



Universidad
Zaragoza

TRABAJO FIN DE MASTER
DE INGENIERÍA INFORMÁTICA

Aprendizaje de estructuras 3D a partir de secuencias de imágenes.

Learning 3D structures from monocular sequences

Directores/Advisors

Javier Civera Sancho (dir)

Luis Montesano del Campo (dir)

Autor/Author

José María Fácil Ledesma



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

ESCUELA DE INGENIERÍA Y ARQUITECTURA

Zaragoza, Septiembre de 2016



**DECLARACIÓN DE
AUTORÍA Y ORIGINALIDAD**

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./D^a. José María Fácil Ledesma

con nº de DNI 18049340-K en aplicación de lo dispuesto en el art.

14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo

de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la

Universidad de Zaragoza,

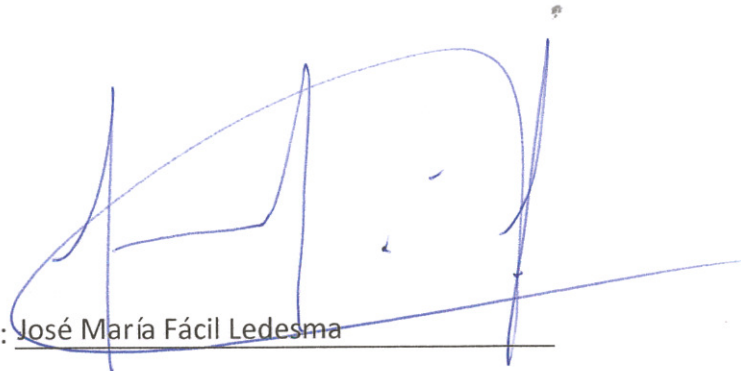
Declaro que el presente Trabajo de Fin de (Grado/Máster)

Master _____, (Título del Trabajo)

Aprendizaje de estructuras 3D a partir de secuencias de imágenes

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 22 de Septiembre de 2016



Fdo: José María Fácil Ledesma

Resumen

La reconstrucción 3D densa a partir de secuencias monoculares es una tecnología clave para varias aplicaciones y todavía un problema de investigación abierto. Este trabajo aprovecha resultados recientes en estimación de profundidad a partir de una sola vista utilizando CNN (Redes neuronales convolucionales) y los fusiona con la estimación de un método directo multi-vista. Ambas aproximaciones muestran fortalezas complementarias. Primero, los métodos basados en múltiples vistas son muy precisos en zonas con mucha textura en secuencias de alto paralaje. Segundo, el método que estima la profundidad a partir de una imagen captura muy bien la estructura local, incluidas las áreas sin textura, aunque carece de coherencia global.

La fusión de estas dos estimaciones que proponemos tiene varios retos. En primer lugar, las dos profundidades están relacionadas por una deformación no rígida que depende en el contenido de la imagen. Y en segundo, la selección de los puntos de alta precisión del método multi-vista puede ser complicada en configuraciones de bajo paralaje. Presentamos una contribución a los dos problemas. Nuestros resultados en los conjuntos de datos públicos de NYU y TUM muestran que nuestro algoritmo mejora a las dos aproximaciones por separado.

Resumen

Dense 3D mapping from a monocular sequence is a key technology for several applications and still a research problem. This work leverages recent results on single-view CNN-based depth estimation and fuses them with direct multi-view depth estimation. Both approaches present complementary strengths. Multi-view depth estimation is highly accurate but only in high-texture and high-parallax cases. Single-view depth captures the local structure of mid-level regions, including textureless areas, but the estimated depth lacks global coherence.

The single and multi-view fusion we propose has several challenges. First, both depths are related by a non-rigid deformation that depends on the image content. And second, the selection of multi-view points of high accuracy might be difficult for low-parallax configurations. We present contributions for both problems. Our results in the public datasets of NYU and TUM shows that our algorithm outperforms the individual single and multi-view approaches.

