UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Reconstrucción de un objeto en 3D a partir de fotografías

Trabajo experimental

Wilman Daniel Vinueza Jiménez

Ingeniería Electrónica

Trabajo de titulación presentado como requisito
para la obtención del título de
Ingeniero Electrónico

Quito, 21 de diciembre de 2016

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO POLITECNICO

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

Reconstrucción de un objeto en 3D a partir de fotografías

Wilman Daniel Vinueza Jiménez

Calificación:

Nombre del profesor, Título académico — Luis Miguel Prócel , Ph.D.

Firma del profesor _____

Quito, 21 de diciembre de 2016

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Wilman Daniel Vinueza Jiménez

Código: 00108277

Cédula de Identidad: 1722315064

Lugar y fecha: Quito, 21 de diciembre de 2016

RESUMEN

Este texto describe el procedimiento y los algoritmos clave para reconstruir un objeto en 3 dimensiones a partir de fotos. El proceso en sí tiene 3 pasos principales. En primer lugar, identificar los rasgos únicos que caracterizan el objeto en la foto (estos rasgos se conocen como “features”). A continuación, se debe encontrar estos “features” en diferentes imágenes, un proceso conocido como "matching". Una vez que hemos hecho coincidir los “features” entre dos o más imágenes, el paso final es utilizar la matriz de proyección de cada cámara y el proceso de triangulación para recuperar puntos en 3D. En este documento se proporciona un trasfondo de la teoría utilizada en cada paso. El conjunto de los diferentes algoritmos utilizados proporciona con éxito un conjunto de puntos en 3D que describen al objeto.

Palabras clave: stereopsis, features, ventana, descriptor, triangulación, matriz de proyección, función gaussiana.

ABSTRACT

This text describes the procedure and the key algorithms to reconstruct an object in 3 dimensions from photos. The process itself has 3 main steps. First, identify unique features that characterize the object in the photo (i.e. corners and blobs). Next, find these features in different pictures, a process known as “matching”. Once we have had matched the features between two or more images, the final step is to use the projection matrix of each camera and the triangulation process to recover points in 3D. This paper provides a background of the theory used in each step. The set of the different algorithms used, successfully provides a set of 3D points that describe the object.

Key words: stereopsis, features, window, descriptor, triangulation, projection matrix, gaussian function.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En el campo de la reconstrucción 3D, existen dos grandes métodos: pasivos y activos. En los métodos activos, se utiliza la luz (láser) en el objeto para obtener un modelo 3D. En el pasivo se utilizan imágenes o video. Los escáneres láser son muy precisos pero caros y voluminosos. Gracias a la investigación intensiva, los algoritmos que se implementan en fotografías igualan la fidelidad del escáner láser (Furukawa, 2008) y ya que lo único que se necesita es una cámara digital común, este método es muy económico.

Una fotografía es una representación en dos dimensiones de un objeto o de un momento en particular. Cada pixel guarda la intensidad del rayo de luz que captura el sensor de la cámara. En la actualidad, las fotografías son muy comunes gracias al desarrollo de la tecnología y de redes sociales. Si vemos un objeto realmente genial, como una estatua, una construcción, un coche o una figura de acción, naturalmente, sacamos nuestro teléfono y tomamos algunas fotos del objeto que nos interesa. El objeto se ve muy bien, pero está "atrapado" en dos dimensiones. En este trabajo se presentan un grupo de algoritmos que permiten dar una dimensión más al objeto de interés. Es como darle más magia a la fotografía.

Una de las primeras ideas que cruzan por la mente cuando se piensa en reconstruir un objeto en 3D, es en imprimirlo en 3D. La mejora continua de métodos rápidos y baratos para el escaneo 3 dimensiones, permitirá a la gente hacer cosas más creativas e inesperadas con esta tecnología. Compartir y hacer objetos estará disponible para todos. Hoy es posible imprimir en 3D una versión de plástico del objeto fotografiado; en un futuro próximo será posible imprimir una réplica completamente funcional. Los algoritmos desarrollados para la

reconstrucción se utilizan también para mapas 3D en smartphones y también para reconstruir el interior de edificios y hacer recorridos virtuales. Es una valiosa herramienta para arquitectos y diseñadores; ya que les permite ver e imprimir en 3D su diseño y mostrarlo.

Este trabajo está dividido en tres secciones. La primera es encontrar features en la imagen, es decir rasgos que sean invariantes en cuanto a rotación, desplazamiento o intensidad lumínica. Se usa el detector de esquinas de Harris y el detector de blobs para esta sección. La segunda parte tiene que ver con encontrar estos features en otras imágenes. Finalmente, la última sección es la obtención de puntos en 3D utilizando el matching entre dos imágenes y las matrices de proyección de las cámaras. Cada proceso es independiente del otro. Si se mejora un proceso se mejora el resultado final. El proceso de reconstrucción en 3D a partir de fotografías es un proceso modular.

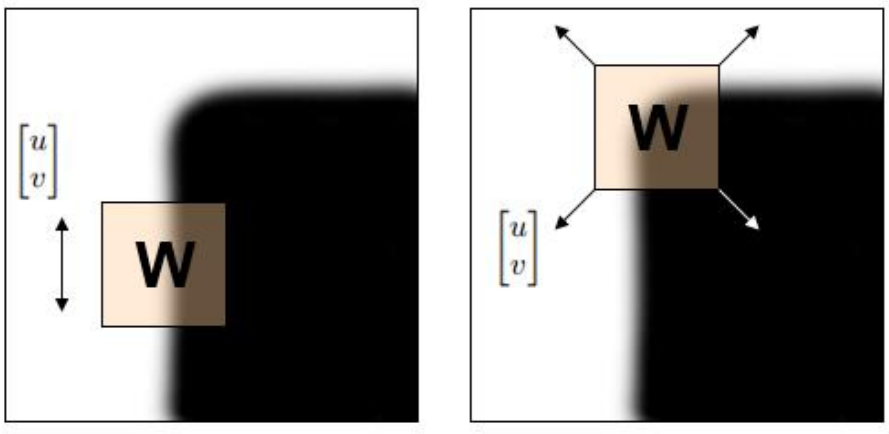
Las imágenes se obtienen de la página web, visión.middlebury.com. Se usó específicamente el dataset de dinoSparseRing y dinoRing que contienen 16 y 48 fotografías respectivamente de un pequeño dinosaurio (stegosaurus) de cerámica y las matrices de proyección para cada fotografía. Los algoritmos implementados en este trabajo se encuentran dentro de un nivel básico dentro de lo que es la reconstrucción en 3D. Implementaciones mucho más sofisticadas de todos los algoritmos discutidos en este trabajo permiten la reconstrucción de un objeto a partir de fotografías descargadas del internet, sin matrices de proyección y sin control sobre las imágenes tomadas. El trabajo *"Building Rome in a Day"* de Agarwal et al. (Agarwal, 2005) es un ejemplo de una reconstrucción sofisticada.

DESARROLLO DEL TEMA

Detección de features

La detección de features es un componente esencial en muchas aplicaciones de visión computarizada. Es un paso previo para recuperar las poses de las cámaras que tomaron las fotografías, se usa para alinear dos fotos y formar una panorámica, para el reconocimiento de objetos, la construcción de un objeto en 3D, etc. (Szeliski, 2010). Un feature debe cumplir ciertas características básicas. Debe ser posible encontrarlo en otra imagen a pesar de transformaciones geométricas o fotométricas. Ocupan una pequeña área de la imagen, lo que los hacen robustos ante la oclusión. Además, cada feature tiene una descripción diferente por sí mismo (Szeliski & Seitz, 2008). En las siguientes secciones describiremos dos algoritmos populares: detección de esquinas y blobs.

Detector de esquinas de Harris.



a. b.

Figura 1: Ventana *W* y vector desplazamiento *[u, v]* en un borde (a) y en una esquina (b). (Lazebnik, 2010).

Si nos colocamos en una esquina del objeto fotografiado y movemos una ventana *W* en cualquier dirección, el cambio en la intensidad de la ventana es grande, como se muestra en la Figura 1.b. Por otro lado, si la ventana se desplaza a lo largo del borde (Figura 1.a) no existe cambio en la intensidad. Podemos calcular cuánto cambia una ventana *W* al moverse en una determinada dirección si calculamos la suma de diferencias al cuadrado de las intensidades entre cada ventana (Szeliski & Seitz, 2008):

$$E(u,v) = \sum_{(x,y) \in W} [I(x+u, y+u) - I(x,y)]^2 \qquad (\mathbf{Ec.\ 1})$$

Donde $I(x,y)$ es el valor de la imagen en la coordenada (x,y). Podemos rescribir la expresión $I(x+u, y+u)$ (tomando en cuenta que u y v son desplazamientos muy pequeños) usando series de Taylor.

$$I(x+u, y+u) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \qquad (\mathbf{Ec.\ 2})$$

Reemplazando ecuación 2 en la ecuación 1 tenemos:

$$E(u,v) = \sum_{(x,y) \in W} \left[\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v\right]^2 = \sum_{(x,y) \in W} \left[\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}\right]^2 \qquad (\mathbf{Ec.\ 3})$$

Donde $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$. Finalmente, podemos escribir el cambio en una ventana W de la siguiente forma:

$$E(u,v) = \sum_{(x,y)\in W} [u \quad v] \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \qquad (\mathbf{Ec.\,4})$$

En donde:

$$H = \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \qquad (\mathbf{Ec.\,5})$$

La matriz *H* se conoce como matriz hessiana. Para encontrar las direcciones u, v que maximizan E es necesario un análisis de los vectores y valores propios de esta matriz. Al resolver la ecuación $\det(H - \lambda I) = 0$, obtenemos dos valores propios de la matriz hessiana dados por:

$$\lambda_{\pm} = \frac{1}{2}\left[h_{11} + h_{22} \pm \sqrt{4h_{12}h_{21} + (h_{11} - h_{22})^2}\right] \qquad (\mathbf{Ec.\,6})$$

Con los valores propios podemos calcular los vectores propios asociados x_+ y x_-. El vector propio x_+ es la dirección en donde se observa el mayor cambio en la ventana Wal desplazarse en esa dirección, ese cambio está dado por λ_+. De manera análoga la dirección en la que la ventana tiene el mínimo cambio es x_- y la cantidad de ese cambio es su valor propio λ_- (Szeliski & Seitz, 2008).

Para obtener un buen feature, la función $E(u,v)$ debe ser grande para pequeños desplazamientos en todas las direcciones. En otras palabras, el mínimo de la función $E(u,v)$ debe ser grande para todos los vectores $[u\ v]$; éste mínimo está dado por el valor propio más pequeño λ_- (Szeliski & Seitz, 2008). Si λ_- es grande significa que incluso si se mueve la

ventana en la dirección de menos cambio, se obtiene un cambio significativo. El valor de $\lambda\_$ podría usarse como parámetro para encontrar esquinas en la imagen. Sin embargo, como se puede apreciar en la ecuación 6, el cálculo de este valor involucra una raíz cuadrada, lo que es costoso computacionalmente. Por esta razón, se acostumbra a usar el "Operador de Harris" que está dado por:

$$f = \frac{\det(H)}{traza\ (H)} = \frac{h_{11} \times h_{22} - h_{12} \times h_{21}}{h_{11} + h_{22}} \qquad (\mathbf{Ec.\ 7})$$

Existen muchos otros operadores que evitan el cálculo de la raíz cuadrada, éste es el más popular. En este operador, el valor de threshold que se usa para detectar esquinas fuertes es de alrededor de 2.7. Para la reconstrucción en 3D necesitamos detectar el mayor número de features en la imagen y para ciertas imágenes, las esquinas fuertes no son suficientes. En estos casos, es posible reducir el threshold a 1.3 para detectar esquinas normales o incluso a 0.02 para detectar bordes (Furukawa, 2016). En la Figura 2 podemos ver un ejemplo de estos casos. Se puede apreciar como la variación del threshold afecta el tipo y número de features detectados; en las primeras dos imágenes los features son esquinas; en las siguiente dos imágenes, bordes empiezan a ser detectados.

