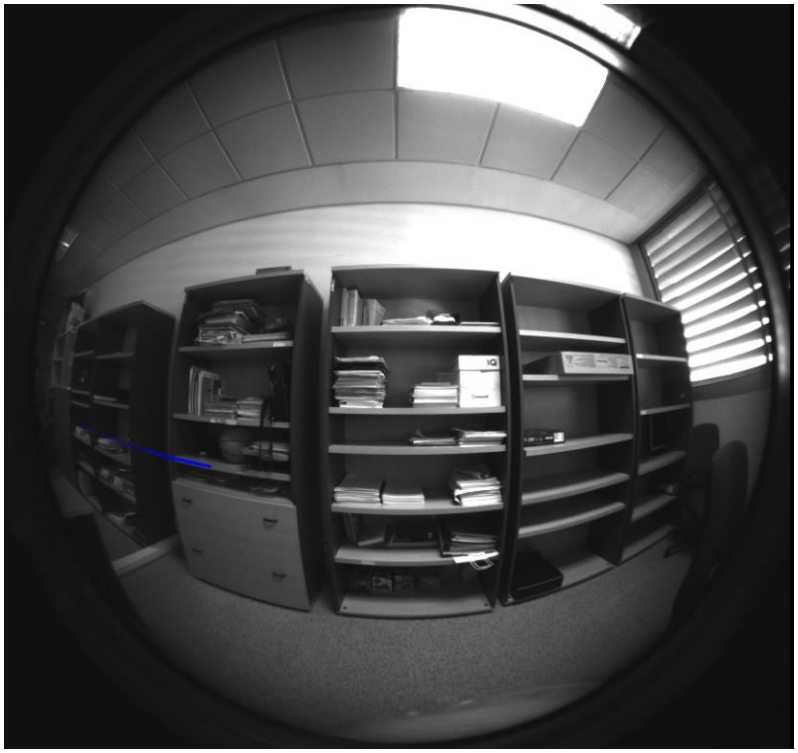


Figura 26. Línea Epipolar

Una vez obtenida la recta epipolar, se calcula una matriz 7x7 (denominada kernel) que contiene los valores en escala de grises de los píxeles que se encuentran alrededor del píxel objeto de estudio (píxel de la imagen izquierda desde el que se calcula el rayo 3D). De igual forma se calcula esta matriz para cada píxel que compone la recta epipolar.

Mediante la correlación cruzada normalizada (NCC), se tratará de buscar el píxel de la imagen derecha que más se asemeje al píxel de la primera imagen. Dicha correlación se define como:

$$NCC = \frac{\sum_i [I_0(x_i) - \bar{I}_0][I_1(x_i + u) - \bar{I}_1]}{\sqrt{\sum_i [I_0(x_i) - \bar{I}_0]^2} \sqrt{\sum_i [I_1(x_i + u) - \bar{I}_1]^2}} \quad (8)$$

Donde

$$\bar{I}_0 = \frac{1}{N} \sum_i I_0(x_i) \quad (9)$$

$$\bar{I}_1 = \frac{1}{N} \sum_i I_1(x_i + u) \quad (10)$$

Donde $I_0(x_i)$, es valor de gris (entre 0 y 255) que tiene el kernel definido en la primera imagen para el píxel i, e $I_1(x_i)$ es el valor de gris que toma el el kernel definido en la segunda imagen para el píxel i, es decir se compara valor a valor los diferentes kernels.

El valor de NCC puedes estar entre [-1,1] y se considerará que se ha obtenido un buen nivel de semejanza en el momento en el que NCC tomé un valor igual o superior a 0.7. Se calcula el valor de NCC, para cada píxel obtenido en la imagen izquierda comprando con cada punto de la recta epipolar.

Aquel que nos proporcione el valor más cercano a 1 será el punto escogido, y nos quedaremos tanto con la posición del píxel como con la posición del punto 3D. Obteniendo así un vector que contiene los puntos 3D, y otro vector que almacena los píxeles para los que se ha encontrado un matching en la segunda imagen.

Cabe destacar que como se observa en las figuras hay una zona negra en la cual no existe imagen por tanto debemos eliminar dicha zona para no obtener ningún píxel, ya que podría generar resultados erróneos.

A continuación se muestran los matching obtenidos de los respectivos ejemplos:

Figura 27. Matching fotométrico (1)

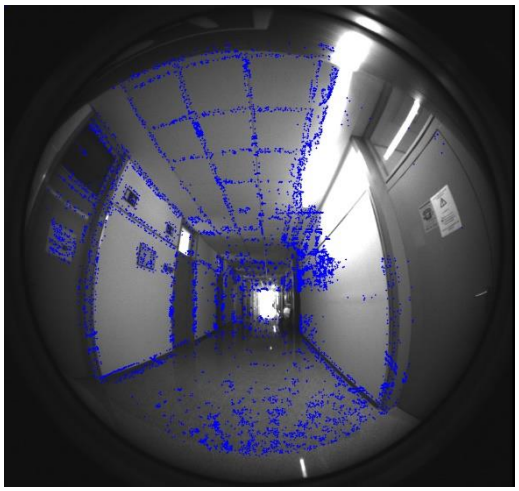


Figura 28. Matching fotométrico (2)

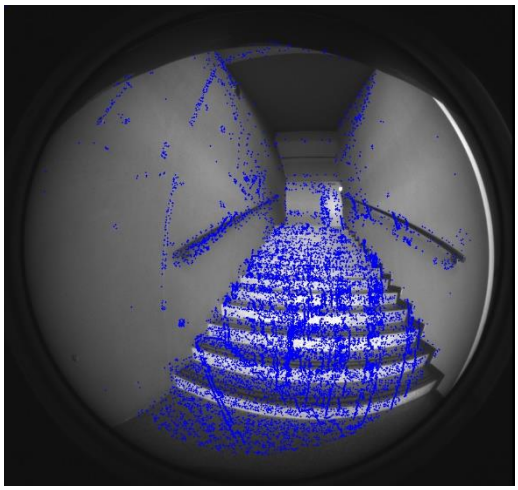


Figura 29. Matching fotométrico (3)

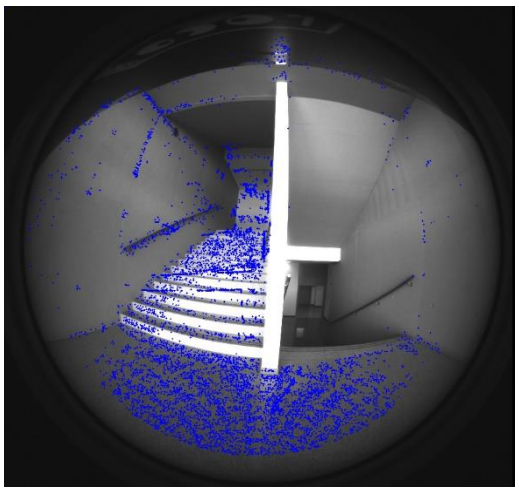


Figura 30. Matching fotométrico (4)

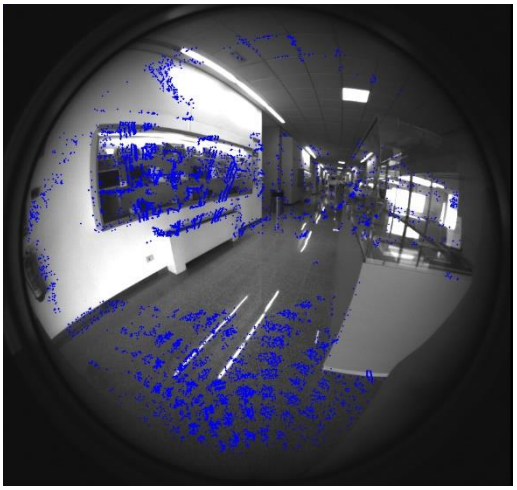


Figura 31. Matching fotométrico (5)

A modo de resumen y para la posterior comparación se van a almacenar en una tabla los resultados obtenidos durante la implementación del programa respecto a cada par de imágenes. Estos resultados son los píxeles encontrados mediante el método SIFT y el método Canny y la cantidad de píxeles con los que se ha logrado hacer matching.

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	19247
2	51719	8988
3	36817	17284
4	34248	10432
5	46900	17401

Figura 32. Tabla resultados con parámetros iniciales

Una vez tenemos el vector de puntos 3D generamos la nube de puntos mediante la librería Open3D. Hay que destacar que por cada par de imágenes se obtiene una nube de puntos de la cual se han sacado capturas desde diferentes puntos de vista para poder tener una mejor idea de la solución obtenida:

- Resultados fotografía I:

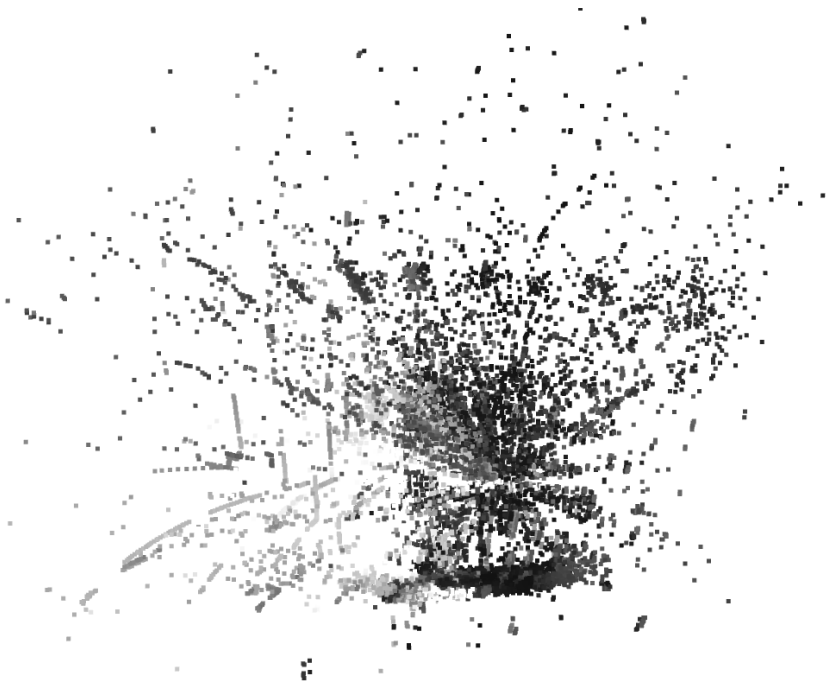


Figura 33. Nube de puntos 1.1

Figura 34. Nuevo de puntos 1.2

- Resultados fotografía II:

Figura 35. Nube de puntos 2.1

Figura 36. Nube de puntos 2.2

- Resultados fotografía III:

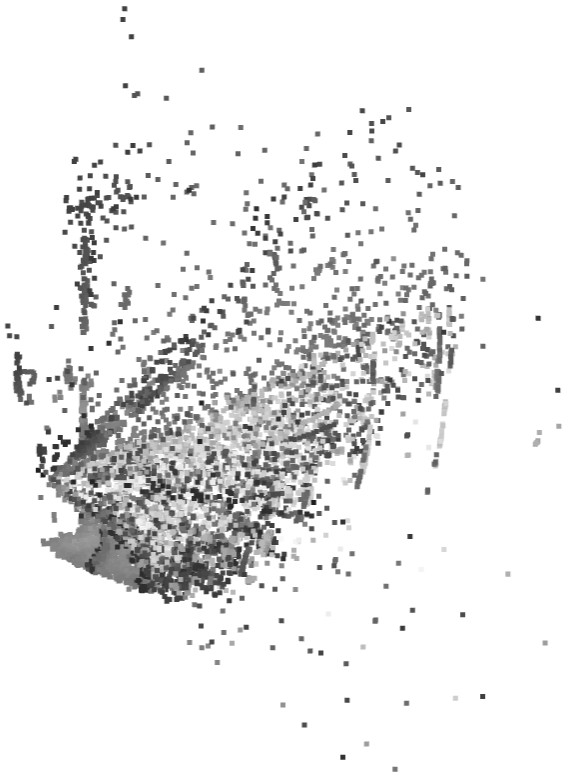


Figura 37. Nube de puntos 3.1

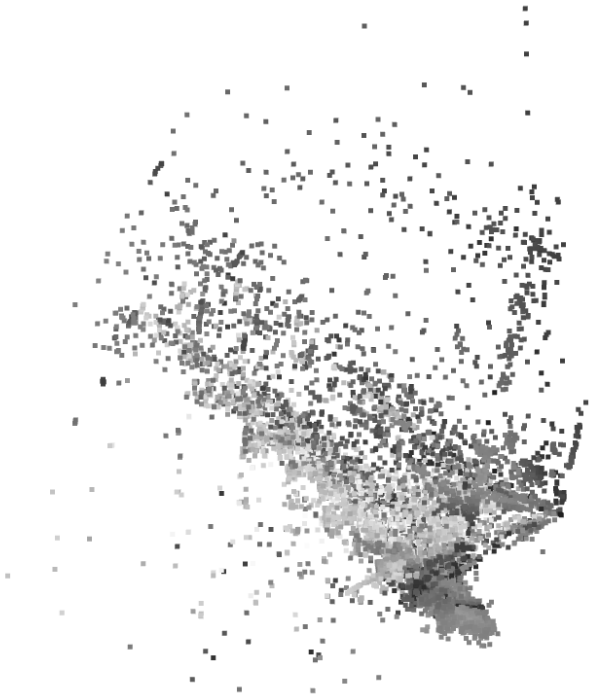


Figura 38. Nube de puntos 3.2

- Resultados fotografía IV:

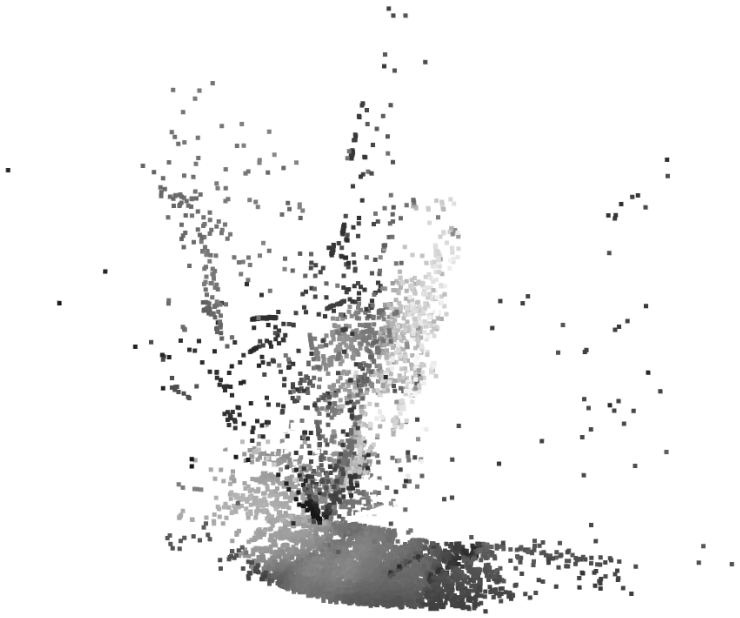


Figura 39. Nube de puntos 4.1

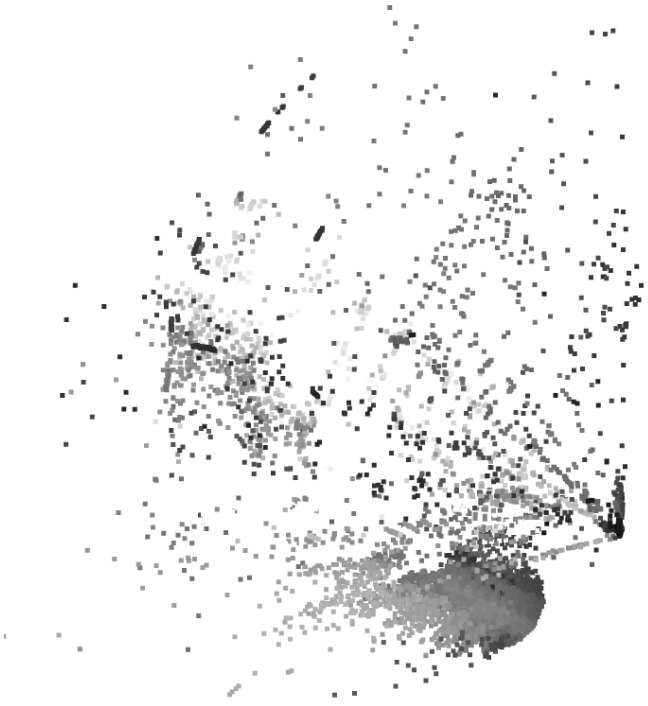


Figura 40. Nube de puntos 4.2

- Resultado fotografía V:

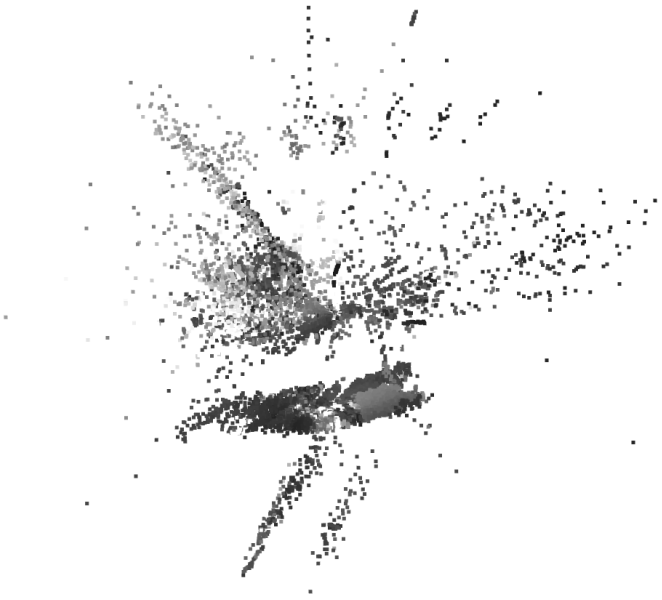


Figura 41. Nube de puntos 5.1

Figura 42. Nube de puntos 5.2

Como se observa en los resultados el algoritmo funciona mejor en espacios bien delimitados y sin muchos detalles, como por ejemplo en la fotografía 3 y en la fotografía 4 en las cuales se distinguen claramente el suelo, las escaleras y la barandilla.

En cambio por ejemplo en la fotografía 1 en la que nos encontramos con una considerable cantidad de detalles se ve como el resultado no es tan bueno, no pudiendo distinguir claramente los armarios.

También hay que remarcar que en espacios excesivamente abiertos el algoritmo tampoco da unos resultados óptimos como son la fotografía 2 y la 5, en las que aún se distinguen ciertas partes, como por ejemplo la profundidad del pasillo en la fotografía 2 o el suelo en la 5, pero no nos encontramos con un mapa muy denso. Esto es debido a la limitación en profundidad intrínseca en el código.

Por último, hay ciertos puntos 3D que son falsos positivos, se identifican como una serie de puntos que salen de un mismo origen formando una línea. Esto es debido a cierta imprecisión implícita en el funcionamiento del algoritmo. Para tratar de solucionar esto, se van a aplicar 3 diferentes soluciones por separado y luego de forma conjunta viendo cómo afecta cada una de estas a los resultados expuestos.

5.1. Resolución mediante aproximación

Tratando de mejorar el funcionamiento del algoritmo se va a introducir una condición en el código, basada en que como ambas fotografías se han tomado desde objetivos muy cercanos entre sí, los píxeles que representan el mismo punto tridimensional en cada imagen tienden a estar a distancias cercanas el uno del otro. Por tanto se va a definir una nueva variable, la distancia geométrica entre píxeles:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (11)$$

Siendo (x_1, y_1) el píxel en la imagen 1 y (x_2, y_2) el píxel candidato a ser un matchig en la imagen 2, si dicha distancia es menor que cierto valor umbral definido por el usuario, dicho punto será admitido como matching, si no supera dicho valor será rechazado.

Se van a realizar 3 pruebas viendo cómo afecta este parámetro a la precisión y dispersión de los resultados, por claridad las capturas de los resultados están en el Anexo D. Recopilación de resultados con la resolución mediante aproximación.

Estableciendo un valor de distancia límite de 15 píxeles obtenemos los siguientes resultados:

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	16530
2	51719	7322
3	36817	16067
4	34248	9909
5	46900	15558

Figura 43. Tabla de resultados con distancia límite de 15 píxeles

Estableciendo un valor de distancia límite de 10 píxeles obtenemos los siguientes resultados:

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	14958
2	51719	6962
3	36817	13785
4	34248	8002
5	46900	10984

Figura 44. Tabla de resultados con distancia límite de 10 píxele

Estableciendo un valor de distancia límite de 5 píxeles obtenemos los siguientes resultados:

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	4957
2	51719	4751
3	36817	6878
4	34248	3092
5	46900	5251

Figura 45. Tabla de resultados con distancia límite de 5 píxeles

Como se observa en los resultados a medida que reducimos el valor límite admisible de distancia el número de píxeles se reduce. Hay ciertos valores que son contraproducentes ya que se eliminan ciertos píxeles que eran correctos en primer lugar, por tanto para escoger este valor deberemos hacerlo de forma experimental y teniendo en cuenta que en cada par de fotografías algunos valores funcionarán mejor que otros.

Por último hay que destacar que este método no nos ha permitido eliminar el error de los falsos positivos.

5.2. Resolución aumentando del valor umbral de NCC

Como se ha visto anteriormente el valor de NCC determina como de similares son los kernels que comparamos y que cuanto más cercano sea el valor de este a 1 mayor probabilidad hay de que el píxel candidato a ser match corresponda al mismo punto tridimensional que representa en la imagen.

En la primera resolución se ha definido un valor umbral de NCC de 0.7, pero si aumentamos dicho valor a 0.8 obtenemos los siguientes resultados

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	14660
2	51719	4626
3	36817	12478
4	34248	5906
5	46900	9914

Figura 46. Tabla de resultados con el valor umbral de NCC de 0.8

Y aumentando dicho valor a 0.9:

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	9763
2	51719	1846
3	36817	7306
4	34248	2096
5	46900	3843

Figura 47. Tabla de resultados con el valor umbral de NCC de 0.9

Como se observa el aumentar el valor umbral de NCC, el algoritmo se vuelve más restrictivo eliminando más puntos de la solución, se observa una notable mejoría de las soluciones aportadas con un valor de 0.8 eliminando falsos positivos, pero en cambio para un valor de 0.9 es excesivamente restrictivito provocando una disminución muy notable de la densidad de la solución.

5.3. Resolución aumentando el tamaño del kernel

Por último también se va a ver cómo afecta el tamaño del kernel a los resultados, hay que destacar que la matriz siempre tiene que ser cuadrada y con número impar de elementos para que el píxel objeto de estudio se encuentre como elemento central de la matriz.

Se va a probar únicamente con una matriz 9x9 ya que el aumento de kernel también tiene implícito un aumento en el tiempo de cómputo, obteniendo los siguientes resultados:

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	21186
2	51719	10622
3	36817	19079
4	34248	11298
5	46900	19589

Figura 48. Tabla de resultados con un kernel de 9x9

Como se ve en los resultados en el Anexo F. Recopilación de resultados mediante el aumento del tamaño de Kernel, hay un considerable aumento de la densidad de pixeles. Esto a prior parce bueno, ya que cuanto más aumentemos el tamaño del kernel, mayor será la densidad del mapa obtenido, pero hay que tener en cuenta que también implica un aumento en los falsos positivos.

Así la mejor opción para obtener los resultados consiste en combinar las soluciones vistas en los apartados anteriores obteniendo un mapa más refinado y con una densidad considerable como se verá a continuación.

5.4. Resolución con combinación de métodos

Como se ha visto la mejor opción para obtener un mapa de profundidad es la combinación de los distintos métodos aplicados anteriormente.

Con esta idea vamos a utilizar la siguiente configuración de parámetros:

-Distancia límite de 10 píxeles

-Valor umbral de NCC de 0.8

- Kernel de 9x9 obteniendo los siguientes resultados:

Par de imágenes	Features encontradas	Píxeles con matching
1	35989	13389
2	51719	4946
3	36817	12233
4	34248	5187
5	46900	8050

Figura 49. Tabla de resultados con combinación de métodos

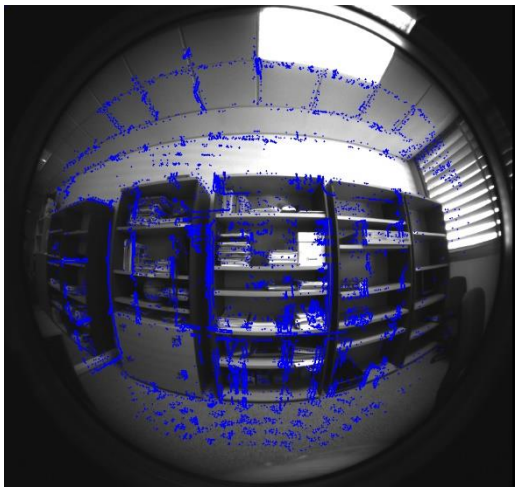


Figura 50. Matching fotométrico (1)

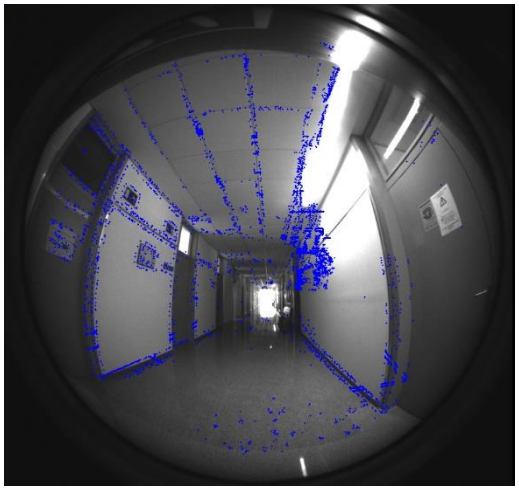


Figura 51. Matching fotométrico(2)

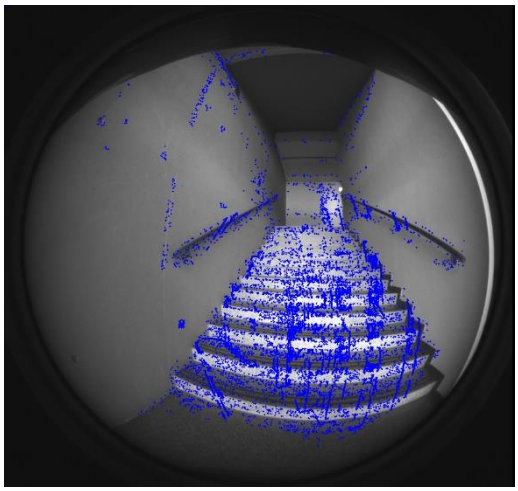


Figura 52. Matching fotométrico (3)

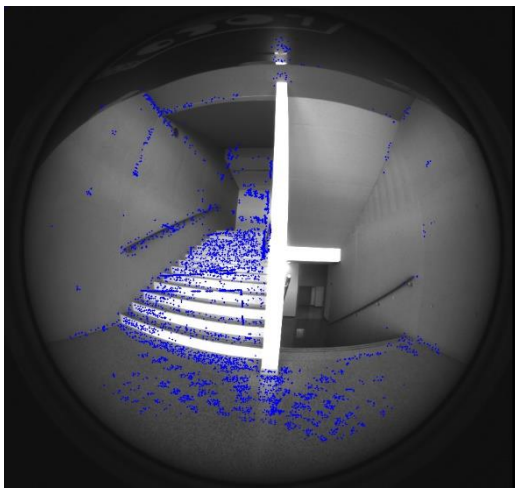


Figura 53. Matching fotométrico (4)