Índice general

1. Introducción	1
2. Estado del arte	6
2.1. Profundidad a partir de múltiples vistas	6
2.2. Profundidad a partir de una única vista	7
3. Notación	8
3.1. Modelo proyectivo y notación	8
3.2. Redes Neuronales Artificiales	12
3.2.1. Modelo de una RNA	13
3.2.2. Estructura de las capas de una RNA	14
3.2.3. Redes Neuronales 2D con imágenes	16
4. Fusión de profundidad	19
4.1. Profundidad a partir de múltiples vistas	21
4.2. Profundidad a partir de una única vista	22
4.3. Deformación no rígida	23
4.4. Selección de puntos de bajo error	26
5. Resultados experimentales	28
5.1. Evaluación de la fusión	28
5.2. Evaluación y selección de odometría RGB-D	34
5.3. Evaluación de tiempos	34
5.4. Herramientas y tecnología utilizada	35
6. Conclusiones	36
Anexos	41
A. Artículo enviado al IEEE ICRA	41

Índice de figuras

1.1. Aplicaciones y problemas	1
1.2. Reconstrucciones 3D	2
1.3. Ventajas de los mapas densos en RA	3
1.4. Visión general del problema	4
3.1. Modelo de cámara de Tsai	9
3.2. Composición de transformaciones	12
3.3. Funciones de activación	13
3.4. Red neuronal artificial	14
3.5. Tipos de capas	15
3.6. Convolución a una imagen	16
3.7. Pooling a un mapa de características	17
4.1. Visión general del trabajo	19
4.2. Histograma del error de profundidad	20
4.3. Red neuronal convolucional profunda	22
4.4. Influencia no normalizadas	23
4.5. Influencia normalizada	24
4.6. Influencia normalizada, en detalle	25
5.1. Estimaciones de profundidad en el NYUv2	30
5.2. Imágenes de error en el NYUv2	31
5.3. Estimaciones de profundidad en el TUM	32
5.4. Imágenes de error en el TUM	32
5.5. Diagrama de cajas y bigotes de los errores	33

Índice de tablas

4.1. Error mediano de dos aproximaciones para estimar profundidad	20
5.1. Resultados de la fusión	29
5.2. Resultado de la fusión con selección de puntos manual	29
5.3. Evaluación SLAM y Odometría RGB-D	34

1. Introducción

La Visión por Computador es el campo de la computación cuyo objetivo es conseguir que los ordenadores vean. Dicho de otra manera, trata de extraer información del mundo a través de las imágenes obtenidas por una cámara. La visión por computador cubre un amplio abanico de problemas y técnicas tales como el reconocimiento de patrones, la construcción de modelos del entorno, la identificación y seguimiento de objetos o la localización. Tiene aplicaciones muy diversas tanto en robótica, en imagen médica, en fotogrametría, reconocimiento facial (Figura 1.1a), tareas de vigilancia (Figura 1.1b), realidad aumentada y virtual entre otras muchas.