Determinar el threshold es una parte importante de cualquier algoritmo; si es muy bajo se aceptan falsos positivos y si es muy alto se rechazan buenos puntos. Para aumentar la cantidad de features que se pueden encontrar en la imagen, se pueden recurrir a otros algoritmos, como el detector de blobs, por ejemplo, en el que hablaremos en la siguiente sección. El detector de Harris implementado es invariante a rotación y a intensidad. Es decir

que encontraremos los mismos features en otra imagen a pesar de que la imagen haya sido rotada y sus pixeles hayan cambiado de intensidad (Lazebnik, 2016).

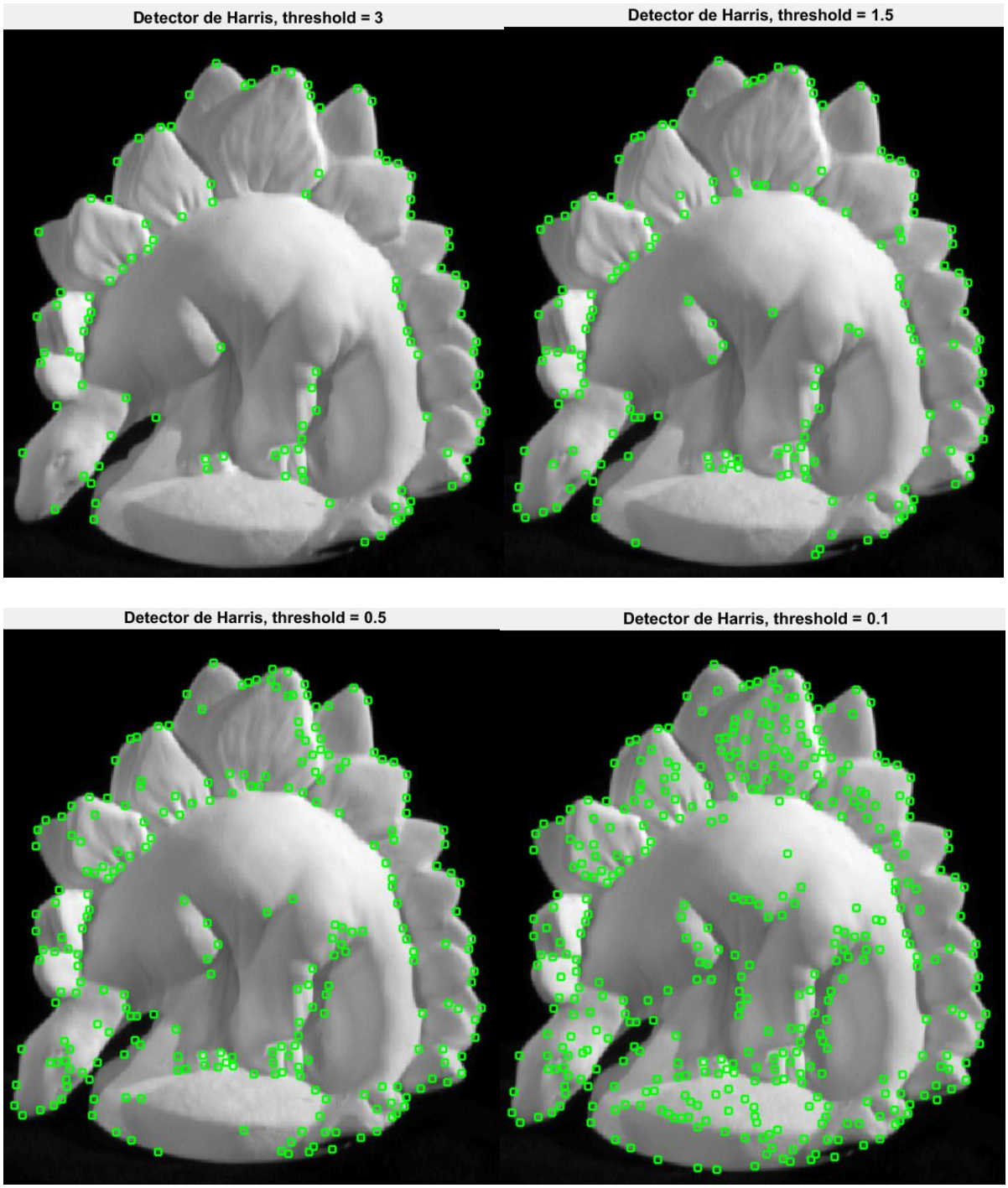


Figura 2. Variación del threshold para detectar diferentes features (esquinas y bordes).

El algoritmo es el siguiente:

1. Calcular el gradiente en cada punto en la imagen.
2. Crear la matriz *H* a partir del gradiente.
3. Convolucionar cada parámetro de la matriz *H* con una gaussiana.
4. Calcular el operador de Harris *(f)*.
5. Encontrar puntos en donde *f* es mayor que el threshold.
6. Escoger aquellos puntos donde *f* es máximo local como features. (Furukawa, 2016).

El gradiente puede ser calculado usando una variedad de técnicas. El detector "Harris" clásico usa un filtro de la forma [-2 -1 0 1 2] (Szeliski, 2010). En este trabajo se calculan máscaras derivativas gaussianas que luego se convolucionan con la imagen. Ya que la máscara gaussiana y su derivada son simétricas se puede usar también el producto punto entre la máscara y la imagen para calcular el gradiente, así como para realizar la convolución entre la imagen y una gaussiana de dos dimensiones (Jacobs, 2005). En la Figura 3 podemos observar los pasos 4 y 5 del algoritmo, la Figura 4 muestra el resultado final después de encontrar máximos locales para un threshold de 2.

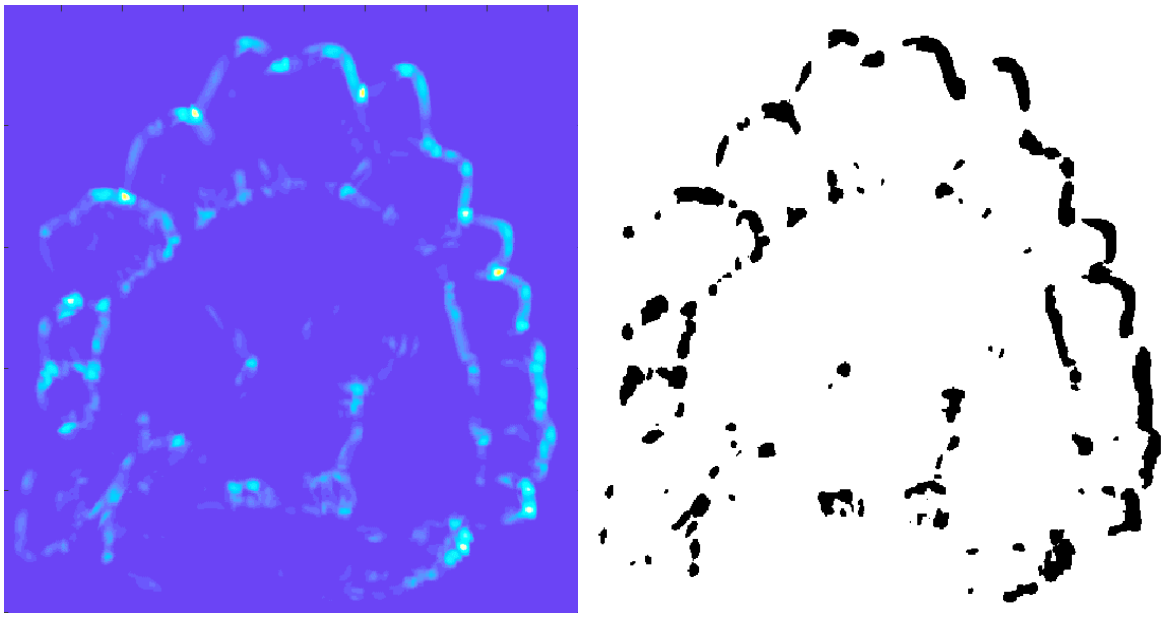


Figura 3. A la izquierda, valores del operador *f*. En la derecha, pixeles en donde *f* es mayor que el threshold.

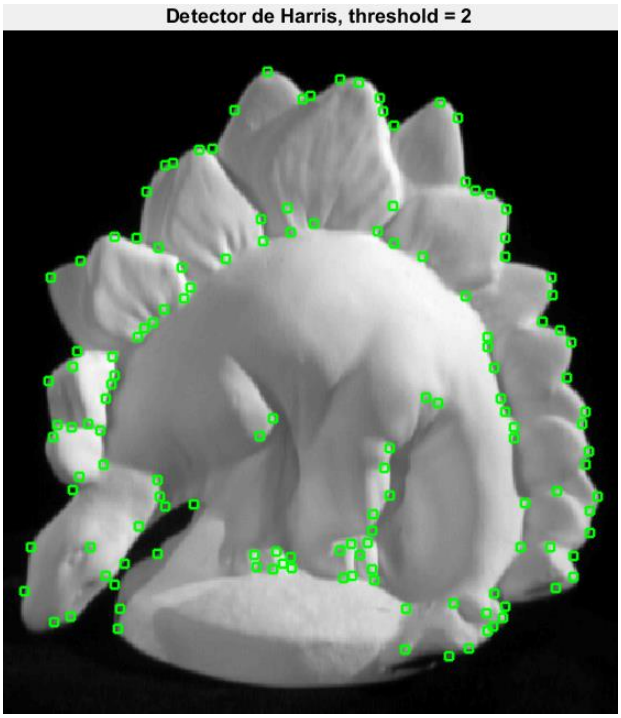


Figura 4. Resultado final después de encontrar máximos locales en *f*.

Detector de blobs.

Los blobs son rasgos en la imagen que tienen forma circular o similar. Para detectarlos se convoluciona la imagen con un filtro laplaciano, también conocido como LoG (Laplacian of Gaussian). La respuesta a este filtro alcanzará un máximo en el centro del blob, únicamente cuando la escala del laplaciano concuerde con la escala del blob (Lazebnik, 2010). En la Figura 5 podemos ver un ejemplo en una dimensión. Para encontrar blobs a diferentes escalas se realizan varias convoluciones entre la imagen y el laplaciano a diferentes σ.

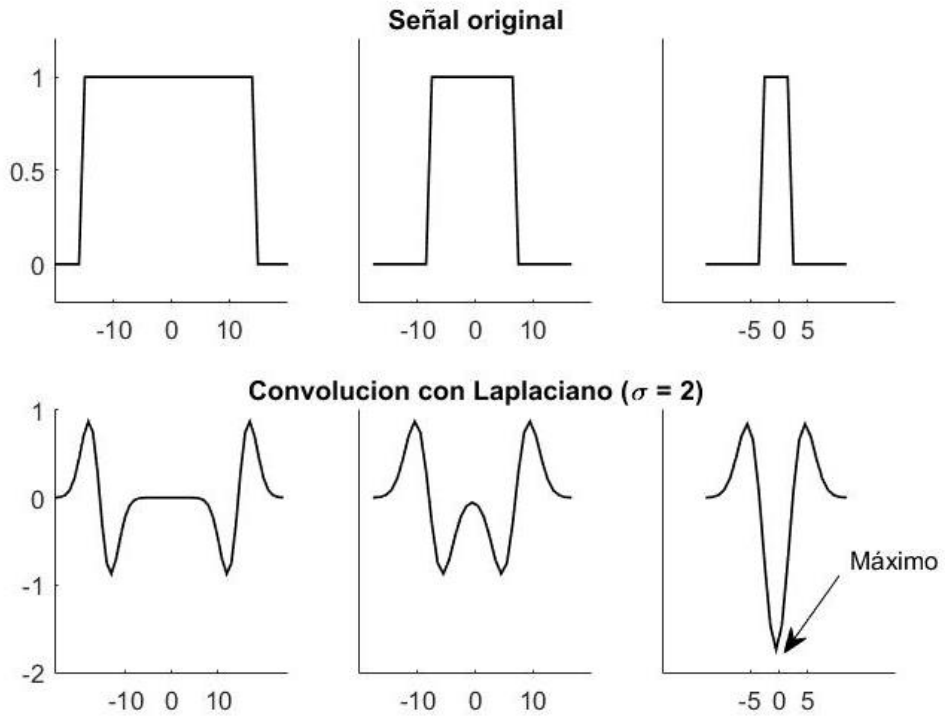


Figura 5. Ejemplo de detección de blobs en una dimensión. Se alcanza un máximo únicamente cuando la escala del laplaciano (σ) concuerda con la escala de la señal (blob).