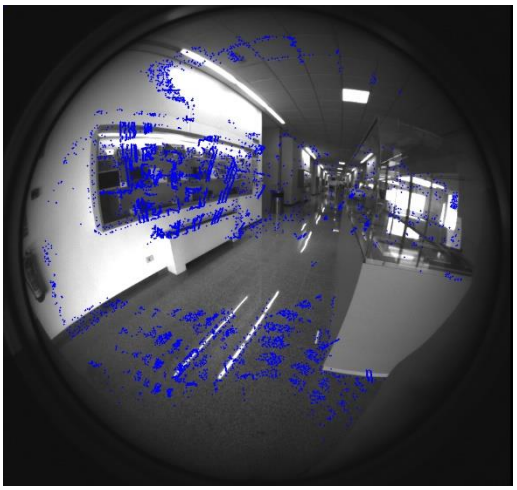


Figura 54. Matching fotométrico (5)

- Resultado fotografía I:

Figura 55. Nube de puntos 1.1

Figura 56. Nube de puntos 1.2

- Resultados fotografía II:

Figura 57. Nube de puntos 2.1

Figura 58. Nube de puntos 2.2

- Resultados fotografía III:

Figura 59. Nube de puntos 3.1

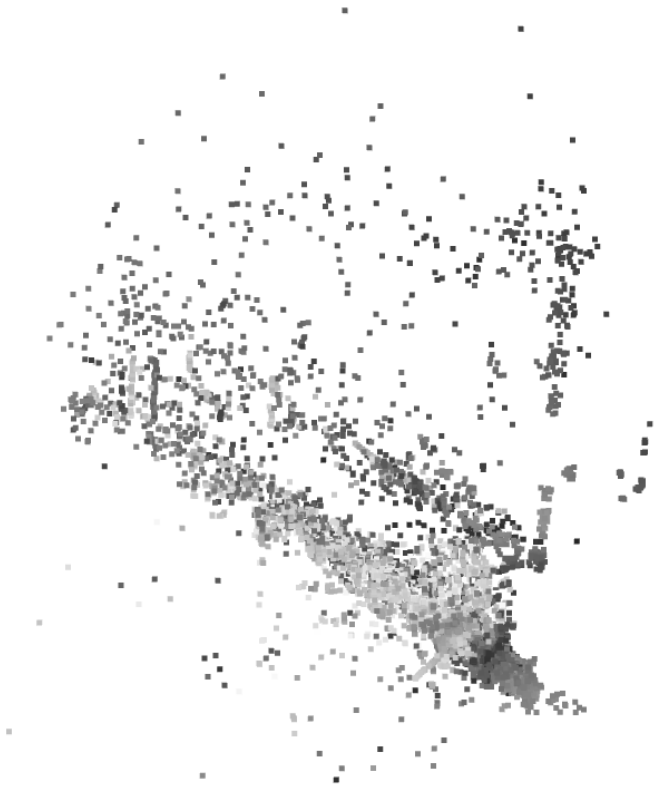


Figura 60. Nube de puntos 3.2

- Resultados fotografía IV:

Figura 61. Nube de puntos 4.1

Figura 62. Nube de puntos 4.2

- Resultados fotografía V:

Figura 63. Nube de puntos 5.1

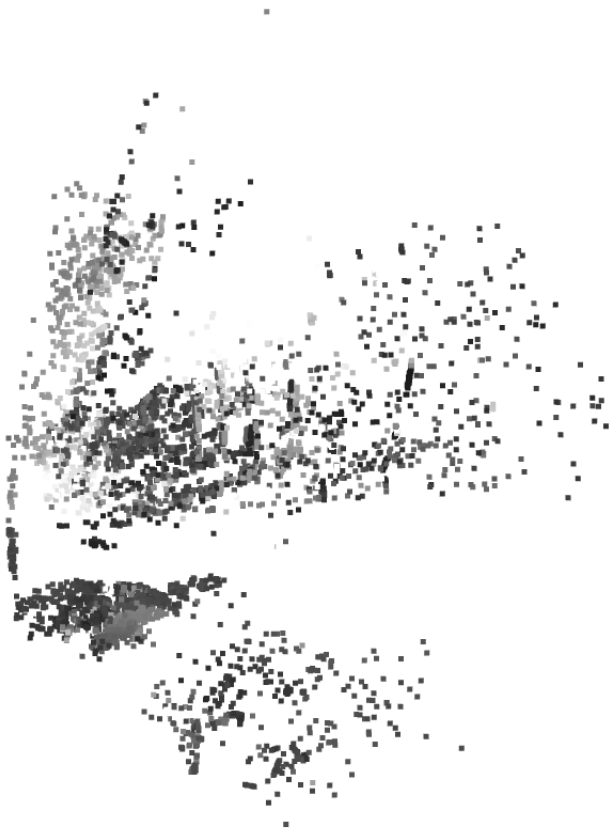


Figura 64. Nube de puntos 5.2

6. Conclusiones

Como se ha visto el cálculo de un mapa de profundidad semidenso es una tarea compleja, pero de gran importancia en el contexto de la visión por computador.

A través de este trabajo también se ha demostrado la importancia que tiene el modelo de cámara en el diseño del algoritmo, así como la necesidad de una buena calibración tanto en los parámetros intrínsecos como extrínsecos para obtener unos resultados precisos.

Si se busca obtener una solución de calidad, es imprescindible resaltar la importancia de utilizar técnicas de correspondencia estero que sean robustas y precisas.

Por último se ha visto como el desempeño del algoritmo está ligado a ciertos parámetros, pudiendo mejorar los resultados variando estos parámetros.

A modo de resumen en la siguiente tabla se presentan todos los resultados obtenidos a lo largo de este trabajo:

Ensayo inicial		
Par de imágenes	Features encontradas	Píxeles con matching
1	35989	19247
2	51719	8988
3	36817	17284
4	34248	10432
5	46900	17401
Distancia límite de 15 píxeles		
1	35989	16530
2	51719	7322
3	36817	16067
4	34248	9909
5	46900	15558
Distancia límite de 10 píxeles		
1	35989	14958
2	51719	6962
3	36817	13785
4	34248	8002
5	46900	10984
Distancia límite de 5 píxeles		
1	35989	4957
2	51719	4751
3	36817	6878
4	34248	3092
5	46900	5251
Valor umbral de NCC = 0.8		
1	35989	14660
2	51719	4626
3	36817	12478
4	34248	5906
5	46900	9914

Valor umbral de NCC = 0.9		
1	35989	9763
2	51719	1846
3	36817	7306
4	34248	2096
5	46900	3843
Tamañio de kernel 9x9		
1	35989	21186
2	51719	10622
3	36817	19079
4	34248	11298
5	46900	19589
Resolución final		
1	35989	13389
2	51719	4946
3	36817	12233
4	34248	5187
5	46900	8050

En cuanto a las líneas futuras de investigación, se propone el uso combinando del cálculo de mapas de profundidad de forma geométrica y con redes neuronales, tratando de obtener un mapa de mayor densidad sacrificando la precisión, así como también realizar la combinación de mapas de profundidad generados por diferentes cámaras para obtener una imagen tridimensional más compleja y precisa.

Bibliografía

[1] A. Eichenseer & A.Kaup, “A data set providing synthetic and real-world fisheye video sequences”, Multimedia Communications and Signal Processing, Freiderich-Alexander University Erlangen-Nürnberg, Erlangen, Germany, 2016.

[2] J. Bermúdez, “Calibración de Sistemas Catadióptricos de Visión Omnidireccional”, PFC, Dept. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, España, 2009.

[3] L. Fernandez, V. Avila & L. Gonçalves, “A Generic Approach of Error Estimation of Depth Data from (Stereo and RGB-D) 3D Sensors”, Universidade Federal do Rio Grande do Norte, Brasil, 2017.

[4] J. Kannala & S. S. Brandt, “A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses”, University of Copenhagen, Denmark, University of Oulu, Finland, 2006.

[5] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed., 2021.

[6] A. Nordmann, Epipolar geometry, 2007.

[7] B. Mičušík, “Two-View Geometry of Omnidirectional Cameras”, Ph.D., Dept. of Cybernetics, Czech Technical University in Prague, Czech Republic, 2004.

[8] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, Dept. of Computer Science, University of British Columbia, Canada, 2004

Anexo A. Parámetros Intrínsecos y Extrínsecos de una cámara

Los parámetros intrínsecos definen la geometría interna de la cámara, describen las características físicas de la cámara y cómo esta transforma el mundo tridimensional en una imagen bidimensional, dichos parámetros se agrupan en una matriz triangular superior 3x3 denominada la matriz de calibración , esta matriz de calibración es distinta para cada cámara y de forma general se puede definir como:

$$\boldsymbol{K} = \begin{pmatrix} f & s & c_x \\ 0 & af & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Donde f es la distancia focal entre el centro óptico y el centro del plano de la imagen, s es un factor define la inclinación entre los ejes, a es la relación de aspecto que define la relación entre el ancho y el alto de la imagen., y $\boldsymbol{C} = (c_x, c_y)^T$ es el centro de la imagen expresado en coordenadas de píxel. En la mayoría de las ocasiones $a = 1$ y $s = 0$ como en nuestro caso, dando lugar a las siguientes matrices de calibración (una para cada objetivo):

$$\boldsymbol{K}_1 = \begin{pmatrix} 286.9510 & 0 & 420.1034 \\ 0 & 286.9510 & 402.1838 \\ 0 & 0 & 1 \end{pmatrix} \quad (12)$$

$$\boldsymbol{K}_2 = \begin{pmatrix} 286.7023 & 0 & 428.1151 \\ 0 & 286.6902 & 399.3409 \\ 0 & 0 & 1 \end{pmatrix} \quad (13)$$

Se debe tener en cuenta que a la hora de realizar la parte computacional de este TFG no se han utilizado exactamente estos parámetros ya que los parámetros mostrados en las matrices (12) y (13) están aproximados al cuarto decimal.

Los parámetros extrínsecos describen la posición y orientación de la cámara en relación con el mundo tridimensional en coordenadas homogéneas, y esta relación se expresa mediante una matriz de transformación 4x4:

$$\boldsymbol{T} = [\boldsymbol{R}|\mathbf{t}] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (14)$$

En nuestro caso tendremos dos matrices de transformación, una para cada cámara, que relaciona el sistema de referencia de dichas cámaras con el sistema W localizado entre las dos cámaras:

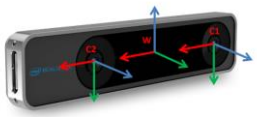


Figura 65. Referencias del sistema estéreo.

Anexo B. Scale Invariant Feature Transform (SIFT) [8]

Es un algoritmo empleado para obtener las características más relevantes de una imagen, y nos va a permitir emparejar puntos (píxeles) en imágenes con diferentes puntos de vista (cámara estéreo).

Las características obtenidas son invariantes a la escala y la rotación de la imagen y parcialmente invariantes a la iluminación y el punto de vista de la cámara. Lo que nos garantiza que los puntos se puedan detectar de forma confiable y con mucha exactitud.

Este algoritmo se realiza en 4 etapas:

1. Detección de extremos en el espacio de escala

En este primer paso se buscan potenciales puntos de interés a través de diferentes escalas y a lo largo de todo el espacio de la imagen, se ejecuta de forma eficiente gracias al uso de la diferencia gaussiana, así obtenemos puntos característicos estables e independientes de la escala.

2. Localización de puntos de interés

Se refina la localización de cada punto de interés utilizando la expansión de serie de Taylor de la escala-espacio, si el valor del punto es menor a un cierto umbral el punto es descartado eliminado así los puntos con un contraste bajo, también se eliminan los puntos a lo largo de los bordes ya que la diferencia gaussiana tiene una fuerte respuesta en estos mismos.

3. Asignación de orientaciones

En este paso se asigna una orientación a cada punto de interés basándonos en el gradiente de direcciones locales, todas las operaciones que se realicen a partir de ahora utilizaran esta orientación asignada, de esta forma logramos que los puntos de interés sean invariantes a la rotación de las imágenes.

4. Descriptor de puntos de interés

Se miden los gradientes locales alrededor de un punto de interés a una escala seleccionada, para cada punto se toma 16x16 puntos que se reducen a bloques de 4x4 y se crea un histograma de orientaciones. La concatenación en un vector de los valores de las cajas de cada histograma para los 16 bloques del punto de interés constituye el descriptor, permitiendo identificar estos puntos con cambios de iluminación.

B.1. Detección de extremos en el espacio de escala

Buscamos localizaciones que puedan ser asignadas de forma robusta, en diferentes situaciones de puntos de vista y escalas. Para lograr esto debemos encontrar características estables a través de todas las posibles escalas.