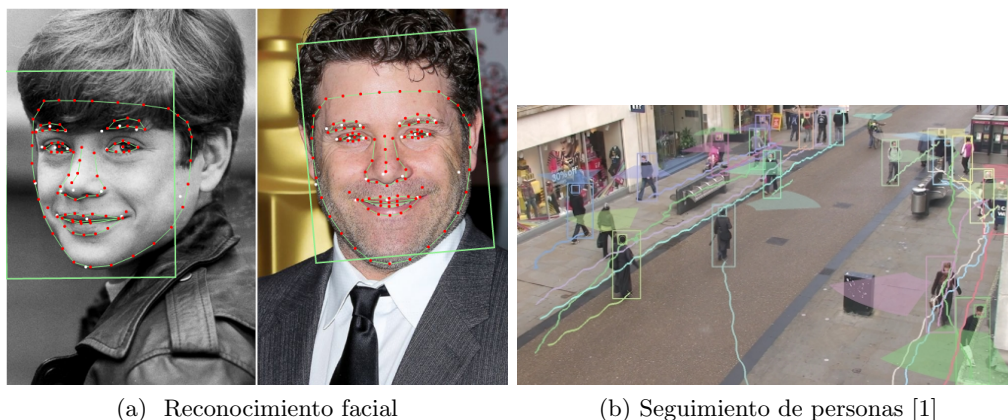


Figura 1.1: Aplicaciones y problemas de la visión por computador

Uno de los problemas de investigación fundamentales en visión por computador es la estimación *online*¹, precisa y densa de la reconstrucción 3D de una escena a partir de una secuencia de imágenes. El problema tiene actualmente gran relevancia, y es una tecnología clave en varias aplicaciones en mercados emergentes (realidad aumentada y virtual, coches autónomos y robótica en general). Los algoritmos tradicionales de reconstrucción 3D usan puntos característicos. El motivo es que son fáciles de emparejar entre distintas vistas. Sin embargo el resultado de estas técnicas no es un mapa completo de la escena, solamente de algunos puntos dispersos. Un ejemplo de estos dos tipos de mapas

¹*online* - Utilizado para representar que la estimación del mapa 3D se hace con una ventana de imágenes en la secuencia, en lugar de con la secuencia entera.

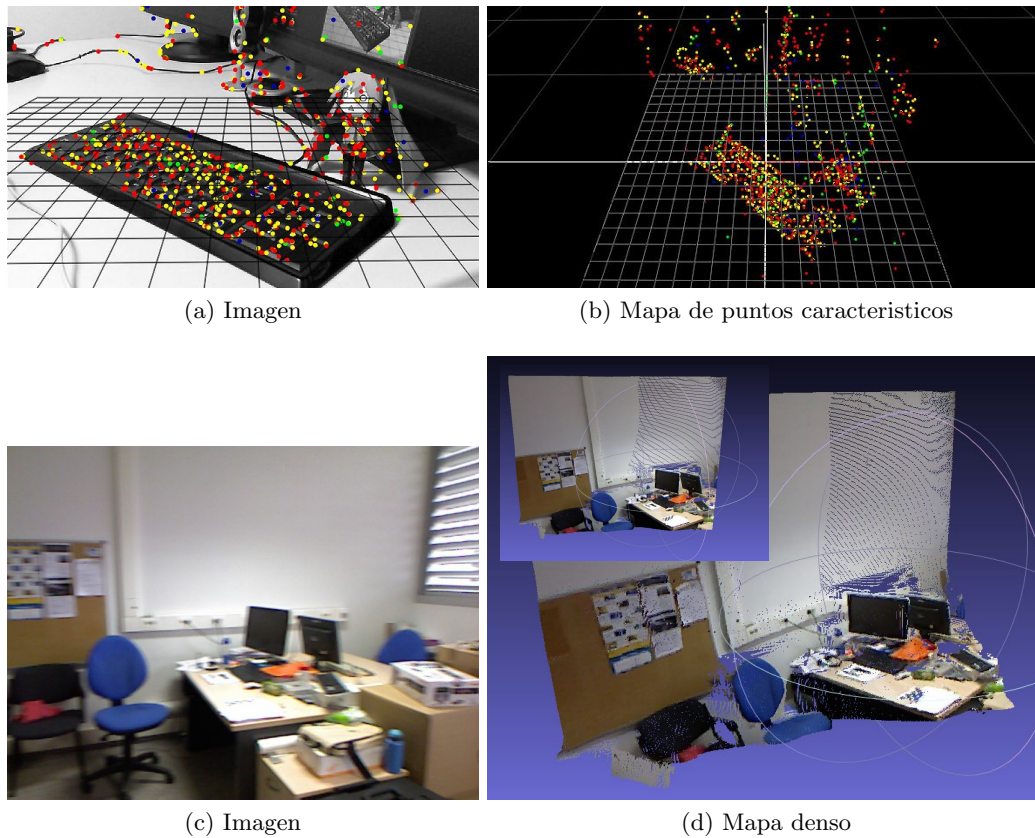


Figura 1.2: Comparación entre mapas basados en puntos salientes y mapas densos. En la primera fila se muestra el caso de una reconstrucción de puntos salientes. La imagen a) muestra los puntos seleccionados de la imagen, la imagen b) muestra la reconstrucción 3D de los puntos de la imagen. En la segunda fila se muestra un caso de reconstrucción 3D densa. En la imagen c) la imagen de la cual se hace la reconstrucción, la imagen d) en mapa denso.

se puede ver en la Figura 1.2.