Para un círculo en la imagen de radio *r* el laplaciano alcanza su máximo cuando $\sigma = r/\sqrt{2}$. Un factor importante a tomar en cuenta es que la respuesta entre el laplaciano y la señal decae conforme aumenta sigma. Por esta razón, el filtro laplaciano se debe multiplicar por σ^2. Sin embargo, en la práctica es mejor implementar lo que se conoce como diferencia de gaussianas (DoG); esta resta entre gaussianas se aproxima al laplaciano como se muestra en la Figura 6 y ya que no usa derivadas su implementación es más eficiente (Lazebnik, 2010).

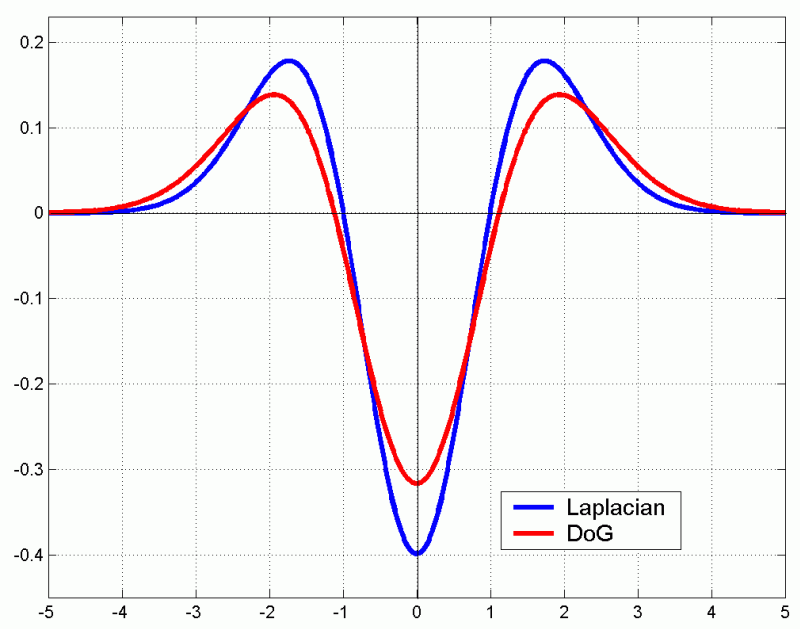


Figura 6. Ilustración de como una diferencia entre gaussianas (DoG) se aproxima al laplaciano. (Lazebnik, 2010).

Los blobs tienen la importante característica de encontrar features invariantes a escala y también pueden ser transformados a features afines. Sin embargo, su desempeño no es el mejor si en la imagen no existen patrones circulares. Además, en esta parte del desarrollo del programa no nos interesa encontrar features invariantes a escala ya que las

fotografías usadas para la reconstrucción en 3D no presentan esta variación. El algoritmo descrito en esta sección tiene el fin de conseguir más puntos para la reconstrucción y no necesariamente encontrar features invariantes a escala. En las Figuras 7 y 8 se observa los resultados de la implementación de este algoritmo en dos imágenes. En la parte izquierda de las Figuras 7 y 8, se muestra la imagen original con los blobs detectados. En la parte derecha se muestra el resultado de la convolución entre la imagen y el laplaciano a diferentes σ. Nótese como cambia la respuesta a la convolución al cambiar σ. La imagen mariposa, tiene varias formas circulares de diferentes tamaños y por lo tanto se encuentran diferentes blobs cuando cambia σ. Por otro lado, la imagen dino1 no presenta formas circulares. Por esta razón, los features detectados son similares a los obtenidos con el detector de esquinas de Harris.

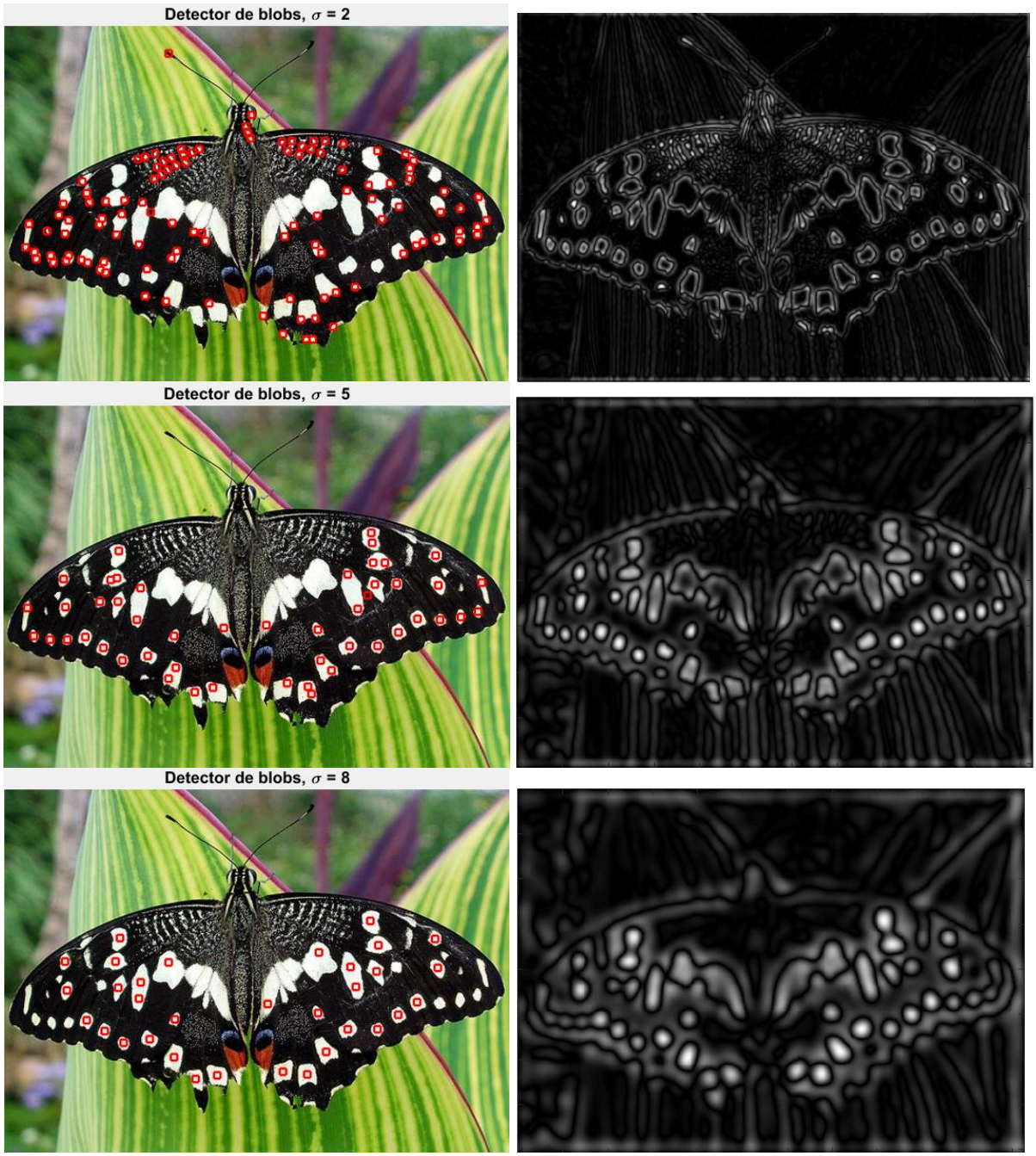


Figura 7. Resultados del algoritmo de detección de blobs para la imagen mariposa.

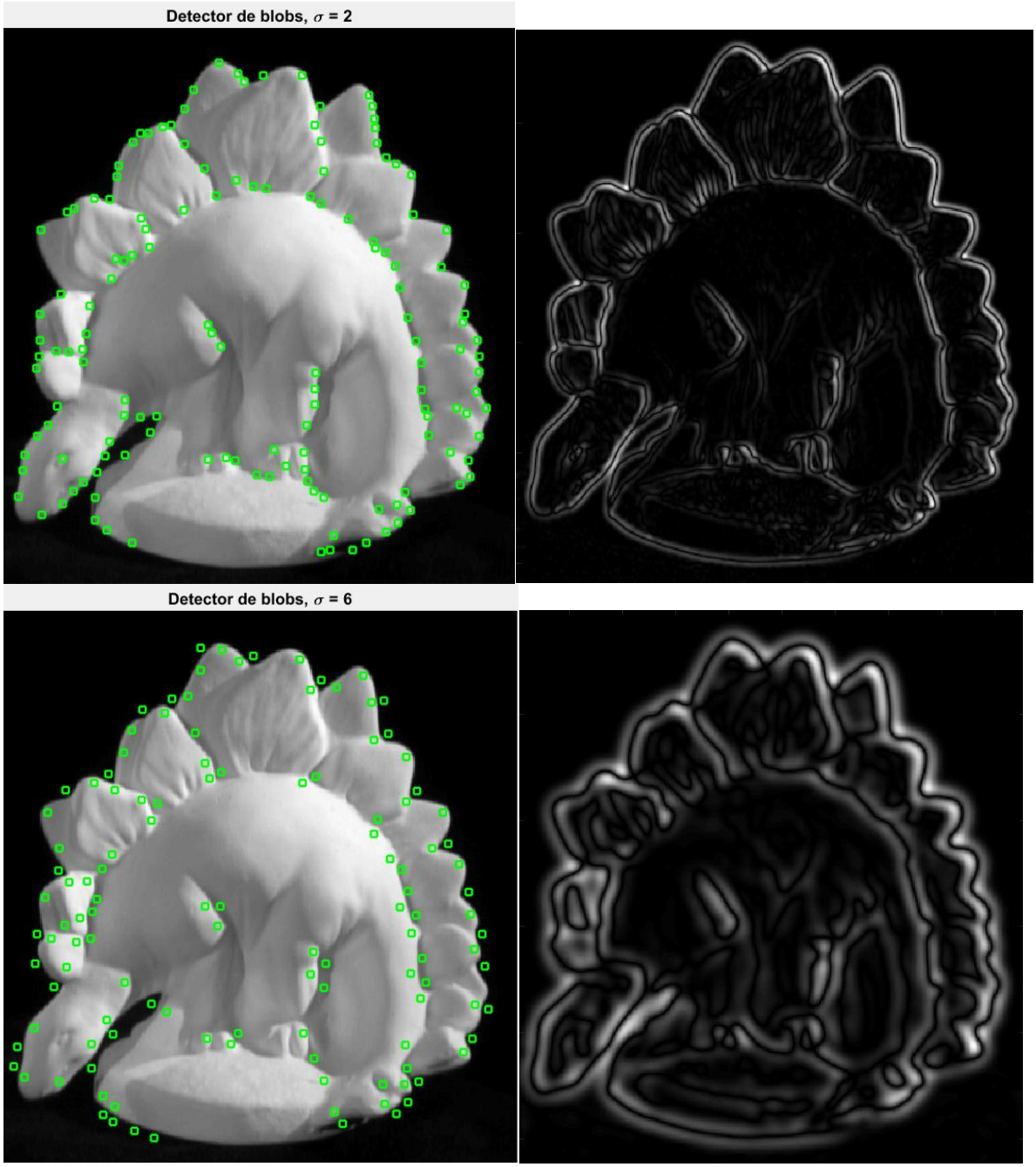


Figura 8. Resultados del algoritmo de detección de blobs para la imagen dino1.