Para ello generamos el espacio de escala de las diferentes octavas de la imagen original a través de la función:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Donde $G(x, y, \sigma)$ es un filtro Gaussiano , $I(x, y)$ es la imagen y * es el operador de convolución, una vez obtenidas las diferentes imágenes gaussianas calculamos el espacio de

escala extremo a través de las diferencias de gaussianas (DoG), las cuales se calculan como la diferencia de dos escalas separadas por un factor constante k:

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y)$$

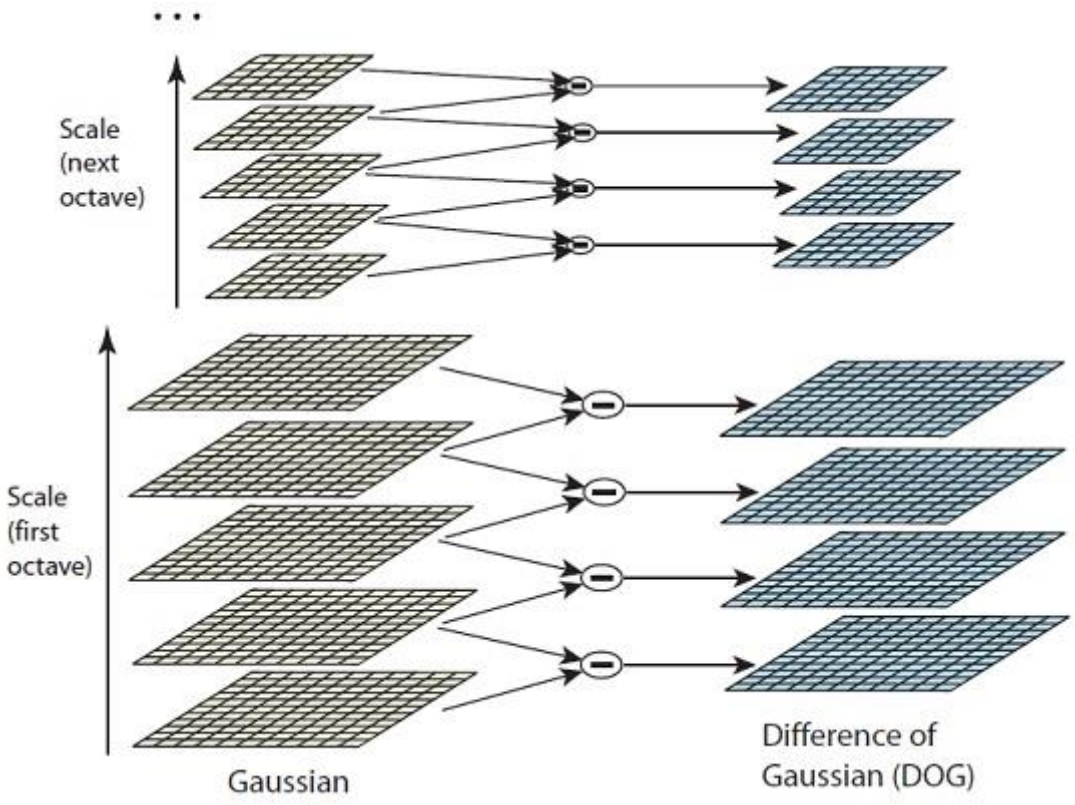


Figura 66. Espacio de escala para cada octava, realizando la diferencia de los Guassianos obtenemos las Diferencias Gaussianas en los que busacaremos los máximos y los mínimos.[8]

Una vez obtenido las DoG debemos buscar los máximos y mínimos de la función $D(x, y, \sigma)$, para ello cada punto se compara con los 8 puntos adyacentes y con los nueve de la escala superior e inferior comparándose con un total de 26 puntos. Si es mayor o menor que todos estos el punto es seleccionado como un posible punto de interés así obtenemos puntos independientes a la escala y con un alto contraste.

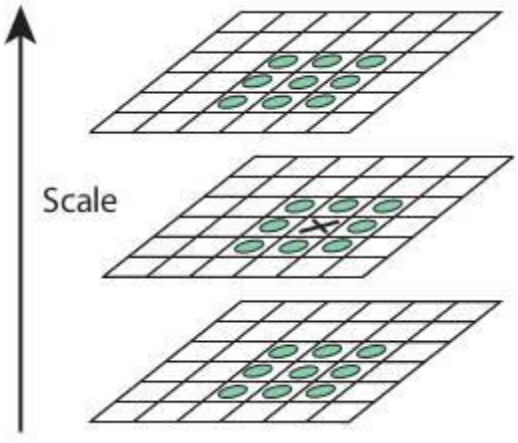


Figura 67. Puntos de comparación para obtener un máximo o un mínimo.[8]

B.2. Localización de puntos de interés

En este paso se logrará un refinamiento de los puntos de interés a través de un análisis más detallado de su localización, escala y la proporcionalidad de sus curvaturas principales, esto nos permitirá eliminar puntos con un bajo contraste y los que están localizados sobre los bordes.

Para esto se sustituye la localización de los posibles puntos candidatos en una función cuadrática 3D determinando la localización interpolada del máximo, se emplea la Serie de Taylor hasta el termino cuadrático de la función del espacio de escala $D(x, y, \sigma)$ desplaza para que el origen sea el punto candidato.

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x$$

Esta función es evaluada en el punto candidato y $x = (x, y, \sigma)^T$ es la distancia hasta este punto. La localización del extremo, $\hat{x}$, se calcula derivando esta función respecto a x e igualando a cero, obteniendo:

$$\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}$$

Se aplica una aproximación en diferencias finitas para facilitar el tiempo de computación y el problema resultante es un sistema lineal. Una vez calculado se sustituye este valor en la función tal que:

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}$$

Y si el valor de esta función es menor que un valor umbral fijado por el usuario el extremo es descartado.

Para asegurar una mayor estabilidad también debemos eliminar puntos localizados a lo largo de los bordes ya que la función de diferencias de gaussianas tiene una respuesta fuerte a lo largo de estos, y puede provocar la localización de puntos inadecuados.

En este caso los puntos definidos a lo largo de los bordes tienen una gran curvatura principal, esta curvatura puede ser calculada con a la matriz Hessiana evaluada en la localización y a la escala del punto candidato. Al igual que en el caso anterior las derivadas se aproximan con diferencias finitas:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Los valores propios de esta matriz son proporcionales a la curvatura principal de D. Podemos facilitar el cálculo ya que únicamente nos interesa la proporción entre los valores propios.

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

Donde α es el valor propio mayor y β es el valor propio menor y r es la proporción entre el valor propio mayor y el valor propio menor $\alpha = r\beta$

Por tanto para eliminar los puntos que no nos interesan debemos fijar un umbral r y ver si nuestro punto cumple la siguiente desigualdad, si no la cumple eliminamos el punto:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

B.3. Asignación de orientaciones

Asignando una orientación a los puntos de interés basada en las propiedades locales, lograremos que este punto sea invariante a la rotación de la imagen. Para lograr esto se selecciona la imagen más alisada por el filtro Gaussiano en la escala en la que se encuentra el punto de interés, y para cada muestra de la imagen se calcula la magnitud del gradiente, $m(x,y)$ y la orientación, $\theta(x,y)$:

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-q,y)\big)^2 + \big(L(x,y1) - L(x,y-1)\big)^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-q,y)}\right)$$

Se crea un histograma de orientaciones con los gradientes de los puntos que se encuentran alrededor del punto de interés.

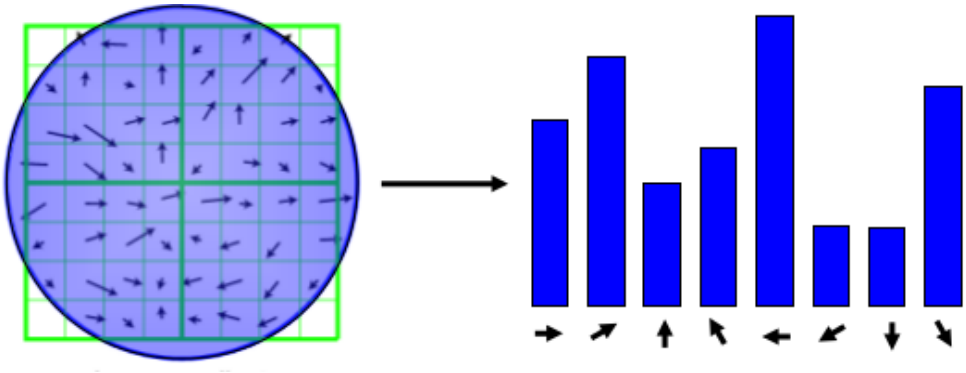


Figura 68. Histogrma de orientaciones.[8]

Los picos del histograma de orientación se corresponden con las direcciones dominantes de los gradientes locales. Se detecta el pico más alto y cualquier otro pico del histograma que se encuentre dentro del 80% del más alto se emplea para crear otro punto clave con otra orientación, por lo que tendremos puntos clave con la misma localización y escala pero distintas orientaciones. Por último una parábola se ajusta a los 3 valores más cercanos al pico y se interpola para obtener un nuevo pico con mayor precisión.

B.4. Descriptor de putos de interés

En los pasos previos se han asignado a los puntos de interés localización, escala y orientación, estos parámetros nos permiten generar un sistema local 2D coordenado y repetible describiendo una región de la imagen lo que da lugar a que podamos identificar los mismos puntos en imágenes diferentes. Lo último que debemos hacer es generar un descriptor local de la región de la imagen que se distintivo e invariante a los cambios de iluminación o de punto de vista.

Para calcular este descriptor se hace algo muy similar a lo visto en el apartado anterior, para cada punto de interés se toma 16x16 puntos alrededor de este y se calcula el histograma de

orientaciones, pero esta vez los gradientes calculados se rotan a la posición relativa del punto de interés para lograr la invarianza en orientación.

Se ponderan los puntos dependiendo de los lejos que se encuentren del punto clave , esto se realiza para dar menos importancia a los gradientes que se encuentran lejos del descriptor, ya que suelen ser los más afectados por errores de registro.

El descriptor se forma a partir de los valores de orientación del histograma, resultando en un array 4x4 que contiene 8 orientaciones.

Por último el vector es modificado para reducir los efectos de los cambios de iluminación.

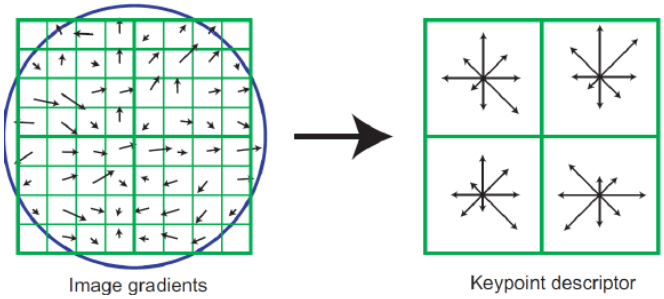


Figura 69. Formación del descriptr a traves del histograma local de orientaciones.[8]

Anexo C. Canny Edge Detector

El método Canny es un algoritmo utilizado en visión por computador que identifica los bordes de las imágenes. Para ello, primero suaviza la imagen original a través de un filtro Gaussiano (en la librería OpenCV se utiliza un filtro Gaussiano de 5x5) eliminado ruido y variaciones de intensidad mejorando así la detección de bordes.

Se calculan los gradientes de intensidad de la imagen, aplicando el operador Sobel, que calcula el gradiente de la imagen en las direcciones verticales y horizontales. Para calcular dicho gradiente se realiza la convolución de la imagen con dos kerenels 3x3 para calcular las aproximaciones de las derivadas:

$$\boldsymbol{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \boldsymbol{A} \tag{15}$$

$$\boldsymbol{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \boldsymbol{A} \tag{16}$$

Combinando el gradiente horizontal con el gradiente vertical obtenemos la magnitud del gradiente, expresado como:

$$\boldsymbol{G} = \sqrt{\boldsymbol{G}_x^2 + \boldsymbol{G}_y^2} \tag{17}$$

Y también se calcula la dirección del gradiente, como:

$$\boldsymbol{\theta} = atan2(\boldsymbol{G}_y, \boldsymbol{G}_x) \tag{18}$$

Se obtienen dos imágenes, una que representa la magnitud del gradiente y otra que indica la dirección de dicho gradiente en cada punto.

Una vez obtenida la magnitud y la dirección del gradiente se realiza un análisis de la imagen para suprimir los píxeles que no corresponde al borde, para cada píxel se compara su magnitud de gradiente con los píxeles contiguos en la dirección del gradiente y solo se mantiene el píxel si tiene la magnitud de gradiente más alta respecto de los contiguos, si no es descartado.

Se aplican dos valores límite, uno superior y otro inferior para clasificar los píxeles, los píxeles con una magnitud de gradiente mayor al umbral superior se clasifican como píxeles pertenecientes a algún borde, los píxeles con una magnitud de gradiente menor al umbral inferior se considera que no pertenecen al borde y se descartan y por último los píxeles con una magnitud de gradiente entre el umbral inferior y el umbral superior, se les aplica un proceso de histéresis.

Este proceso de histéresis consiste en ver si los píxeles que se encuentra entre el valor límite superior y el valor límite inferior están conectados a píxeles pertenecientes al borde, si es así dicho píxel se clasifica como parte del borde, si no tiene ningún píxel cercano considerado como borde se descarta, este se repite hasta que dejen de existir píxeles no identificados, este proceso mejora la continuidad de los bordes y eliminar los bordes espurio

Anexo D. Recopilación de resultados con la resolución mediante aproximación

Resultados con valor de distancia límite de 15 píxeles:

- Resultados fotografía I:

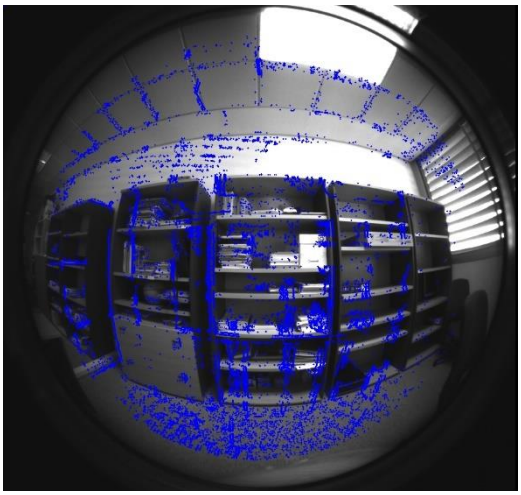


Figura 70. Matching fotométrico (I) con distancia de 15 píxeles

Figura 71. Nube de puntos 1.1 con distancia de 15 píxeles

Figura 72. Nube de puntos1.2 con distancia de 15 píxeles

- Resultados fotografía II:

Figura 74. Nube de puntos 2.1. con distancia de 15 píxeles

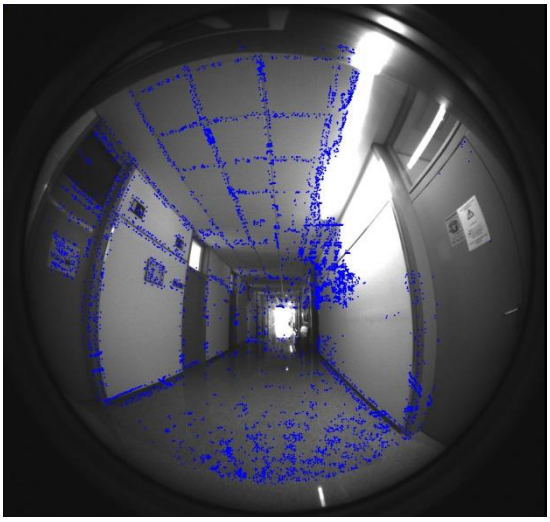


Figura 73. Matching fotométrico (II) con distancia de 15 píxeles

Figura 75. Nube de puntos 2.2 con distancia de 15 píxeles

- Resultados fotografía III:

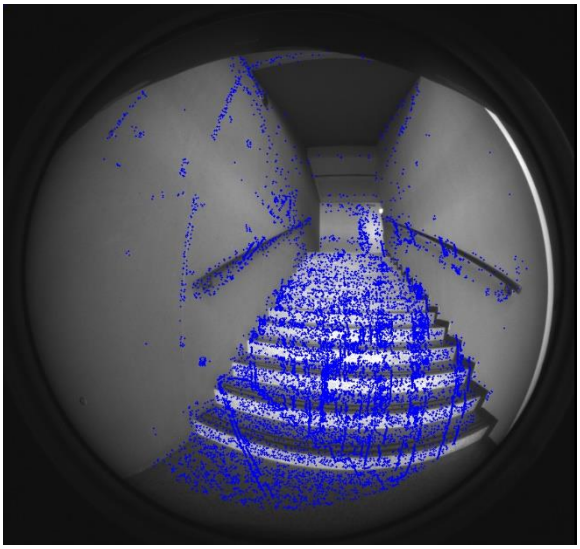


Figura 76. Matching fotométrico (III) con distancia de 15 píxeles

Figura 77. Nube de puntos 3.1 con distancia de 15 píxeles

Figura 78. Nube de puntos 3.2 con distancia de 10 píxeles

- Resultados fotografía IV:

Figura 80. Nube de puntos 4.1 con distancia de 15 píxeles

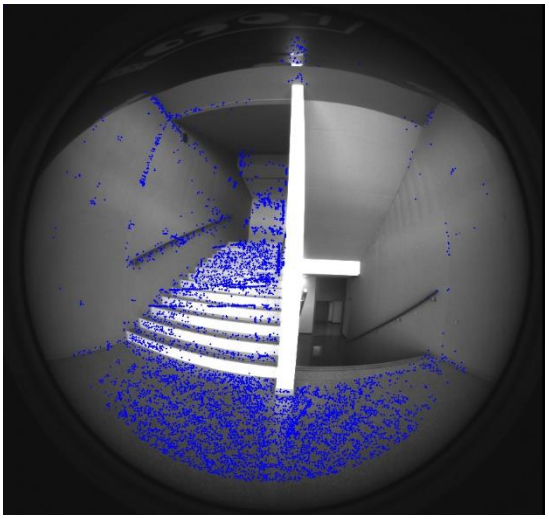


Figura 79. Matching fotométrico (IV) con distancia de 15 píxeles

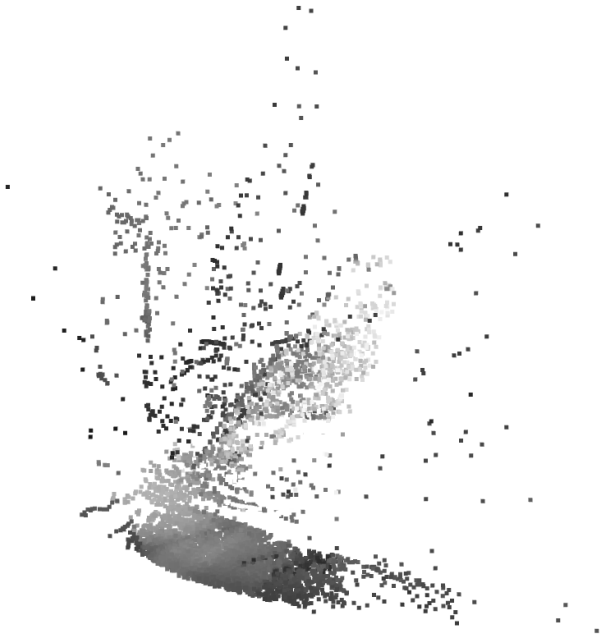


Figura 81. Nube de puntos 4.2 con distancia de 15 píxeles

- Resultados fotografía V:

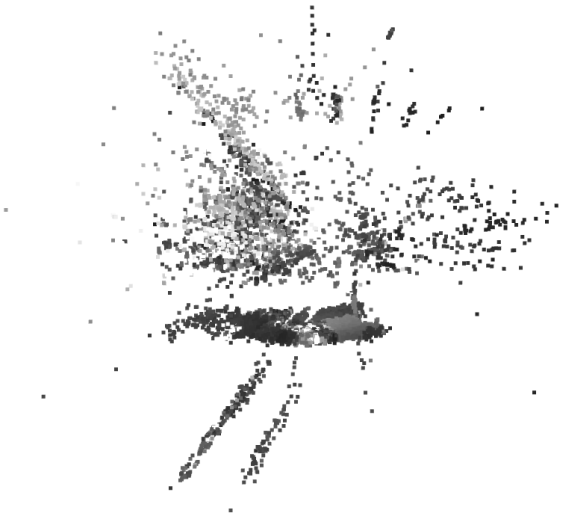


Figura 83. Nube de puntos 5.1 con distancia de 15 píxeles

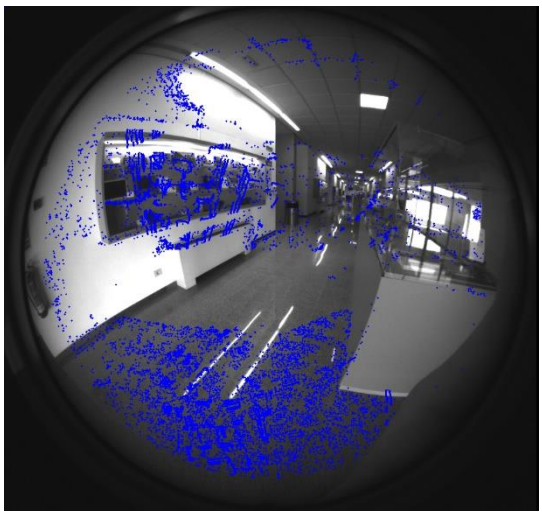


Figura 82. Matching fotométrico (V) con distancia de 15 píxeles

Figura 84. Nube de puntos 5.2 con distancia límite de 15 píxeles

Resultados con valor de distancia límite de 10 píxeles:

- Resultados fotografía I:

Figura 86. Nube de puntos 1.1 con distancia límite de 5 píxeles

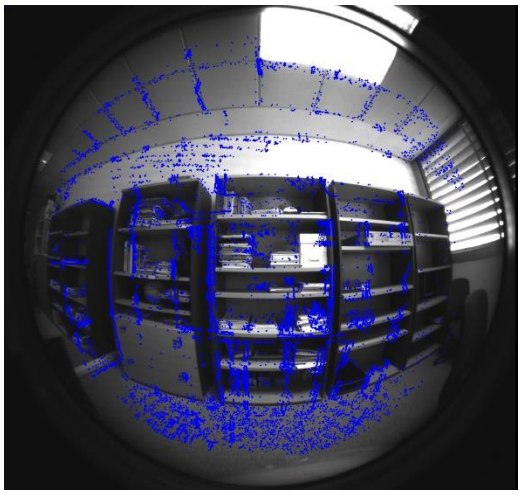


Figura 85. Matching fotométrico (I) con distancia de 10 píxeles

Figura 87. Nube de puntos 1.2 con distancia límite de 10 píxeles

- Resultados fotografía II:

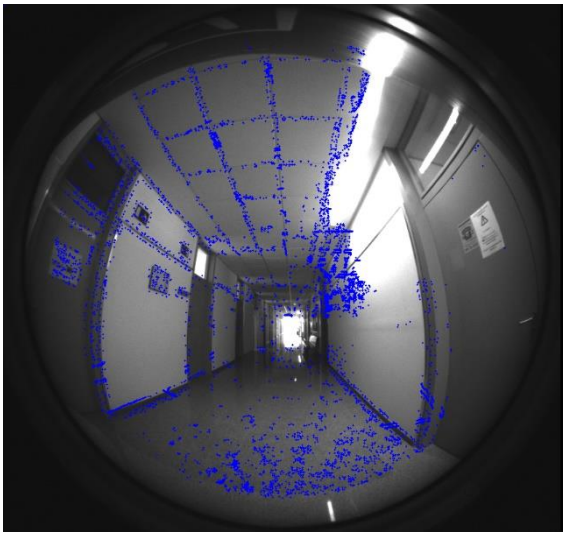


Figura 88. Matching fotométrico (II) con distancia de 10 píxeles

Figura 89. Nube de puntos 2.1 con distancia límite de 10 píxeles

Figura 90. Nube de puntos 2.2 con distancia límite de 10 píxeles

- Resultados fotografía III:

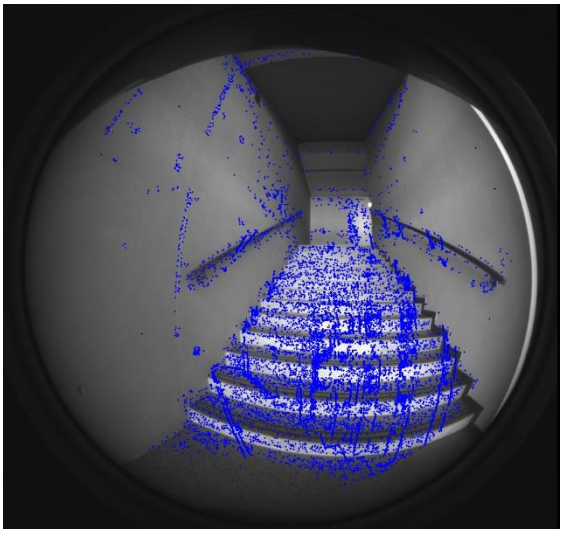


Figura 91. Matching fotométrico (III) con límite de 10 píxeles

Figura 92. Nube de puntos 3.1 con distancia límite de 10 píxeles

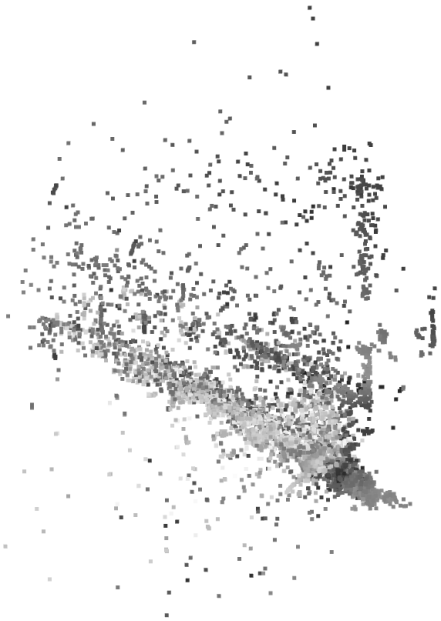
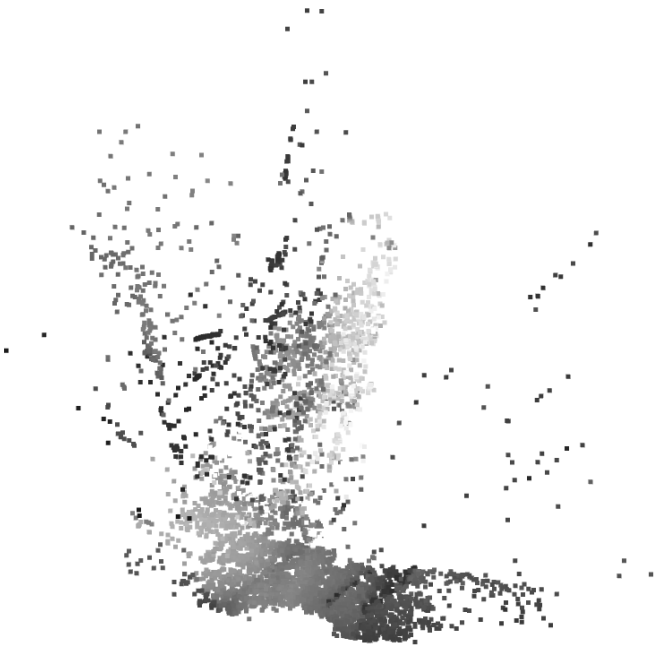


Figura 93. Nube de puntos 3.2 con distancia límite de 10 píxeles

- Resultados fotografía IV:



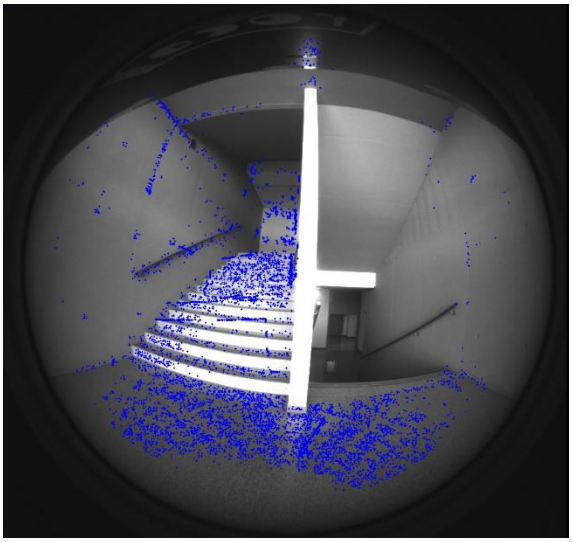


Figura 94. Matching fotométrico (IV) con límite de 10 píxeles

Figura 95. Nube de puntos 4.1 con distancia límite de 10 píxeles

Figura 96. Nube de puntos 4.1 con distancia límite de 10 píxeles

- Resultados fotografía V:

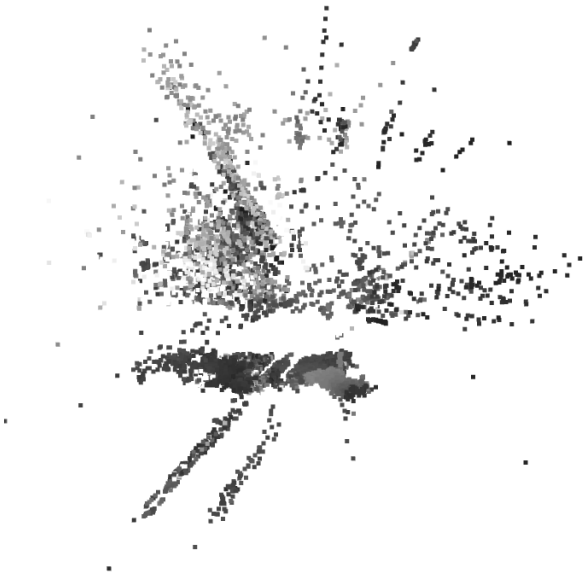


Figura 98. Nube de puntos 5.1 con distancia límite de 10 píxeles

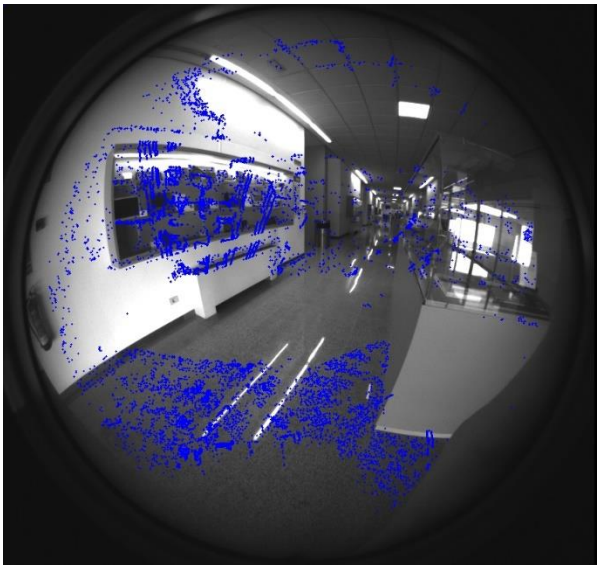


Figura 97. Matching fotométrico (V) con límite de 10 píxeles

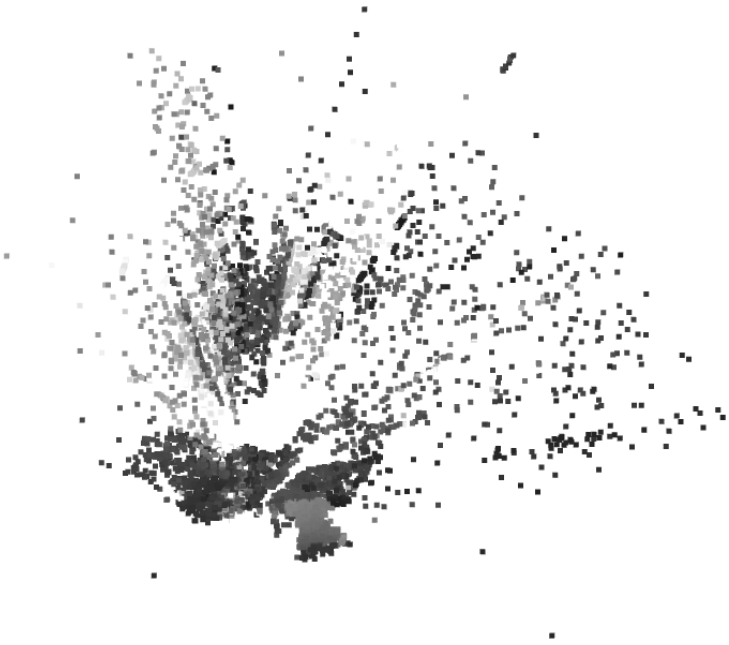


Figura 99. Nube de puntos 5.2 con distancia límite de 10 píxeles

Resultados con valor de distancia límite de 5 píxeles:

- Resultados fotografía I:

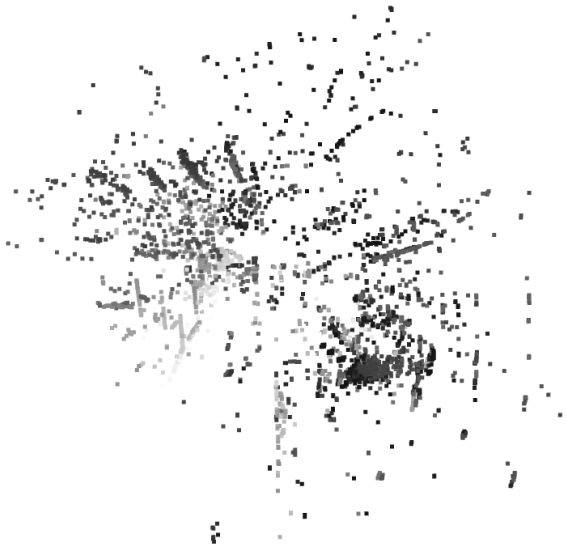


Figura 101. Nube de puntos 1.1 con distancia límite de 5 píxeles

Figura 100. Matching fotométrico (I) con límite de 5 píxeles

Figura 102. Nube de puntos 1.1 con distancia límite de 5 píxeles

- Resultados fotografía II:

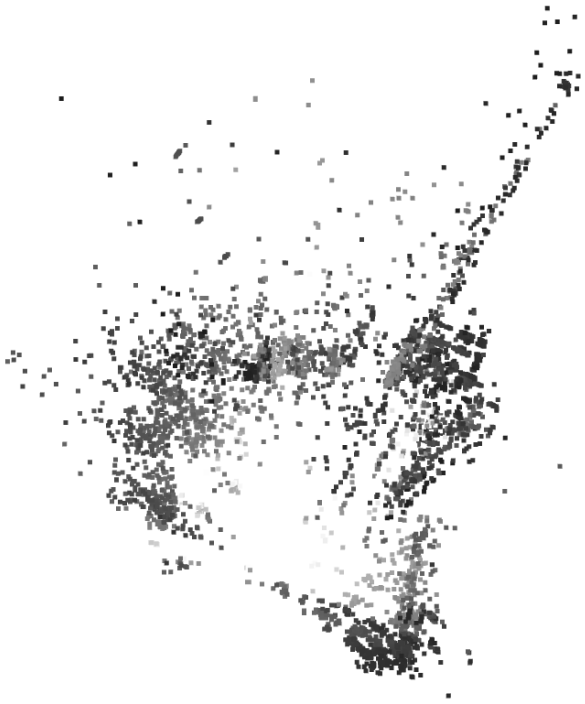


Figura 104. Nube de puntos 2.1 con distancia límite de 5 píxeles

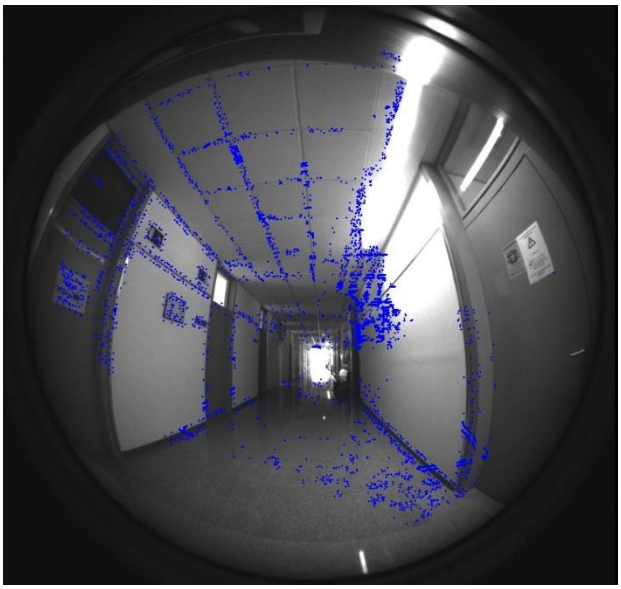


Figura 103. Matching fotométrico (II) con límite de 5 píxeles

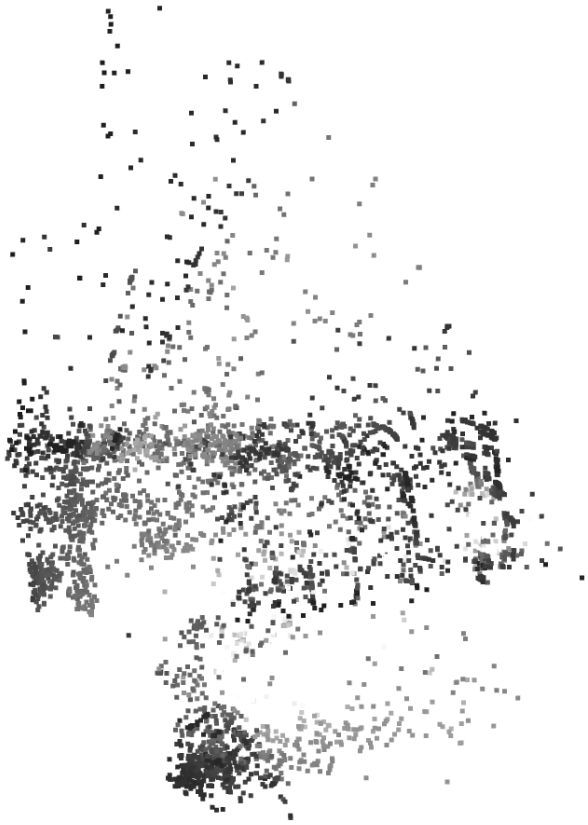


Figura 105. Nube de puntos 5.2 con distancia límite de 5 píxeles

- Resultados fotografía III:

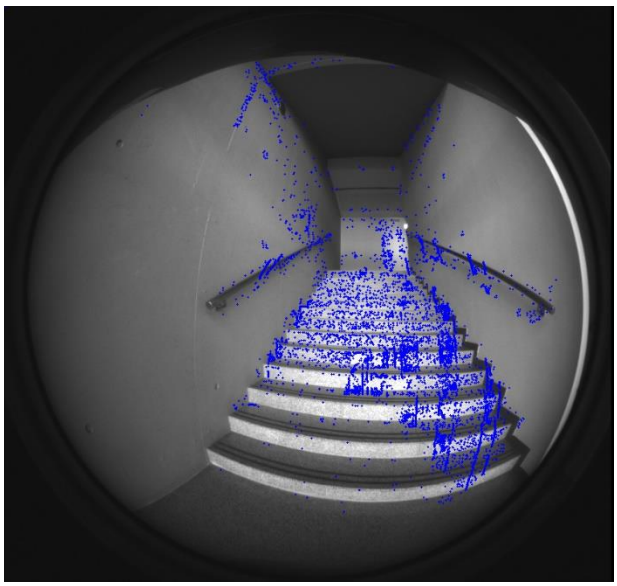


Figura 106. Matching fotométrico (III) con límite de 5 píxeles

Figura 107. Nube de puntos 3.1 con distancia límite de 5 píxeles

Figura 108. Nube de puntos 3.2 con distancia límite de 5 píxeles

- Resultados fotografía IV:

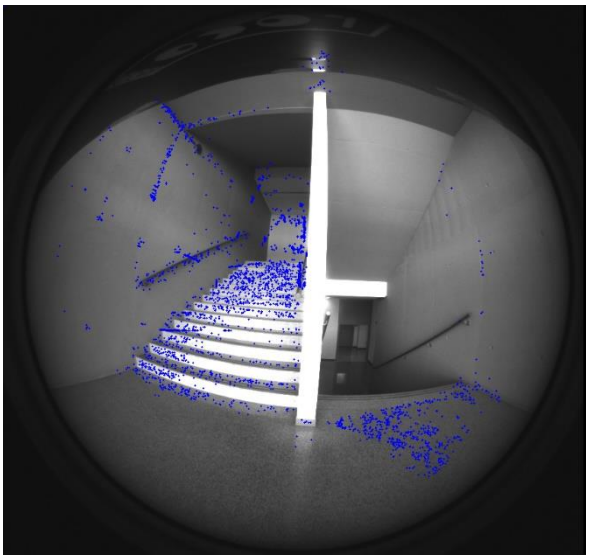


Figura 109. Matching fotométrico con límite de 5 píxeles

Figura 110. Nube de puntos 4.1 con distancia límite de 5 píxeles

Figura 111. Nube de puntos 4.2 con distancia límite de 5 píxeles

- Resultados fotografía V:

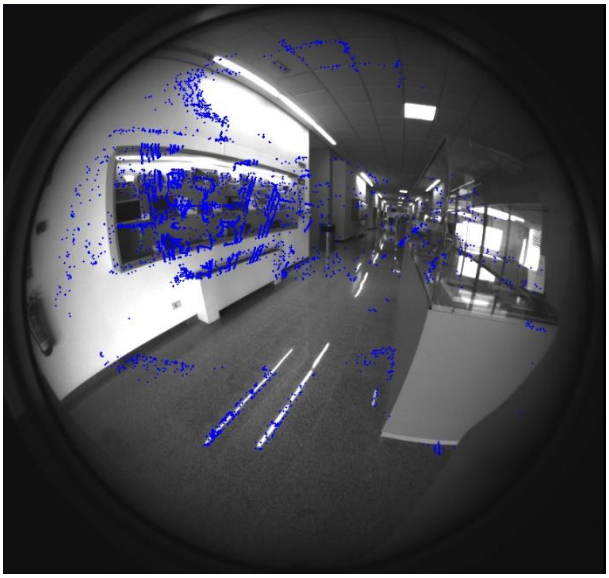


Figura 112. Matching fotométrico con límite de 5 píxeles

Figura 113. Nube de puntos 5.1 con distancia límite de 5 píxeles

Figura 114. Nube de puntos 5.2. con distancia límite de 5 píxeles

Anexo E. Recopilación de resultados mediante el aumento del valor umbral de NCC

Para un valor umbral de NCC de 0.8

- Resultados fotografía I:

Figura 115. Matching fotométrico (I) con valor umbral de NCC de 0.8

Figura 116. Nube de puntos 1.1 con valor umbral de NCC de 0.8

Figura 117. Nube de puntos 1.2 con valor umbral de NCC de 0.8

- Resultados fotografía II:

Figura 119. Nube de puntos 2.1 con valor umbral de NCC 0.8

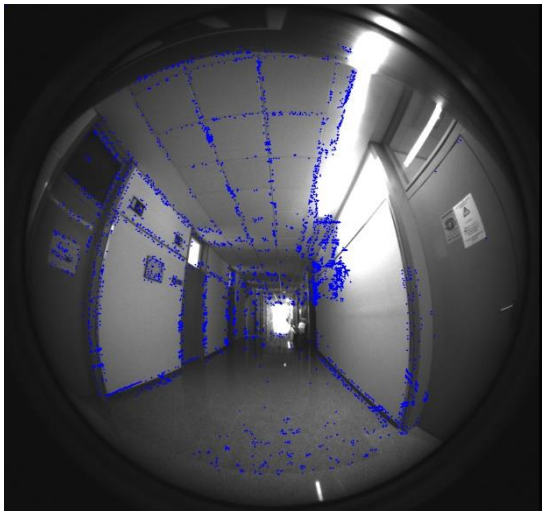


Figura 118. Matching fotométrico (II) con valor umbral de NCC de 0.8

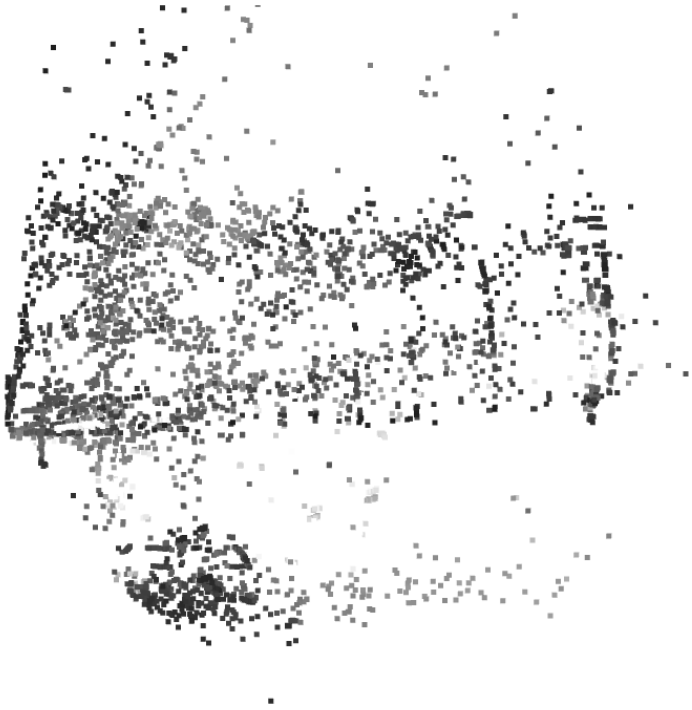


Figura 120. Nube de puntos 2.1 con valor umbral de NCC 0.8

- Resultados fotografía III:

Figura 122. Nube de puntos 3.1 con valor umbral de NCC 0.8

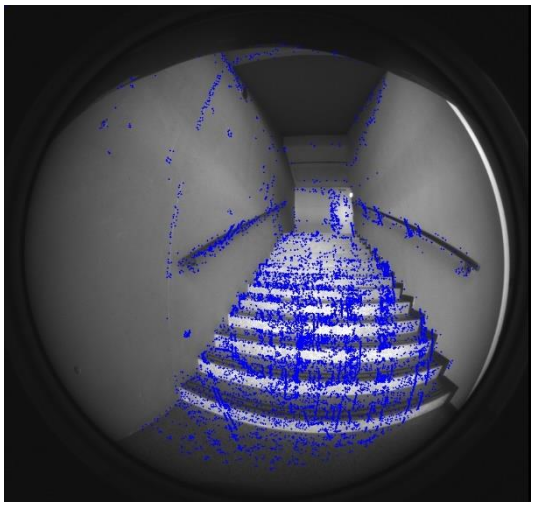
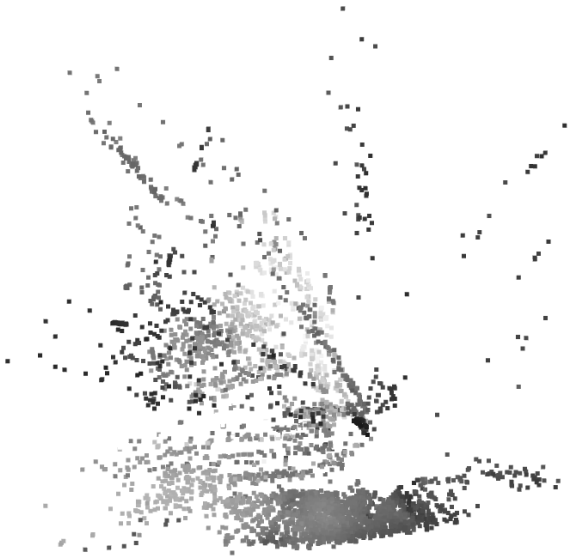


Figura 121. Matching fotométrico (III) con valor umbral de NCC de 0.8

Figura 123. Nube de puntos 2.2 con valor umbral de NCC 0.8

- Resultados fotografía IV:



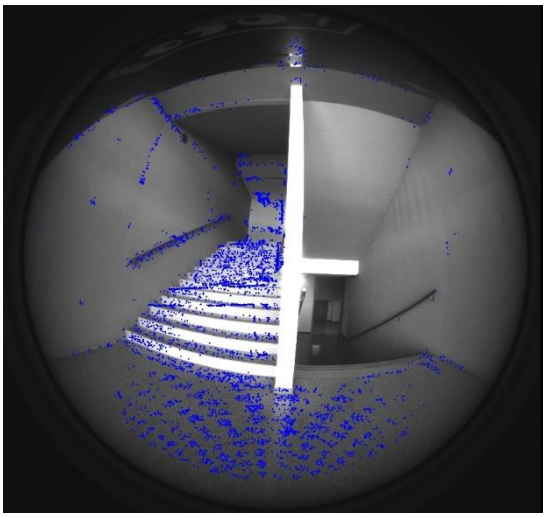


Figura 124. Matching fotométrico (IV) con valor umbral de NCC de 0.8

Figura 125. Nube de puntos 4.1 con valor umbral de NCC 0.8

Figura 126. Nube de puntos 4.2 con valor umbral de NCC 0.8

- Resultados fotografía V:

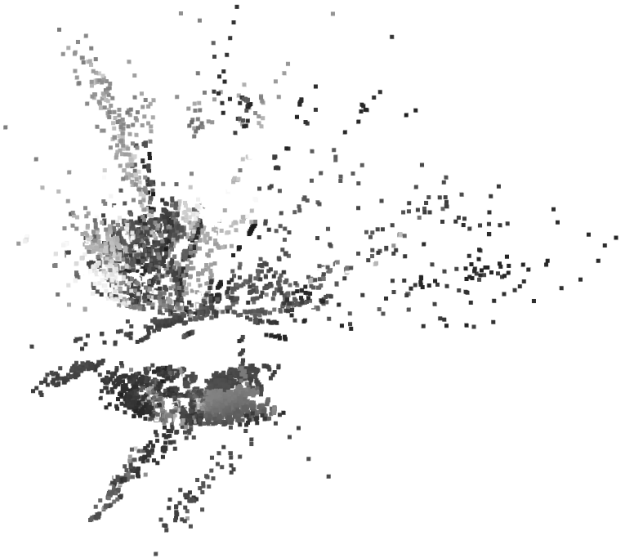


Figura 128. Nube de puntos 5.1 con valor umbral de NCC 0.8

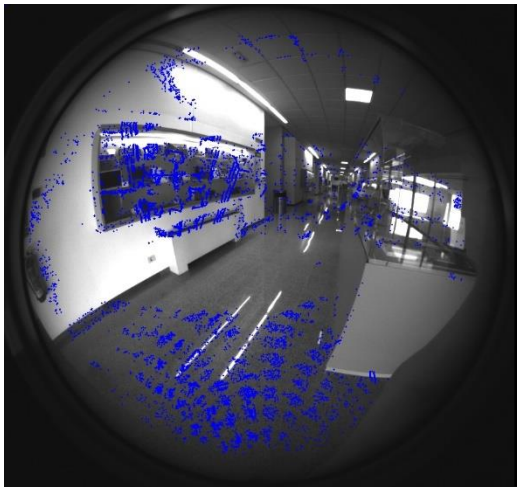


Figura 127. Matching fotométrico (V) con valor umbral de NCC de 0.8

Figura 129. Nube de puntos 5.2 con valor umbral de NCC 0.8

Para un valor umbral de NCC de 0.9

- Resultados fotografía I:

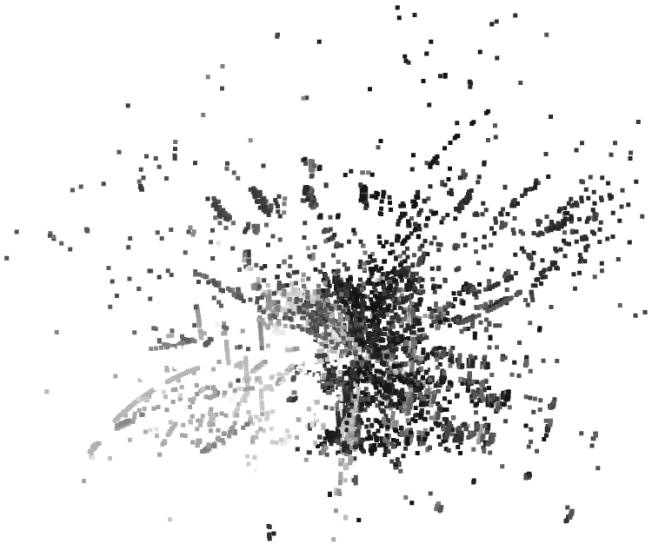


Figura 131. Nube de puntos 1.1 con valor umbral de NCC 0.9

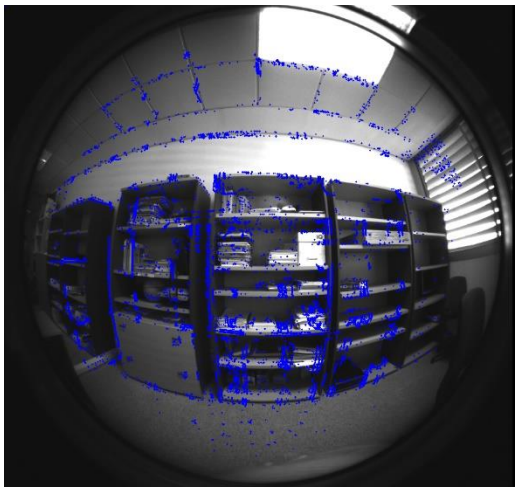


Figura 130. Matching fotométrico (I) con valor umbral de NCC de 0.9

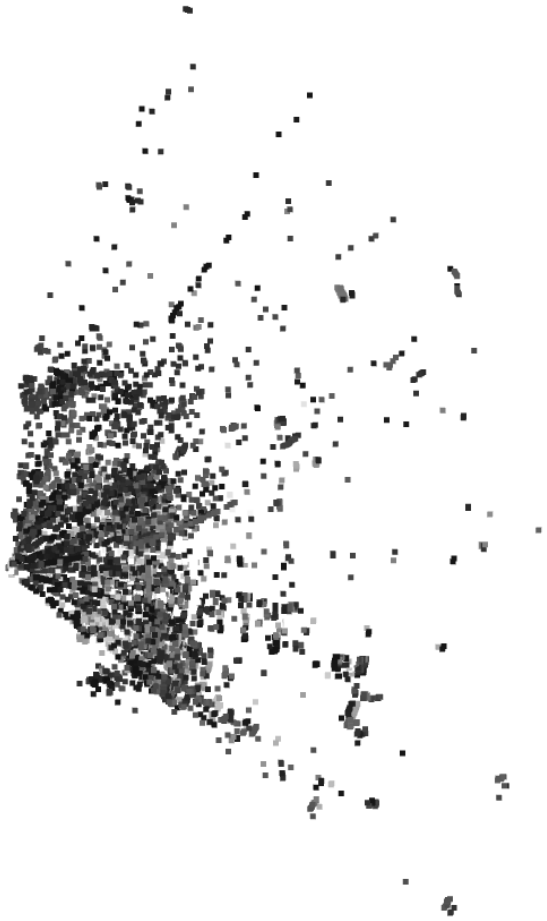


Figura 132. Nube de puntos 1.2 con valor umbral de NCC 0.9

- Resultados fotografía II:

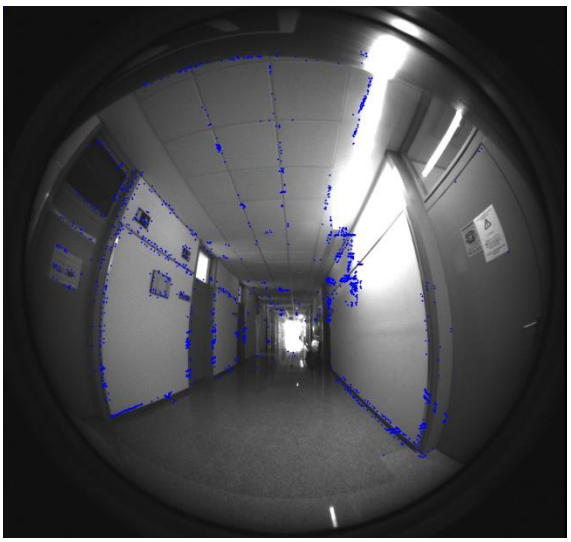


Figura 133. Matching fotométrico (II) con valor umbral de NCC de 0.9

Figura 134. Nube de puntos 2.1 con valor umbral de NCC 0.9

Figura 135. Nube de puntos 2.2 con valor umbral de NCC 0.9

- Resultados fotografía III:

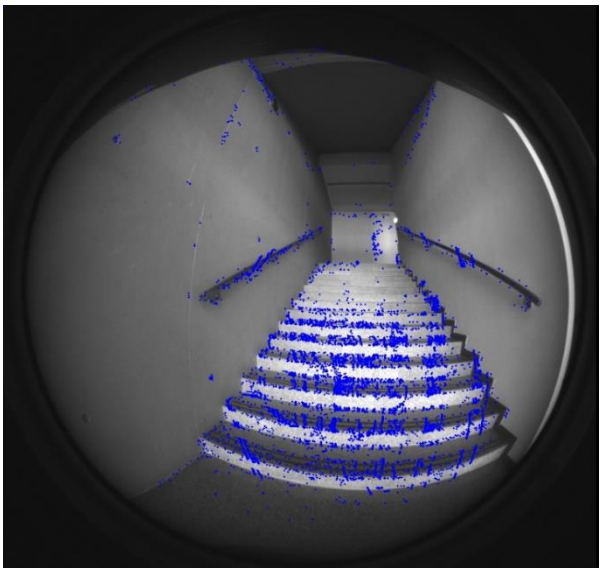


Figura 136. Matching fotométrico (III) con valor umbral de NCC de 0.9

Figura 137. Nube de puntos 3.1 con valor umbral de NCC 0.9

Figura 138. Nube de puntos 3.2 con valor umbral de NCC 0.9

- Resultados fotografía IV:

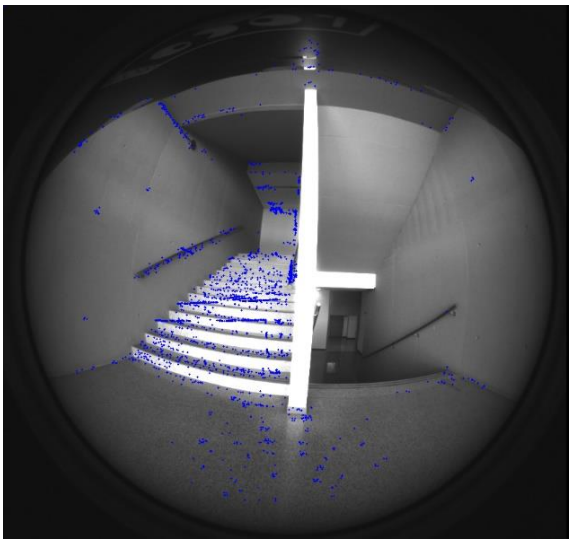


Figura 139. Matching fotométrico (IV) con valor umbral de NCC de 0.9

Figura 140. Nube de puntos 4.1 con valor umbral de NCC 0.9

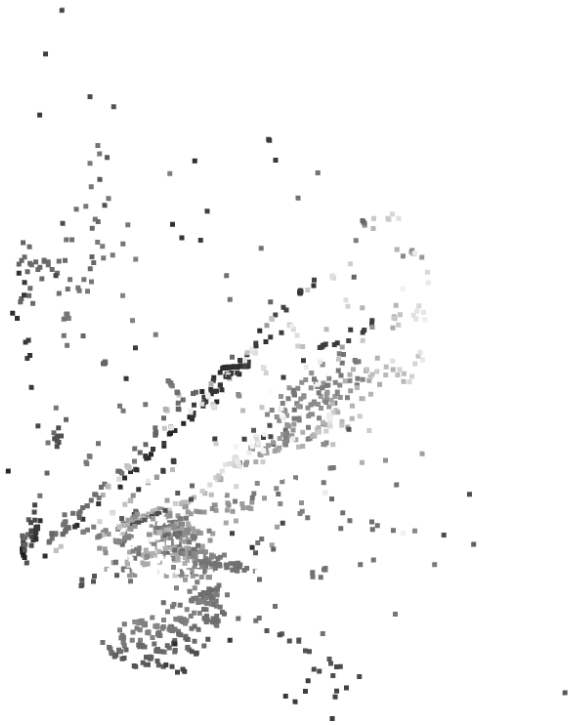


Figura 141. Nube de puntos 4.2 con valor umbral de NCC 0.9

- Resultados fotografía V:

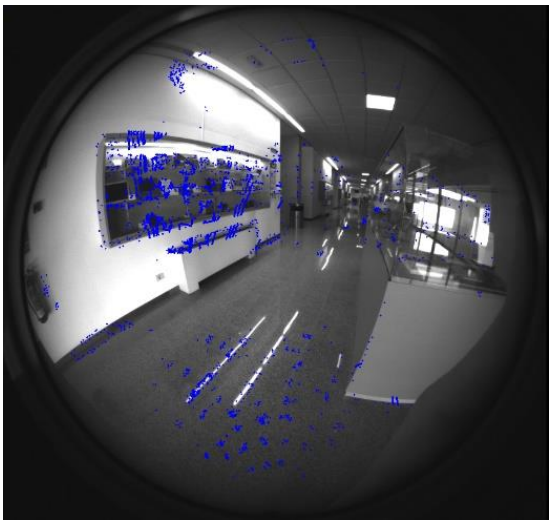


Figura 142. Matching fotométrico (IV) con valor umbral de NCC de 0.9

Figura 143. Nube de puntos 5.1 con valor umbral de NCC 0.9

Figura 144. Nube de puntos 5.1 con valor umbral de NCC 0.9

Anexo F. Recopilación de resultados mediante el aumento del tamaño de Kernel

Para un kernel de 9x9:

- Resultados fotografía I:

Figura 146. Nube de puntos 1.1 con un kernel de 9x9

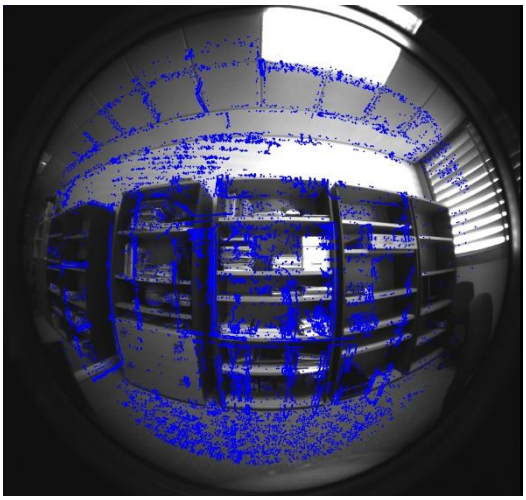


Figura 145. Matching fotométirco (I) con un kernel de 9x9

Figura 147. Nube de puntos 1.1 con un kernel de 9x9

- Resultados fotografía II:

Figura 149. Nube de puntos 2.1 con un kernel de 9x9

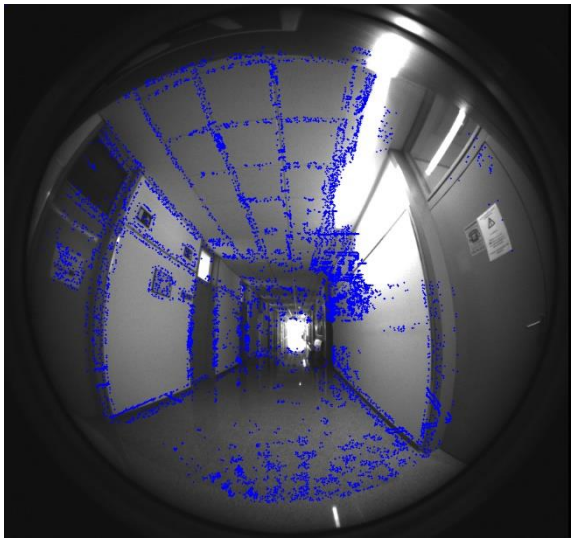
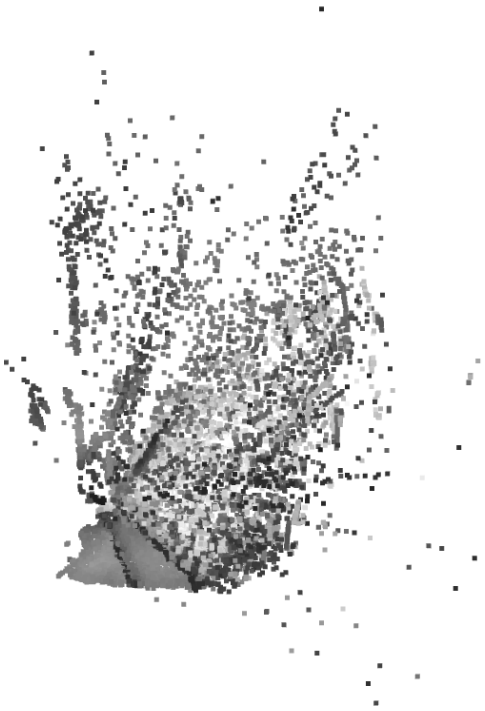


Figura 148. Matching fotométirco (II) con un kernel de 9x9

Figura 150. Nube de puntos 2.2 con un kernel de 9x9

- Resultados fotografía III:



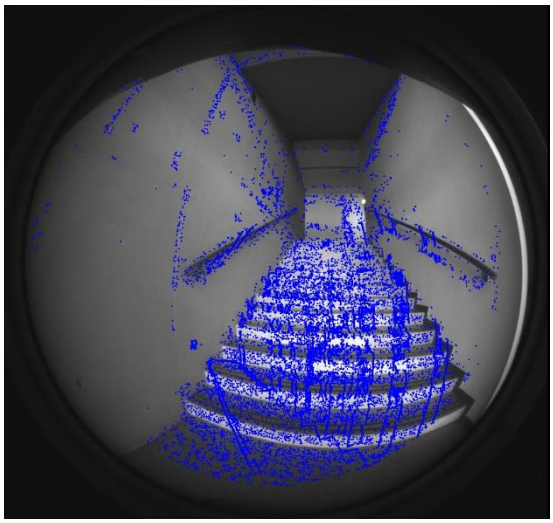


Figura 151. Matching fotométirco (III) con un kernel de 9x9

Figura 152. Nube de puntos 3.1 con un kernel de 9x9

Figura 153. Nube de puntos 3.2 con un kernel de 9x9

- Resultados fotografía IV:

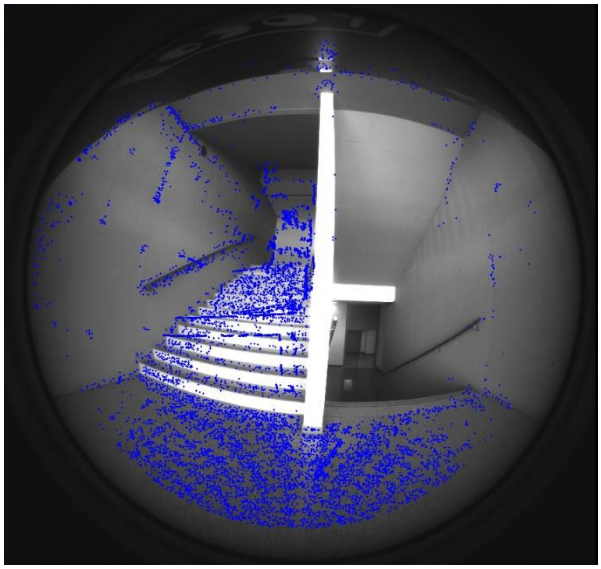


Figura 154. Matching fotométirco (IV) con un kernel de 9x9

Figura 155. Nube de puntos 4.1 con un kernel de 9x9

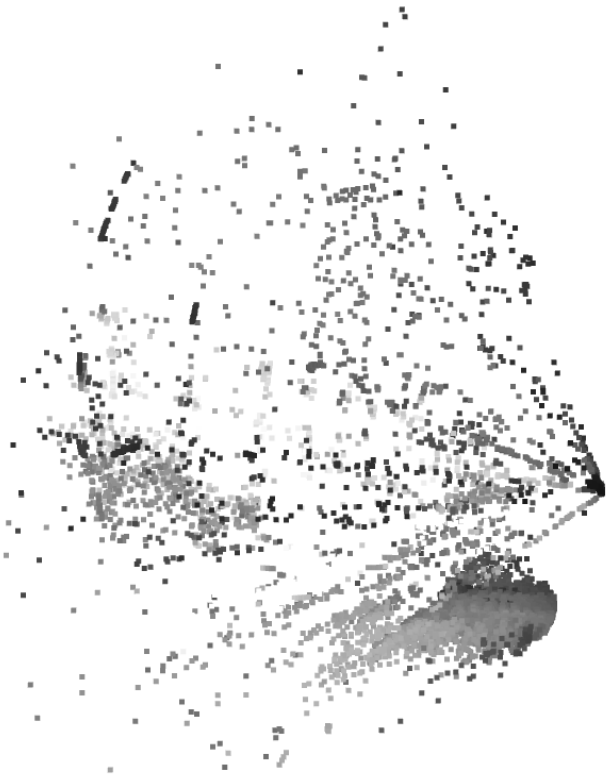
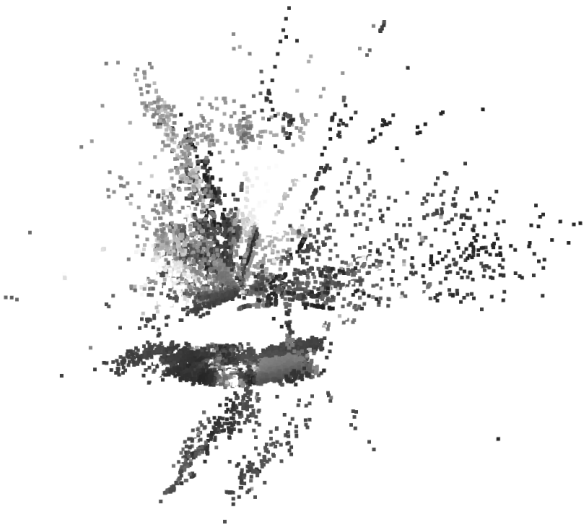


Figura 156. Nube de puntos 4.2 con un kernel de 9x9

- Resultados fotografía V:



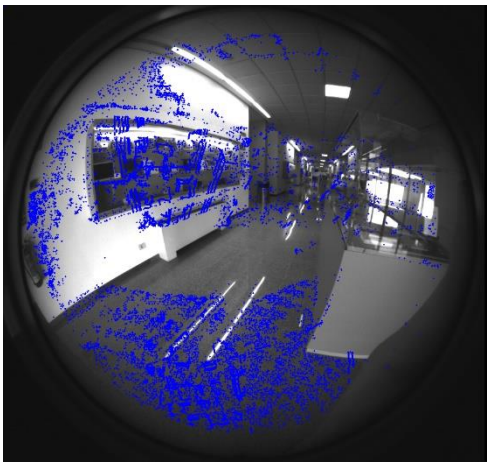


Figura 157. Figura 158. Matching fotométirco (V) con un kernel de 9x9

Figura 159. Nube de puntos 5.1 con un kernel de 9x9

Figura 160. Nube de puntos 5.2 con un kernel de 9x9