Los mapas completamente densos desarrollados en los últimos 5 años presentan múltiples ventajas con respecto a los basados en puntos característicos. Por ejemplo en aplicaciones como la realidad aumentada, el realismo con el que las inserciones virtuales que son incorporadas a la imagen se ve cualitativamente mejorado. Los mapas densos permiten hacer inserciones que respeten más la física y sean coherentes objetos en la escena (i.e. que las inserciones no queden flotando sino que este apoyadas y que si un cuerpo pasa por delante de la cámara se ocluya la inserción virtual entre otras). En la Figura 1.3 se muestran ejemplos de realidad aumentada sin y con mapa denso. El primero es el reciente vídeo-juego *Pokémon Go* (Figura 1.3a). En la primera imagen (izquierda en la figura) la inserción virtual no esta apoyada en una superficie plana, sino que esta flotando



(a) Realidad aumentada sin mapa denso



(b) Realidad aumentada con mapa denso [2]

Figura 1.3: Ejemplos de realidad aumentada con o sin mapa denso.

en el aire. En la segunda imagen (derecha en la figura) la inserción se puede ver de dos formas: 1) La inserción virtual esta detrás de la barandilla en cuyo caso no se ve ocluida por esta y por lo tanto no es coherente y 2) Esta delante de la barandilla en cuyo caso esta flotando en el aire y por lo tanto tampoco es coherente.

El segundo ejemplo es una inserción de realidad aumentada con mapa 3D denso de la escena (Figura 1.3b). Usando un mapa denso elementos virtuales están mucho mejor insertados en la escena. Permite que los elementos virtuales proyecten sombras y/o reflejos de la escena. Incluso si el mapa de un entorno es completo (i.e una habitación entera) las inserciones virtuales pueden reflejar el entorno que hay detrás de la cámara.

La precisión de los métodos densos *multi-view*² descritos anteriormente esta limitada principalmente por tres factores:

1. La configuración geométrica entre la escena y la secuencia, logrando muy poca precisión en configuraciones de bajo paralaje. El paralaje es el angulo formado por dos rayos desde dos posiciones de la cámara a un punto de la escena. El paralaje se reduce cuando las cámaras tienen una traslación relativa pequeña o la profundidad del punto es alta.

²*multi-view* - Basados en múltiples vistas de la escena.

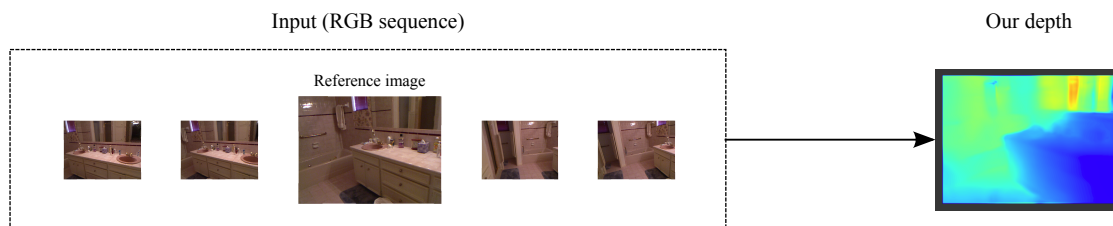


Figura 1.4: Visión general del problema. Nuestra entrada son un conjunto de vistas de una escena tomadas por una cámara monocular. El objetivo es estimar una imagen de profundidad de la imagen de referencia. La imagen de profundidad se representa en una escala continua de colores de azul oscuro para los puntos más cercanos, verde para los puntos intermedios y rojo para los puntos lejanos.

2. La calidad de las correspondencias entre diferentes vistas (correspondencia de píxeles en el caso de reconstrucciones densas), que solo pueden ser estimadas de forma fiable para píxeles de alto gradiente.
3. La función de regularización, típicamente la *Total Variation norm* (TV), que suaviza la predicción de profundidad para eliminar ruidos de alta frecuencia. Logra baja precisión para grandes superficies sin textura por la mala calidad de las correspondencias y por lo tanto de las predicciones.