En general, cuando la varianza del laplaciano se encuentra entre 1 y 2 los features encontrados con este algoritmo son similares a los encontrados con el operador de Harris. Sin embargo, la detección de blobs ayuda satisfactoriamente a elevar el número de features

encontrados en la imagen cuando cambiamos el valor de σ. En la Figura 9 podemos comparar los resultados entre los dos algoritmos.

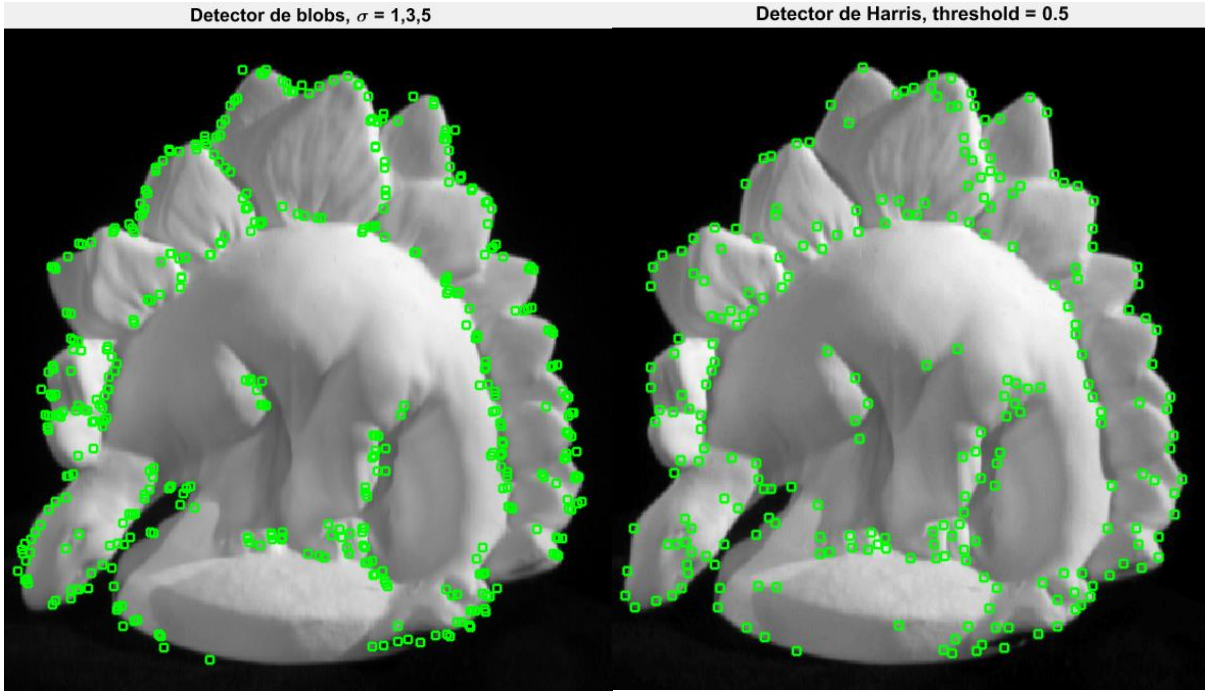


Figura 9. Resultado final del detector de esquinas de Harris (izquierda) y de blobs (derecha). Nótese como algunos features se repiten, pero el detector de blobs también aporta features nuevos.

Descriptor de un feature, matching y tracking

Descriptor.

Una vez que hemos encontrado un considerable número de features en las imágenes, procedemos al siguiente paso importante que consiste en encontrar correspondencias de features entre pares de imágenes. Comúnmente, un paso previo para hacerlo es crear un descriptor a partir del feature encontrado. La función de un descriptor es caracterizar de mejor manera a un feature para poder encontrarlo en otra imagen

(Szeliski, 2010). Existen muchos descriptores, uno de los más populares es SIFT (Scale Invariant Feature Transform) desarrollado por David Lowe. Este descriptor es invariante a transformaciones afines, puede manejar cambios de iluminación, es implementado en tiempo real y es usado en muchas aplicaciones (Szeliski & Seitz, 2008). En este trabajo se implementó un descriptor simple, el cual consiste en una ventana que tiene como centro el feature detectado.

Matching.

El proceso de matching consiste en comparar dos descriptores de acuerdo a una métrica determinada. Dado un descriptor $f1$ en una imagen I_1 tenemos que encontrar su mejor correspondencia en una segunda imagen I_2. Para esto definimos una función de distancia que compara los dos descriptores. Comparamos cada descriptor en I_1 con cada descriptor en I_2 y encontramos aquellos con la mínima distancia. Para determinar la diferencia entre dos descriptores $f1$ y $f2$ el enfoque más simple es la suma de diferencias al cuadrado (SSD por sus siglas en inglés) entre los dos descriptores. Esta métrica puede dar una distancia mínima para malos pares de descriptores (Szeliski & Seitz, 2008). En este trabajo se utilizó la distancia conocida como "ratio distance". Esta distancia se define según la ecuación 8:

$$r = \frac{SSD(f1, f2)}{SSD(f1, f2')} \qquad (\mathbf{Ec.\ 8})$$

En donde $f2$ es la mejor correspondencia de $f1$ en la imagen I_1 y $f2'$ es la segunda mejor correspondencia de $f1$ en I_1. Esta métrica nos da valores muy pequeños para buenos

matches o correspondencias. En este trabajo tomamos como ventaja que las fotografías a usarse no cambian mucho la una con respecto a la otra, esto ayuda que el proceso de matching sea satisfactorio.

Las Figuras 10 y 11 muestran las imágenes originales y los features encontrados. La Figura 12 muestra las correspondencias entre 10 descriptores; el proceso de matching es satisfactorio para este caso. En la Figura 13 se pueden notar ciertos errores de correspondencia al elevar el número de features correspondidos. En el caso de la Figura 14, existe un fuerte cambio de intensidad y tambien en la posición del objeto; el matching implementado falla drasticamente.

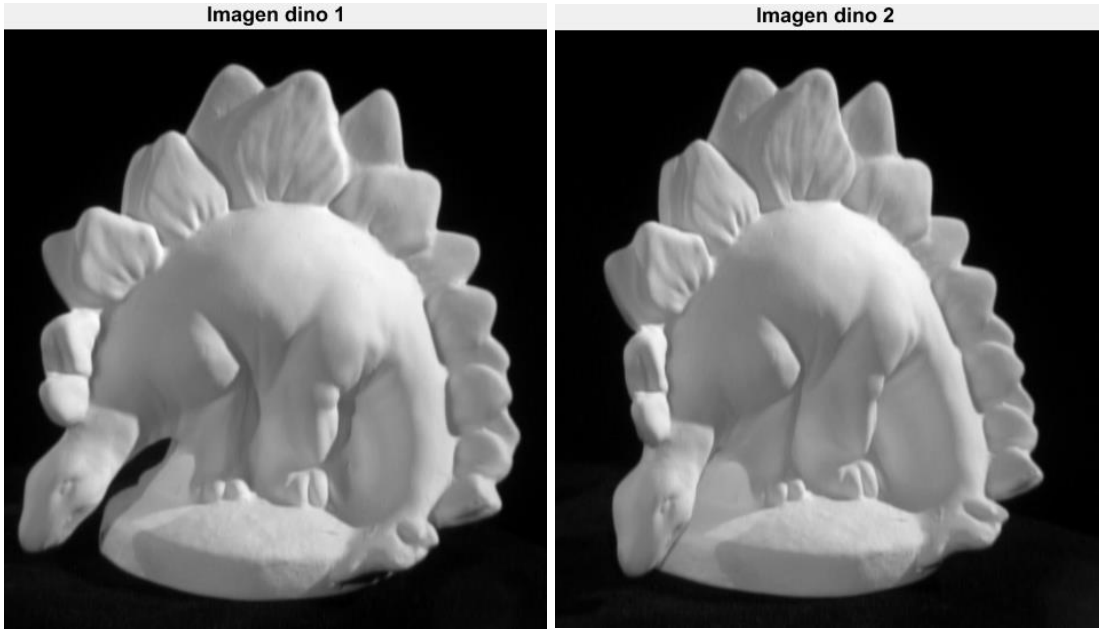


Figura 10. Imágenes originales utilizadas para el proceso de matching.

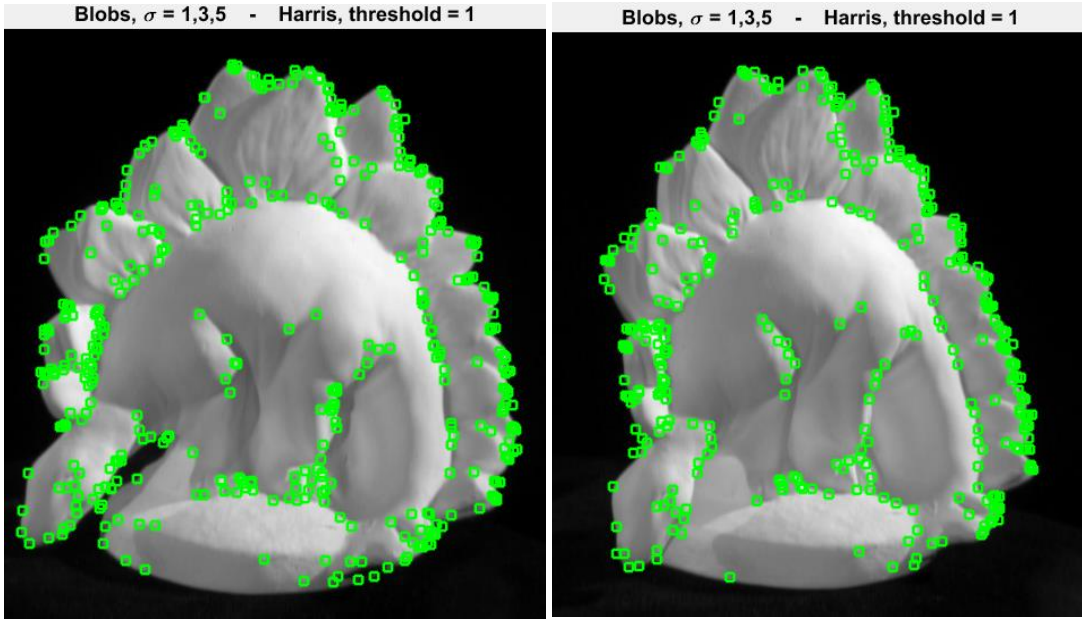


Figura 11. Esquinas y blobs detectados en el par de imágenes.

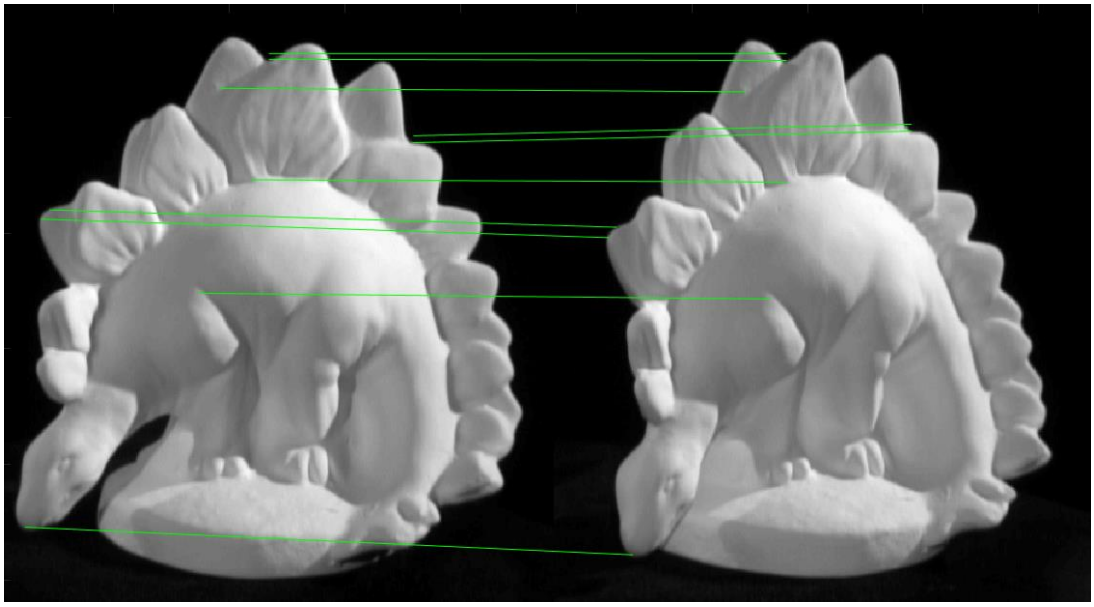


Figura 12. 10 correspondencias entre descriptores.

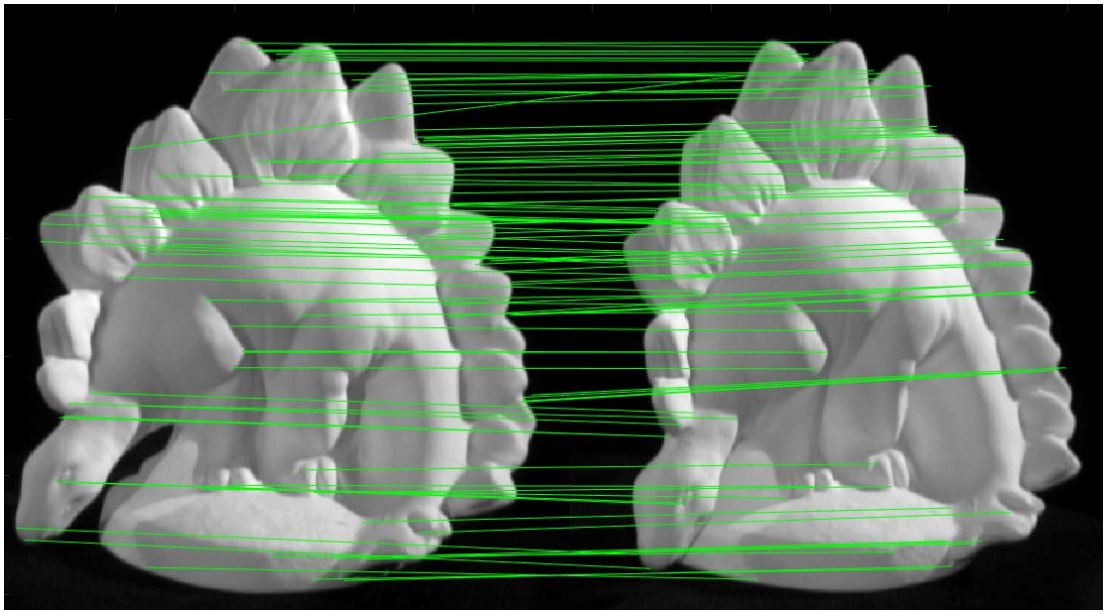


Figura 13. 100 correspondencias entre descriptores.

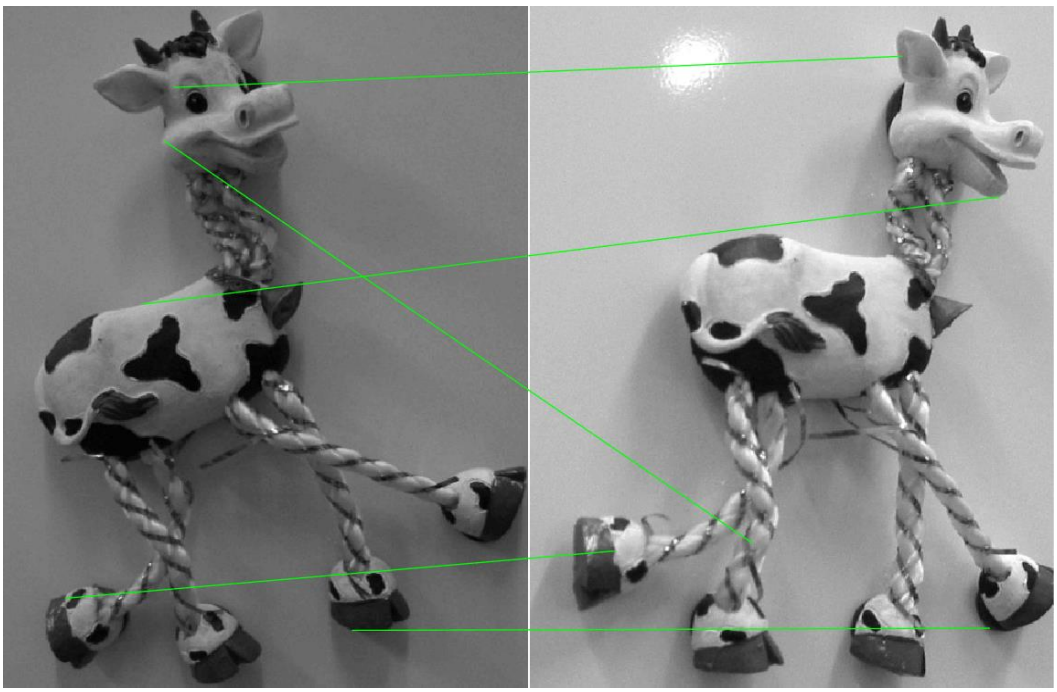


Figura 14. Falla en el proceso de matching cuando existe un cambio considerable entre las dos fotografias.

Tracking.

En el proceso de matching se busca potenciales correspondencias en toda la imagen. Por otro lado, en el proceso del tracking, se busca correspondencias en un vecindario cercano al feature detectado. Se lo utiliza comúnmente en procesamiento de video o si las fotografías no cambian mucha una respecto de la otra (Szeliski, 2010). En este trabajo, primero se utiliza la técnica de matching para encontrar correspondencias de features entre pares de imágenes y después se filtra malas correspondencias si la suma de diferencias al cuadrado, entre las coordenadas de los features supera un determinado threshold.

Las Figuras 15 y 17 muestran 100 correspondencias entre la imagen dino1 – dino2 y dino4 – dino5 respectivamente, se puede apreciar como existen correspondencias erróneas en ambas figuras. Las Figuras 16 y 18 muestran el resultado de las correspondencias después de eliminar malos matches usando la técnica de tracking.

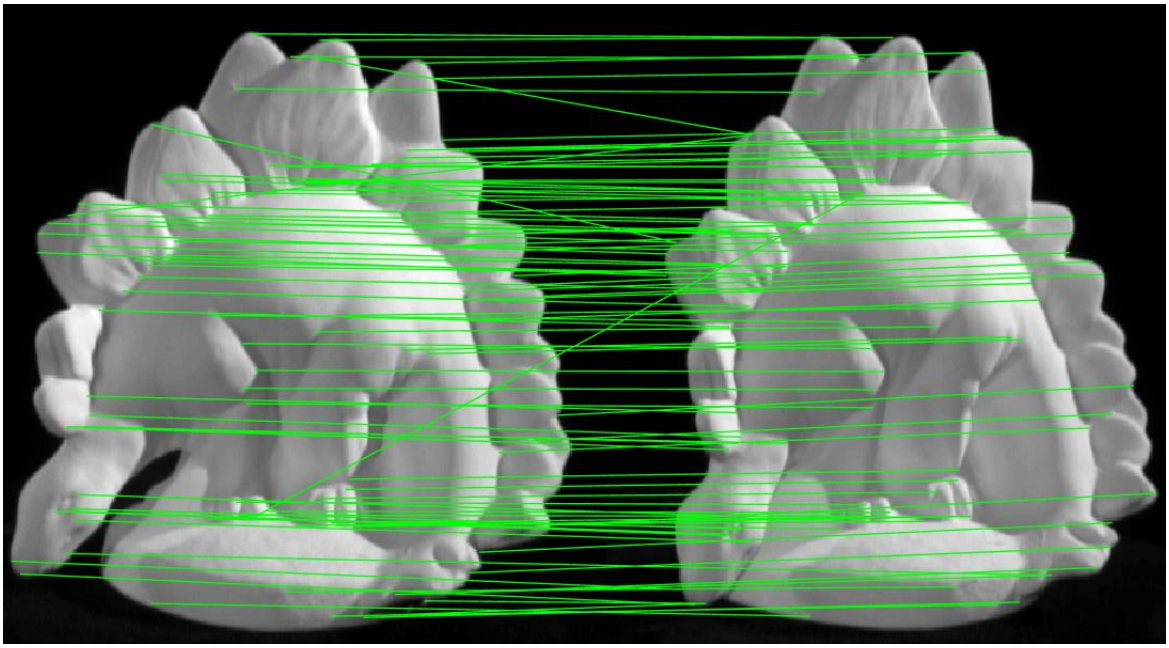


Figura 15. Correspondencias entre la imagen dino1 y dino2.

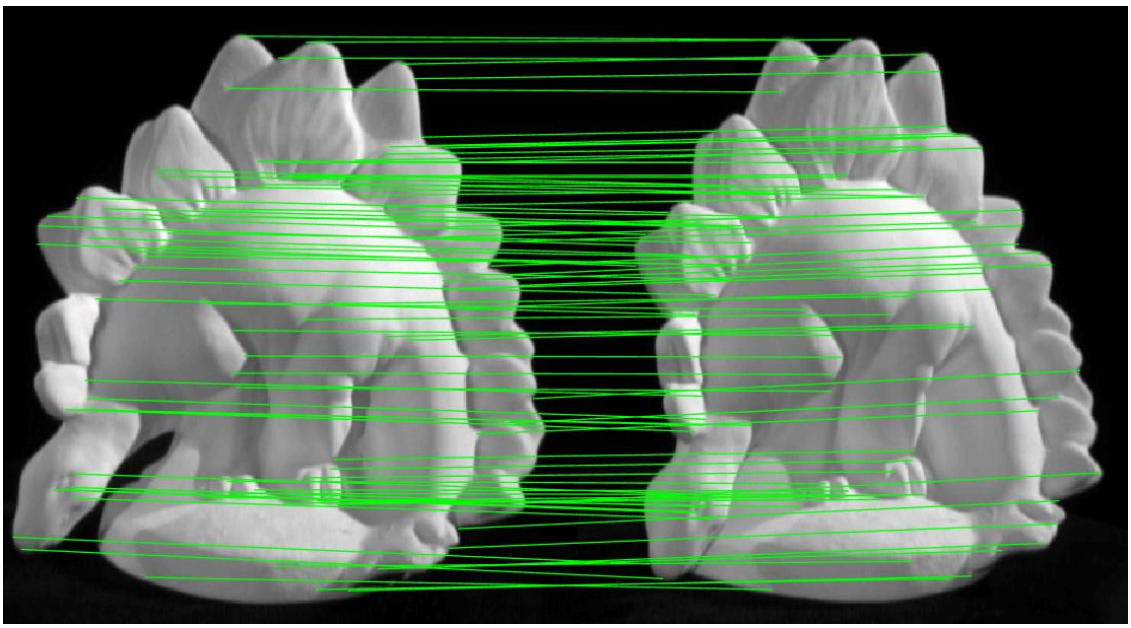


Figura 16. Correspondencias entre la imagen dino1 y dino2 despues de filtrar correspondencias erróneas.