Una alternativa son los métodos *single-view*³ que estiman profundidades para todos los píxeles de una imagen. Estos métodos han experimentado recientemente una mejora cualitativa en su precisión gracias al uso de redes neuronales profundas [3]. Su precisión es todavía inferior a los métodos *multi-view* en puntos de alta textura y alto paralaje. Sin embargo, como argumentaremos en este trabajo, los métodos *single-view* superan a los métodos *multi-view* en las áreas de baja textura.

El objetivo de este trabajo es explotar la información de la estimación de profundidad basada en una vista (*single-view*) y la basada en múltiples vistas (*multi-view*) para obtener una profundidad mejorada incluso en secuencias de bajo paralaje y áreas de poca textura (ver Figura 1.4). La contribución científica de este trabajo es un algoritmo que fusiona estas dos estimaciones de profundidad complementarias. Existen dos principales retos en esta tarea. Primero, la distribución del error en las estimaciones a partir de una sola vista tienen tendencias locales, ya que depende del contenido de la imagen y no de la configuración geométrica. *Single-view* y *multi-view* están por lo tanto relacionadas por una deformación no rígida, basada en el contenido de la escena. Y segundo, dado que pretendemos desarrollar un método general para bajo y alto paralaje, modelar la precisión de los métodos *multi-view* no es trivial. Proponemos un método basado en una interpolación pesada de la estructura local de la estimación *single-view* basado en la calidad y área de influencia del mapa semi-denso de profundidad a partir de *multi-view*. Lo evaluamos en dos conjuntos de datos públicos (NYU y TUM) y los resultados

³ *single-view* - Basados en una única vista de la escena.

muestran que nuestro algoritmo de fusión mejora cuantitativa y cualitativamente a los dos métodos, *single-view* y *multi-view*, por separado.

El resto de esta memoria se estructura de la siguiente manera. El capítulo 2 describe los trabajos relacionados más relevantes. El capítulo 3 introduce la notación y algunos conocimientos previos. El capítulo 4 motiva y detalla el algoritmo de fusión entre *multi-view* y *single-view* presentado en este trabajo. El capítulo 5 presenta los resultados experimentales y, finalmente el capítulo 6 contiene la conclusión del trabajo.

2. Estado del arte

En este capítulo describimos algunos de los trabajos más relevantes de estimación de la profundidad a partir de imágenes tomadas por una cámara monocular. Dado que este trabajo aprovecha el conocimiento de dos áreas diferentes dentro del mismo campo, *single-view* y *multi-view*, vamos a organizar este capítulo en dos secciones. En la sección 2.1 nos centraremos en los trabajos de estimación de profundidad a partir de múltiples vistas (*multi-view*). En la sección 2.2 hablaremos de los trabajos que estiman la profundidad a partir de una única imagen (*single-view*).

2.1. Profundidad a partir de múltiples vistas

[4, 5, 6] son los primeros trabajos que alcanzaron reconstrucciones 3D densas y en tiempo real a partir de secuencias tomadas por una cámara monocular. Algunos de los aspectos más relevantes son la minimización directa del error fotométrico – en lugar del error geométrico de reconstrucciones dispersas que tradicionalmente se usa – y la regularización de la estimación añadiendo la norma *Total Variation* a la función de coste.

[7] demostró que la regularización TV no alcanza buena precisión en largas áreas de poca textura, y propuso un método de regularización por partes añadiendo la asunción de partes planas en la escena; los parámetros de los planos se extraen de una triangulación de super-píxeles a partir de diferentes vistas [8] o de la estimación de la distribución de la escena [9]. [10] propone una regularización que fuerza restricciones entre partes afines de la escena incluso en píxeles separados. [11] selecciona la mejor regularización entre un conjunto de funciones de regularización usando, para la validación, datos dispersos de la escena proveídos por un láser. [12] añade las primitivas 3D dispersas extraídas de los datos de [13] como restricción en la regularización. Comparado con estos trabajos, nuestra propuesta es la primera donde la información añadida a la profundidad *multi-view* es completamente densa, basada en los datos a partir de *single-view*; y además no se basa en sensores adicionales o en asunciones de paralaje o *Manhattan* y partes planas.