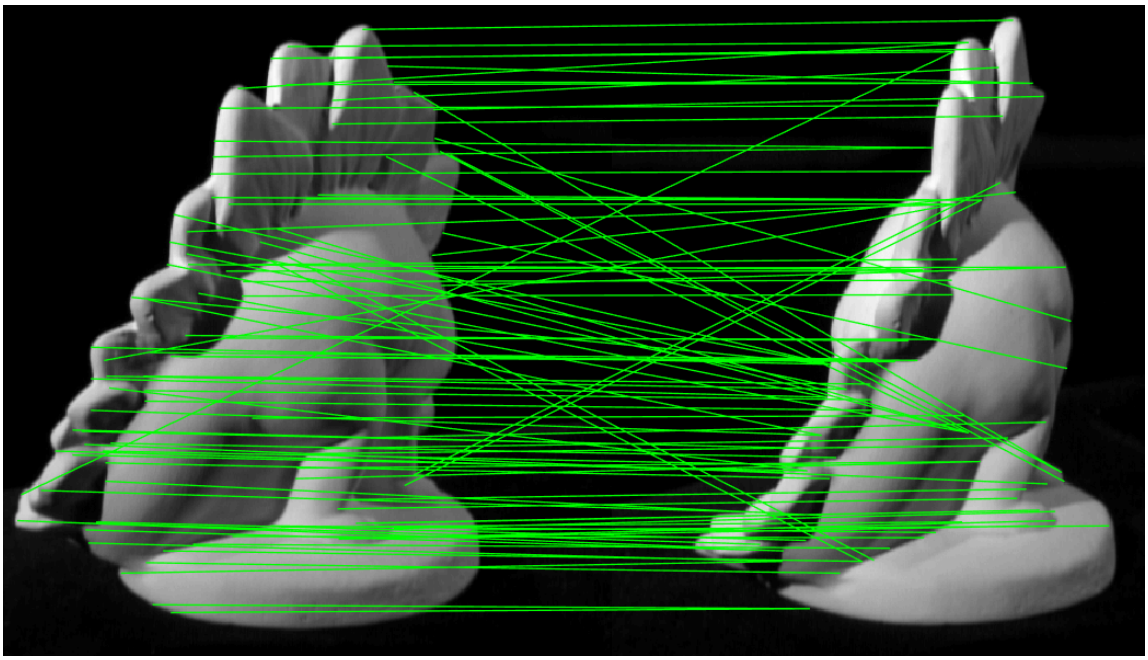


Figura 17. Correspondencias entre la imagen dino4 y dino5.

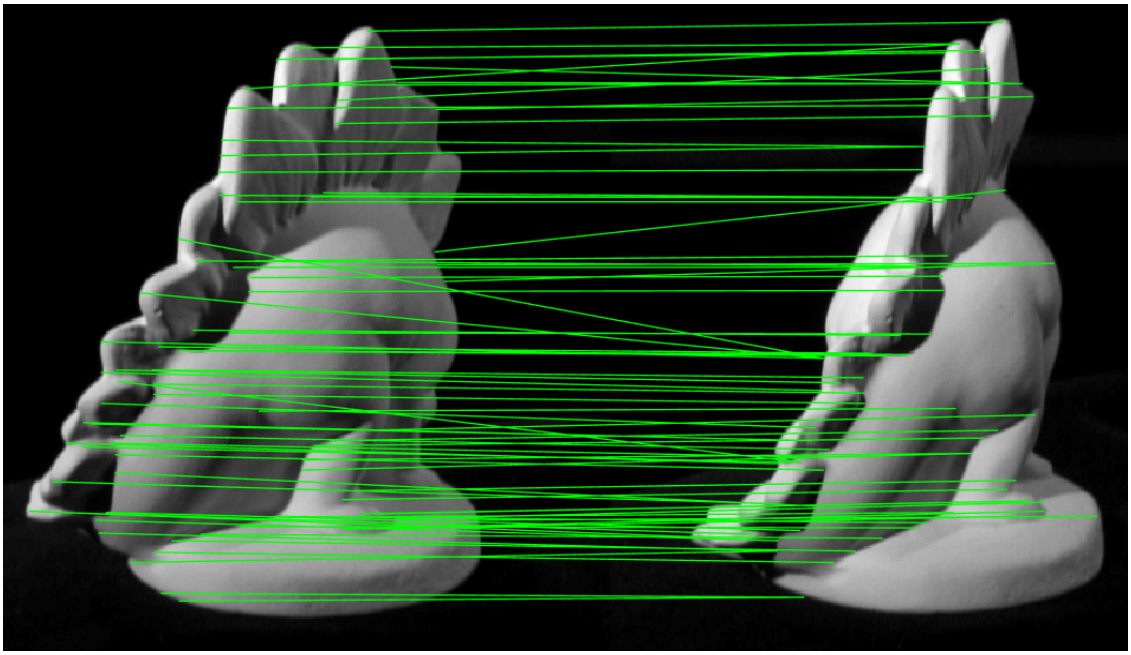


Figura 18. Correspondencias entre la imagen dino4 y dino5 despues de filtrar correspondencias erróneas.

Reconstrucción en 3D

Triangulación.

Para recuperar puntos en 3D a partir de correspondencias entre features usamos un proceso conocido como triangulación. Para realizar este proceso se necesitan las matrices de proyección de las cámaras que tomaron las fotografías. La página web de visión computarizada de Middlebury proporciona las matrices *K*, *R* y *t* para cada imagen utilizada. La matriz de proyección se obtiene como indica la ecuación 9.

$$P = K \times [R \mid t] \qquad (\mathbf{Ec.\ 9})$$

La matriz *K* guarda parámetros intrínsecos y es única para cada cámara. También se la conoce como matriz calibración. La ecuación 10 muestra los parámetros de la matriz *K*.

$$K = \begin{bmatrix} f & s & p_x \\ 0 & \alpha f & p_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (\mathbf{Ec.\,10})$$

En donde f es la distancia focal en pixeles. p_x y p_y son las coordenadas del punto principal (centro de la imagen) en pixeles. α es la relación de aspecto (si los pixeles no son cuadrados). Finalmente, s es un parámetro de corrección si los pixeles del sensor no son rectangulares (Furukawa, 2016).

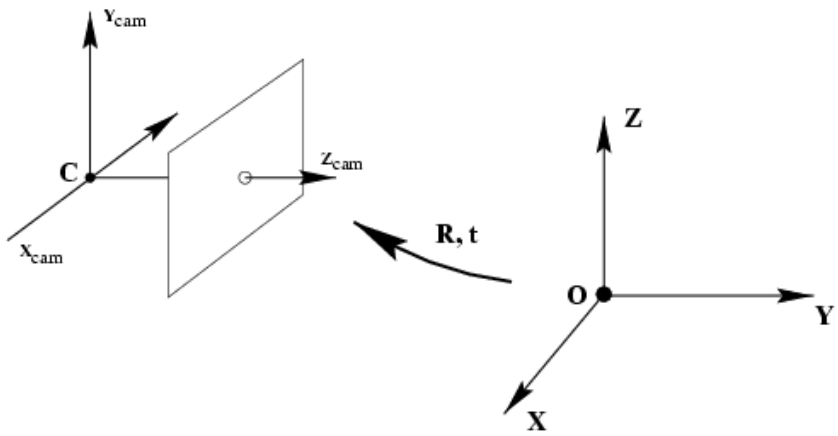


Figura 19. Representación de la función de la matriz extrínseca $[R \mid t]$.

(Lazebnik, 2010).

En la ecuación 9, *R* es una matriz de rotación de 3 × 3 y *t* es un vector traslación de 3 entradas, la combinación $[R \mid t]$ representa los parámetros extrínsecos de la cámara. Esta matriz describe como las coordenadas del mundo real son transformadas a las coordenadas de la cámara (Simek, 2012). La Figura 19 muestra una representación del papel de la matriz

extrínseca en una cámara. La matriz de proyección *P*, calculada según la ecuación 9 es una matriz de 3 × 4. Por ejemplo, una matriz de proyección tiene la forma de la ecuación 11:

$$P^i = \begin{bmatrix} P_{11}^i & P_{12}^i & P_{13}^i & P_{14}^i \\ P_{21}^i & P_{22}^i & P_{23}^i & P_{24}^i \\ P_{31}^i & P_{32}^i & P_{33}^i & P_{34}^i \end{bmatrix} = \begin{bmatrix} p_1^i \\ p_2^i \\ p_3^i \end{bmatrix} \qquad (\mathbf{Ec.\ 11})$$

En donde P^i es la matriz de proyección para la *i*-ésima imagen y p_1^i es la primera fila de esta matriz. En esta sección implementamos el proceso de triangulación mediante mínimos cuadrados. Si multiplicamos la matriz de proyección por un punto en 3D obtenemos en punto en 2D, es decir la coordenada de un pixel en la imagen. Sin embargo, para que esta multiplicación sea una transformación lineal se añade una coordenada más en ambas dimensiones, lo que normalmente se conoce como coordenadas homogéneas, como se muestra en la ecuación 12 (Collins, 2007).

$$w \begin{bmatrix} u^1 \\ v^1 \\ 1 \end{bmatrix} = \begin{bmatrix} P_{11}^1 & P_{12}^1 & P_{13}^1 & P_{14}^1 \\ P_{21}^1 & P_{22}^1 & P_{23}^1 & P_{24}^1 \\ P_{31}^1 & P_{32}^1 & P_{33}^1 & P_{34}^1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \qquad (\mathbf{Ec.\ 12})$$

En donde (u^1, v^1) es la coordenada de un feature detectado en la imagen I_1, w es el tercer componente del punto en 2D (en coordenadas homogéneas) y el vector $[X, Y, Z, W]^T$ representa el punto 3D en coordenadas homogéneas. Para simplificar la notación podemos escribir el punto en 3D Como muestra la ecuación 13.

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = [\bar{X}] \qquad (\mathbf{Ec.\,13})$$

De esta manera podemos escribir la ecuación 12 como se muestra en la ecuación 14 a continuación:

$$w \begin{bmatrix} u^1 \\ v^1 \\ 1 \end{bmatrix} = \begin{bmatrix} p_1^1 \\ p_2^1 \\ p_3^1 \end{bmatrix} [\bar{X}] \qquad (\mathbf{Ec.\,14})$$

De la ecuación 14 resulta fácil escribir:

$$wu^1 = [p_1^1][\bar{X}] \qquad (\mathbf{Ec.\,15})$$

$$wv^1 = [p_2^1][\bar{X}] \qquad (\mathbf{Ec.\,16})$$

$$w = [p_3^1][\bar{X}] \qquad (\mathbf{Ec.\,17})$$

Al reemplazar la ecuación 17 en las ecuaciones 15 y 16 obtenemos:

$$u^1[p_3^1][\bar{X}] - [p_1^1][\bar{X}] = 0 \qquad (\mathbf{Ec.\,18})$$

$$v^1[p_3^1][\bar{X}] - [p_2^1][\bar{X}] = 0 \qquad (\mathbf{Ec.\,19})$$

Si realizamos el mismo proceso para una segunda imagen I_2 obtenemos:

$$u^2[p_3^2][\bar{X}] - [p_1^2][\bar{X}] = 0 \qquad (\mathbf{Ec.\,20})$$

$$v^2[p_3^2][\bar{X}] - [p_2^2][\bar{X}] = 0 \qquad (\mathbf{Ec.\,21})$$

Podemos escribir en forma matricial las ecuaciones 18, 19, 20 y 21

$$\begin{bmatrix} u^1[p_3^1] - [p_1^1] \\ v^1[p_3^1] - [p_2^1] \\ u^2[p_3^2] - [p_1^2] \\ v^2[p_3^2] - [p_2^2] \end{bmatrix} [\bar{X}] = 0 \qquad (\mathbf{Ec.\,22})$$

$$D\bar{X} = 0 \qquad (\mathbf{Ec.\,23})$$

Si resolvemos para el espacio nulo de la matriz *D*, encontraremos el vector $\bar{X}$. Este vector tiene 4 componentes. Para transformar de coordenadas homogéneas a coordenadas en 3 dimensiones, dividimos las tres primeras coordenadas para la cuarta como se muestra en la ecuación 24 (Collins, 2007). De esta manera, usando las matrices de proyección de dos imágenes y las coordenadas de los features que ya han sido correspondidos, podemos formar la matriz *D*, resolver para su espacio nulo y encontrar un punto en 3D.