Debido a la dificultad de estimar mapas de profundidad completamente densos de forma precisa a partir de vistas tomadas por una cámara monocular hay varios métodos

que proponen estimar la profundidad los píxeles de alto gradiente exclusivamente (e.g., [14]). Aunque esta propuesta produce mapas de profundidad con más densidad que los métodos tradicionales basados en puntos salientes (e.g., [15]), todavía son modelos incompletos de la escena y por ello su aplicabilidad podría estar más limitada.

2.2. Profundidad a partir de una única vista

La profundidad también puede ser estimada a partir de una sola vista; estos métodos utilizan diferentes pistas en la imagen, por ejemplo a partir del enfoque de la cámara (e.g., [16]) o pistas sobre la perspectiva de la escena (e.g., [17]). Los métodos basados en aprendizaje, como el que usamos en este trabajo, básicamente descubren patrones en la imagen que son relevantes para una regresión precisa de la profundidad.

El trabajo pionero de Saxena *et al.* [18] consistía en entrenar un MRF para modelar profundidad a partir de un conjunto de características globales y locales de la imagen. Más recientemente, Eigen *et al.* [3] presentaron dos redes neuronales *convolucionales* profundas (CNN) apiladas, la primera para predecir profundidad global y la segunda para refinar la primera de forma local. Siguiendo la línea de este trabajo [19], recientemente presentó una red neuronal a tres escalas para predecir profundidad, la normal de las superficies y etiquetado semántico de la escena. Liu *et al.* [20] usan una estructura de redes neuronales y CRF (*Conditional Random Field*) para estimar profundidad. Las redes neuronales se usan para aprender los potenciales unitarios y por pares, que relacionan diferentes super-píxeles, que el CRF usa para predecir la profundidad de la escena.

3. Notación

En este capítulo se va a introducir brevemente en que consiste el problema de estimación de profundidad densa a partir de imágenes RGB tomadas por una cámara monocular. Más adelante, en el capítulo 4 se detallará la formulación concreta que hemos utilizado en este trabajo.

La reconstrucción de una escena a partir de imágenes RGB tomadas por una cámara monocular consiste en estimar la profundidad de cada píxel de una imagen. Existen dos formas de afrontar este problema. Una de ellas son los métodos basados en múltiples vistas de la escena, en adelante nos referiremos a ellos como métodos *multi-view*. Estos métodos aprovechan el movimiento de la cámara a lo largo de la secuencia para, triangulando los rayos proyectantes de las dos vistas, estimar la profundidad de los puntos. La entrada de estos algoritmos es una secuencia de imágenes y la salida es una imagen de profundidad correspondiente a una imagen de referencia dentro de la secuencia. Por otro lado, están los métodos que utilizan únicamente una imagen, en adelante nos referiremos a estos métodos como *single-view*. Estos métodos no triangulan la posición 3D de los puntos. Los métodos *single-view* requieren el uso de características más globales en la imagen, como los ángulos de las líneas y la perspectiva, formas y tamaños de los objetos, oclusiones para realizar la estimación.

En este trabajo utilizamos tanto un método *multi-view* como uno *single-view* y fusionamos los resultados por lo que en las siguientes secciones vamos a detallar algunos conocimientos y notación de cada uno de ellos. Primero, en la Sección 3.1 explicamos el modelo proyectivo en el que se basa nuestra estimación de profundidad a partir de múltiples vistas. Segundo, dado que en este trabajo hemos utilizado Redes Neuronales Profundas para la estimación de profundidad *single-view*, en la Sección 3.2 vamos a introducir algunos conocimientos previos sobre redes neuronales y los modelos que más utilizamos en este trabajo.