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \Longrightarrow \begin{bmatrix} X/W \\ Y/W \\ Z/W \end{bmatrix} \qquad (\mathbf{Ec.\,24})$$

Se implementa este proceso para todos los features encontrados en cada par de imágenes. Para aumentar el número de puntos en la reconstrucción, se realiza el proceso en dos etapas. En la primera, se utilizan imágenes secuenciales como se indica en la Figura 20 ; para un número de *n* imágenes se debe realizar el proceso *n-1* veces. En la segunda etapa, se realiza el procedimiento saltándose una imagen, como se indica en la Figura 21; en este

caso, el proceso se realiza *n-2* veces. Se implementa el código en Matlab para un anillo de fotografías de 16 y 48 imágenes.

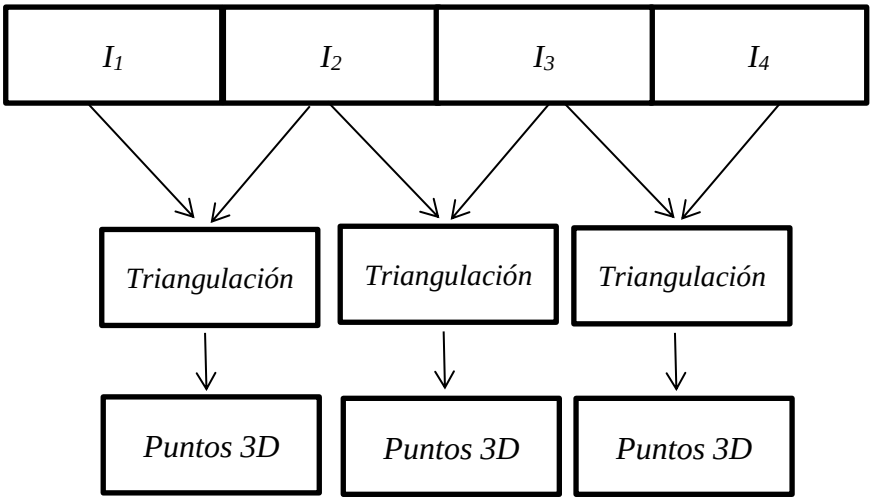


Figura 20. Representación de la primera etapa en el proceso de triangulación para 4 imágenes.

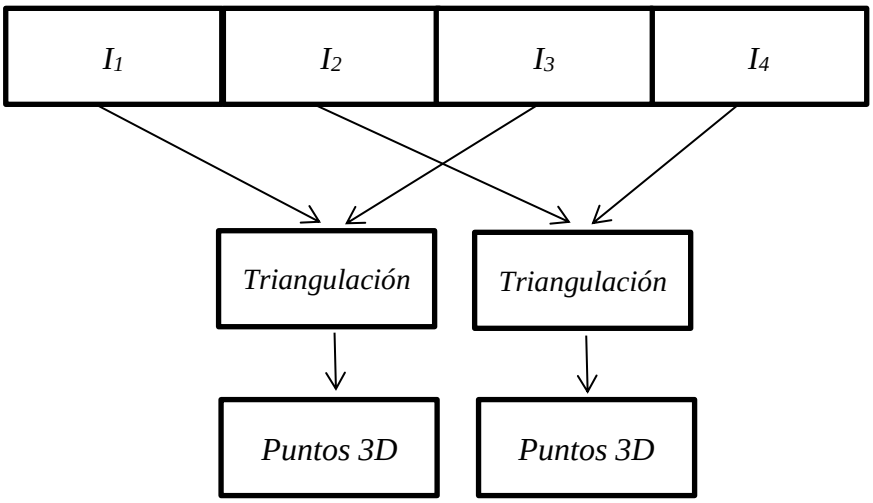


Figura 21. Representación de la segunda etapa en el proceso de triangulación para 4 imágenes.

La Figura 22 muestra las vistas frontal y lateral del objeto para comparar con los resultados obtenidos en la reconstrucción. En las Figuras 23 y 24 se pueden ver los resultados obtenidos después de realizar la triangulación para un anillo de fotografías de 16 y 48 imágenes respectivamente. Podemos observar que lo primero en destacar son las espinas del dinosaurio ya que los dos algoritmos de detección de features encontraron varios puntos en esos lugares.

Figura 22. Vistas lateral y frontal del objeto.

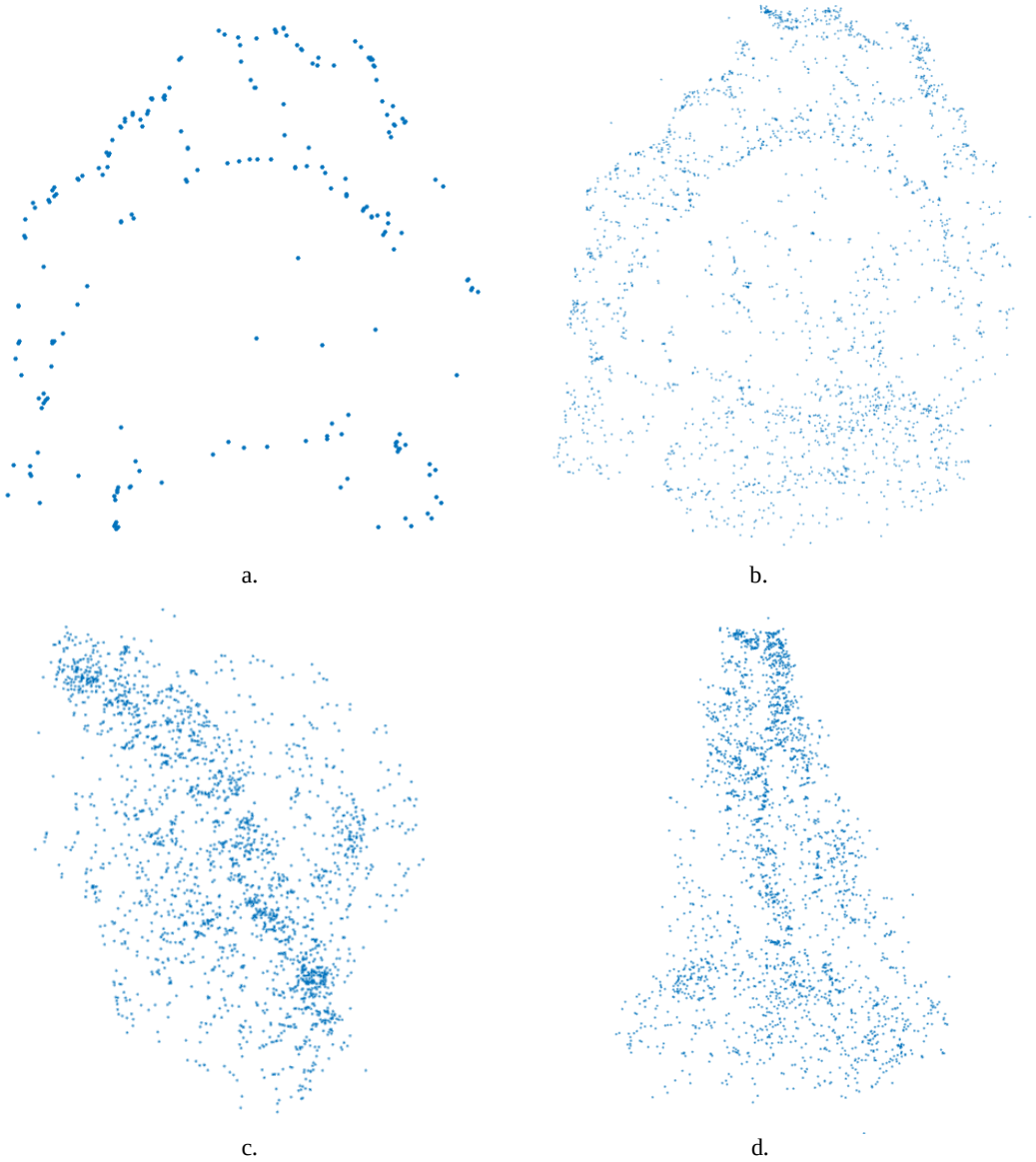


Figura 23. Resultados para el proceso de triangulación implementado en anillo de 16 imágenes. a) Puntos obtenidos al usar 2 imágenes de 16. b) Al usar 16 imágenes de 16, vista lateral. c) Vista superior. d) Vista frontal.

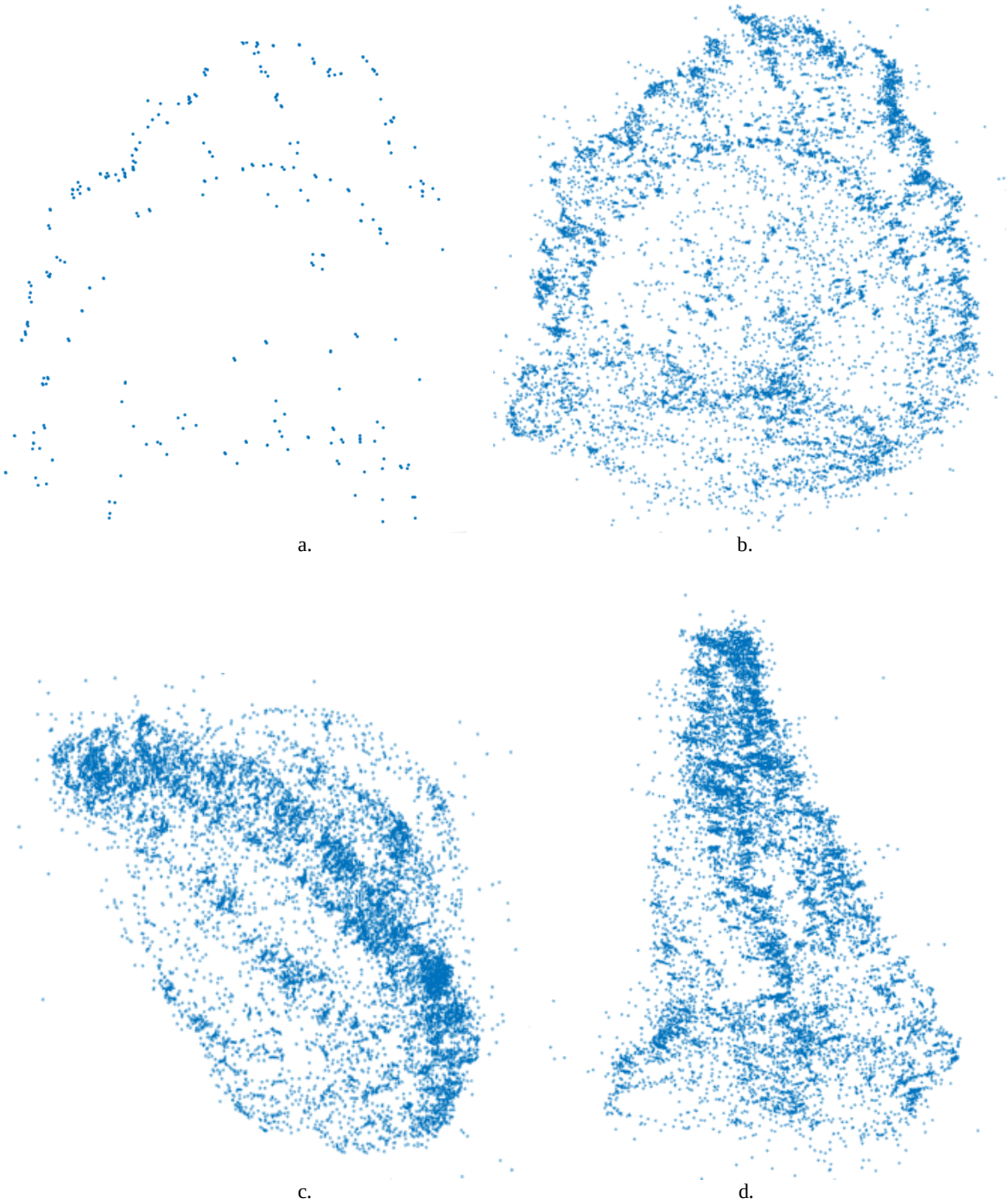


Figura 24. Resultados para el proceso de triangulación implementado en anillo de 48 imágenes. a) Puntos obtenidos al usar 2 imágenes de 48. b) Al usar 48 imágenes de 48, vista lateral. c) Vista superior. d) Vista frontal.