3.1. Modelo proyectivo y notación

La base principal del modelo de cámara que utilizamos es la proyección de 3D a 2D descrito en el modelo de cámara de Tsai [21]. Una representación esquemática del modelo

se muestra en la figura 3.1. El modelo de Tsai se compone de 6 parámetros extrínsecos

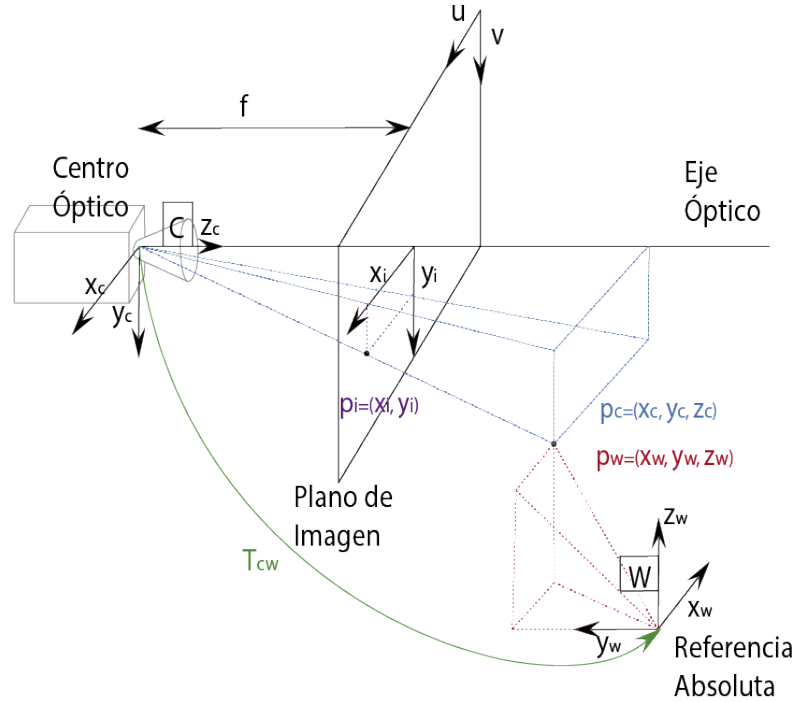


Figura 3.1: Modelo de cámara de Tsai

y 5 parámetros intrínsecos. Los parámetros extrínsecos describen la localización en x , y y z de la cámara respecto a el sistema de coordenadas absoluto, que es el que usaremos como punto de referencia. Son 3 de rotación (r_x , r_y , r_z) y 3 de traslación (t_x , t_y , t_z). Los parámetros intrínsecos (f , C_x , C_y , s , K) dependen de la construcción interna de la cámara. Como suele ser habitual, para asegurar que los píxeles sean cuadrados, el parámetro s (*skew coefficient*), que determina el ángulo entre los ejes x e y de los píxeles, ha sido fijado a 0, y el *aspect ratio* ha sido fijado a 1. El parámetro de distorsión $K = (k_1, \dots, k_n)$ corrige la distorsión radial de la lente. El parámetro f es la distancia focal, que determina la posición del plano de la imagen con respecto al centro óptico de la cámara. Los parámetros C_x y C_y representan las coordenadas en el sensor de la cámara de la intersección con eje óptico (el centro óptico del sensor).

Los 5 parámetros intrínsecos de la cámara son constantes, necesitan ser estimados mediante técnicas de calibración. En el algoritmo que presentamos se suponen conocidos. Por otro lado los 6 parámetros extrínsecos representan el localización de la cámara y por lo tanto varían con su movimiento.

Como se puede ver en la figura 3.1, en el modelo de Tsai la posición de la cámara en el espacio viene dada por la posición del centro óptico, también se utilizará para nombrarlo

O_c , como origen (un punto) del sistema de coordenadas de la cámara respecto al mundo:

$$O_c = (x_w, y_w, z_w). \quad (3.1)$$

La posición del centro óptico define la traslación que tiene la cámara respecto al sistema de referencia:

$$O_c = (t_x, t_y, t_z), \quad (3.2)$$

que coincide con la posición frontal de la cámara y el eje z_c de la cámara coincide con la dirección y el sentido del eje óptico de la cámara. El plano de la imagen es paralelo al plano generado por los vectores x_c e y_c del sistema de coordenadas de la cámara, esta a una distancia f de O_c en la dirección y sentido del eje z_c , donde f es la distancia focal.