La Figura 24 presenta varios puntos erróneos en su reconstrucción, si somos más exigentes con el threshold obtenemos una imagen más “limpia” como se muestra en la Figura 25. Ya que se trata de un objeto en 3D, tenemos acceso a vistas que las fotografías no proporcionan como se muestra en las Figuras 23.c, 24.c, y 25.c.

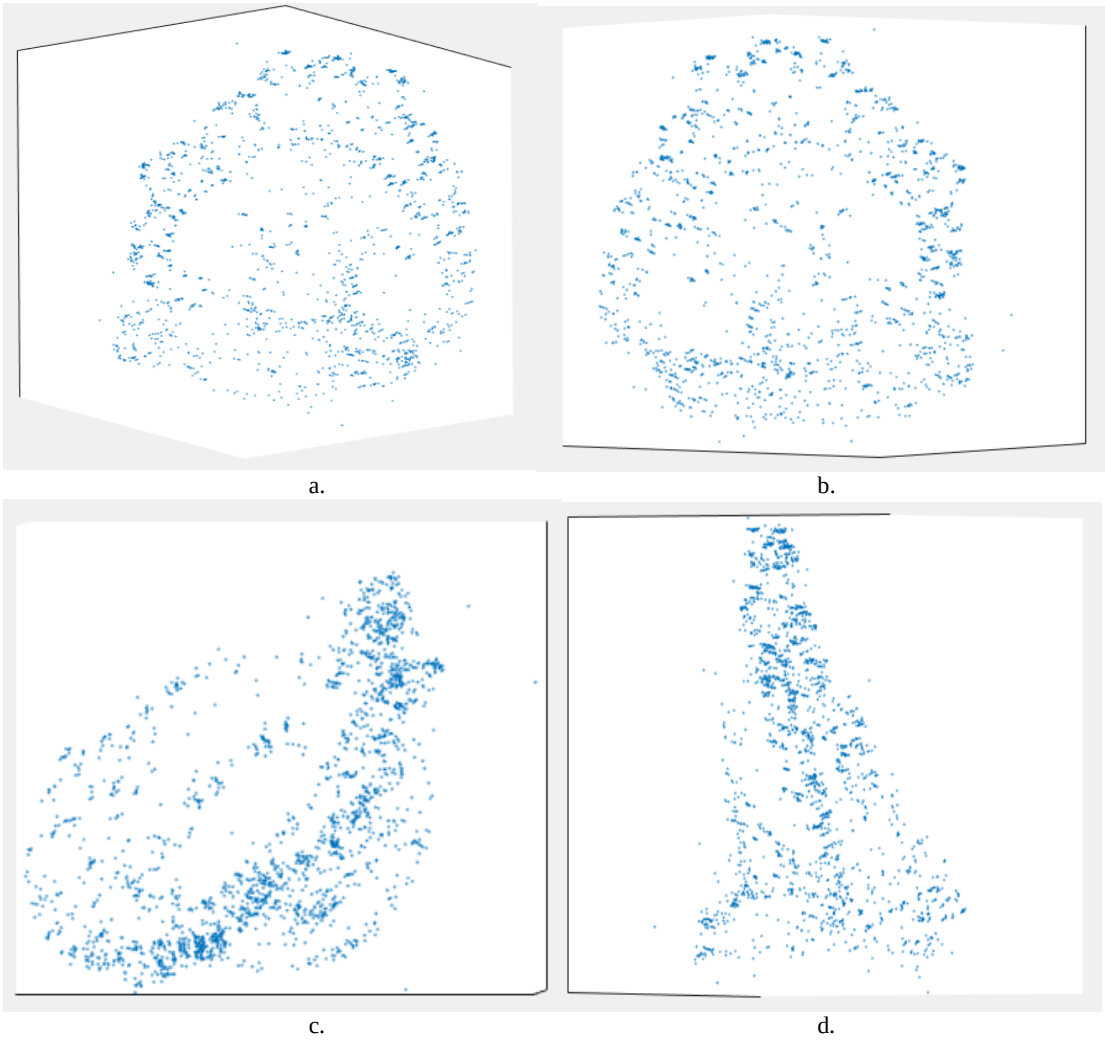


Figura 25. Resultados para el proceso de triangulación en anillo de 48 imágenes con el threshold incrementado para obtener una imagen más limpia. a) Vista lateral. b) Vista lateral. c) Vista superior. d) Vista frontal.

CONCLUSIONES

Las Figuras 24 y 25 muestran que se desarrolló este trabajo satisfactoriamente. Se obtuvo con éxito un conjunto de puntos en 3D a partir de fotografías y sus matrices de proyección. Sin embargo, se pueden aplicar mejoras a los algoritmos ya implementados. En la detección de features por ejemplo, un paso importante es robustecerlos y hacerlos invariantes a transformaciones afines. Con esto no solo mejoraríamos el proceso de matching, también sería posible implementar la triangulación con más etapas y menos errores. Si el descriptor utilizado es simplemente una ventana alrededor del feature detectado las malas correspondencias son muchas. Es necesaria la implementación de un descriptor más robusto como SIFT, de esta manera podría no implementarse el proceso de tracking y seguir obteniendo buenos resultados. En el matching implementado se usó un solo tamaño de ventana para corresponder todos los features. En el caso de los blobs, se puede ajustar la ventana del matching de acuerdo a la escala con la que fue detectado. La implementación de tracking disminuye el número de correspondencias erróneas, pero no todas. Una mejor implementación restringe el matching entre los features con geometría epipolar.

Tabla 1. Tiempos de compilación de cada función.

	16 imágenes	**48 imágenes**
Harris	9,65 min.	26,58 min.
Blobs (sigma 1,3,5)	1,91 min.	5,33 min.
Matching	6,05 min.	17,20 min.

Podrían implementarse mejoras en la eficiencia con la que el código fue escrito. Sin embargo, el objetivo principal era tener un programa que funcione correctamente antes que

un programa eficiente. La implementación en algunas partes podría decirse que fue de “fuerza bruta”, ya que no se implementó un matching inteligente ni tampoco un buen descriptor. En la Tabla 1 se encuentran los tiempos que demora en compilar cada función. Como se menciona al principio de este trabajo, la reconstrucción e 3D es un proceso modular, cualquier mejora implementada en cada sección mejorará el proceso como un todo. Una de las desventajas del método presentado en este trabajo es que tiene muchas variables que tienen que ajustarse, en especial los valores de threshold. El éxito de los algoritmos expuestos está estrechamente ligado con el valor del threshold a usarse.

Los puntos en 3D pueden usarse para seguir con la reconstrucción. Existen algoritmos que pueden generar una superficie a partir de los puntos obtenidos. Con una superficie reconstruida es posible imprimir el dinosaurio en 3D. Gracias a que la visión computarizada es un campo muy activo en la investigación, también existen procesos para poder calcular la matriz de proyección únicamente a partir de las imágenes. Con esto es posible la reconstrucción en 3D para cualquier objeto fotografiado. La Tabla 2 presenta los algoritmos implementados y en rojo los procesos que mejoran y llevan la reconstrucción un paso más adelante.

Tabla 2. Algoritmos para reconstrucción en 3D.

Detección de Features	**Descriptor**	**Matching**		
Harris	Ventana	SSD	**Con matriz de proyección**	Triangulación
Blobs	MOPS	Ratio distance		
Harris afín	MSER	Geometric Hashing	**Sin matriz de proyección**	Bundle Adjustment
Blobs afín	SIFT			RANSAC
				Structure from Motion

REFERENCIAS BIBLIOGRÁFICAS

Agarwal, S. Snavely, N. & Seitz. S. (2005). *Building Rome in a Day.* Obtenido el 2 de noviembre de 2016 de https://grail.cs.washington.edu/rome/rome_paper.pdf

Collins, R. (2007). *The Least-Squares Triangulation by SVD.* Obtenido el 20 de noviembre del 2016 de http://cmp.felk.cvut.cz/cmp/courses/TDV/2012W/lectures/tdv-2012-07-anot.pdf

Forsyth, D. A. & Ponce, J. (2003). *Computer Vision, A Modern Approach.* Estados Unidos, Prentice Hall.

Furukawa, Y. (2016). *Cameras 1*. Missouri. Estados Unidos. Obtenido el 12 de noviembre de 2016 de http://www.cse.wustl.edu/~furukawa/cse559a/2016_fall/notes/01-cameras.pdf

Furukawa, Y. (2016). *Cameras 2*. Missouri. Estados Unidos. Obtenido el 12 de noviembre de 2016 de http://www.cse.wustl.edu/~furukawa/cse559a/2016_fall/notes/04-cameras-2.pdf

Furukawa, Y. (2016). *Cameras 3*. Missouri. Estados Unidos. Obtenido el 12 de noviembre de 2016 de http://www.cse.wustl.edu/~furukawa/cse559a/2016_fall/notes/05-cameras-3.pdf

Furukawa, Y. & Ponce, J. (2008). *Accurate, Dense and Robust Multi-View Stereopsis.* Obtenido el 8 de noviembre del 2016 de http://www.cse.wustl.edu/~furukawa/papers/pami08a.pdf

Jacobs, D. (2005). *Image Gradients*. Maryland. Estados Unidos. Obtenido el 8 de septiembre de 2016 de http://www.cs.umd.edu/~djacobs/CMSC426/ImageGradients.pdf

Jacobs, D. (2005). *Correlation and Convolution*. Maryland. Estados Unidos. Obtenido el 8 de septiembre de 2016 de http://www.cs.umd.edu/~djacobs/CMSC426/ImageGradients.pdf

Simek, K. (2012). *Dissecting the Camera Matrix, The Extrinsic Matrix*. Australia. Sightations. Obtenido el 8 de diciembre del 2016 de http://ksimek.github.io/2012/08/22/extrinsic/

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications. Washington.* Obtenido el 16 de mayo del 2016 de http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

Szeliski, R. & Seitz, S. (2008). *Image Features.* Obtenido el 9 de septiembre de 2016 de http://courses.cs.washington.edu/courses/cse576/08sp/lectures/features.pdf

Lazebnik, S. (2010). *Corner and blob detection.* Obtenido el 14 de noviembre de 2016 de http://www.cs.unc.edu/~lazebnik/spring09/lec07_corner_blob.pdf

Lazebnik, S. (2010). *Single View Geometry.* Obtenido el 14 de noviembre de 2016 de http://www.cs.unc.edu/~lazebnik/spring09/lec11_single_view.pdf