La relación entre un punto definido en el sistema de coordenadas del mundo ($p_w = (x_w, y_w, z_w)$) y el mismo punto en el plano de la imagen de la cámara ($p_i = (x_i, y_i)$) esta compuesta por una secuencia de transformaciones. Una transformación del punto de coordenadas del mundo a coordenadas de la cámara y una proyección del punto al plano de la cámara.

La primera consiste en transformar el punto p_w al sistema de coordenadas de la cámara ($p_c = (x_c, y_c, z_c)$):

$$p_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R_{cw} \cdot p_w + t_{cw} = R_{cw} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (3.3)$$

donde

$$R_{cw} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} = Rot(r_x)Rot(r_y)Rot(r_z), \quad (3.4)$$

siendo R_{cw} una matriz de rotación 3×3 . Los vectores columna de una matriz de rotación representan las direcciones normalizadas de los ejes x , y y z del sistema de referencia al que ejerce la rotación sobre el sistema base a partir del cual la define. En el ejemplo de la ecuación 3.4 el vector $[r_1, r_4, r_5]'$ define la dirección del eje x del sistema de la cámara c con respecto al sistema de ejes absoluto w .

Una matriz de rotación representa una rotación en el espacio euclídeo, no es una representación mínima ya que se representan 3 grados de libertad (r_x , r_y , r_z) mediante una matriz 3×3 . Esta sobrep parametrización se traduce en una serie de restricciones que una matriz de rotación debe cumplir:

- Es una matriz ortogonal, lo que significa que su matriz inversa es igual a su matriz transpuesta.
- Su determinante es igual a 1.

En adelante las transformaciones como la mostrada en la ecuación 3.3 se representarán como transformaciones homogéneas utilizando la siguiente notación:

$$\dot{p}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \omega x_c \\ \omega y_c \\ \omega z_c \\ \omega \end{bmatrix} = T_{cw} \cdot \dot{p}_w, \quad (3.5)$$

donde \dot{p} es la representación homogénea de un punto en el espacio 3D. Las coordenadas homogéneas se utilizan para describir un punto en el espacio proyectivo. En el espacio 3D proyectivo la representación de un punto viene dada por un vector de cuatro elementos, siendo ω un factor de escala que generalmente fijaremos a 1. T_{cw} representa la matriz de transformación de la cámara respecto al mundo, para trabajar con ella es necesario utilizar coordenada homogéneas (ver figura 3.1):

$$T_{cw} = \begin{bmatrix} R_{cw} & t_{cw} \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (3.6)$$

La segunda transformación es una proyección de perspectiva donde:

$$x_i = f \frac{x_c}{z_c} \quad y_i = f \frac{y_c}{z_c}, \quad (3.7)$$

siendo x_i e y_i las coordenadas del punto en el plano de la imagen. Para representar esta transformación proyectiva utilizamos la notación: $p_i = \pi(\dot{p}_c)$. Como resultado obtenemos que la transformación de un punto en el mundo al plano de la cámara es:

$$p_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \pi(\dot{p}_c) = \pi(T_{cw} \cdot \dot{p}_w) \quad (3.8)$$

La composición de transformaciones nos permite pasar de unos sistemas de referencia a otros:

$$\dot{p}_{c_2} = T_{c_2c_1} T_{c_1w} \cdot \dot{p}_w. \quad (3.9)$$

Conocidas dos transformaciones como las que se muestran en la figura 3.2, podemos calcular la matriz que define la transformación entre la cámara C_2 y el mundo mediante una composición de transformaciones:

$$T_{c_2w} = T_{c_2c_1} T_{c_1w}. \quad (3.10)$$

La inversa de una matriz de transformación define una transformación inversa de manera que :

$$T_{c_1c_2} = T_{c_2c_1}^{-1} = \begin{bmatrix} R_{c_2c_1} & t_{c_2c_1} \\ 0_{1 \times 3} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R_{c_2c_1}^{-1} & -R_{c_2c_1}^{-1} \cdot t_{c_2c_1} \\ 0_{1 \times 3} & 1 \end{bmatrix} \quad (3.11)$$

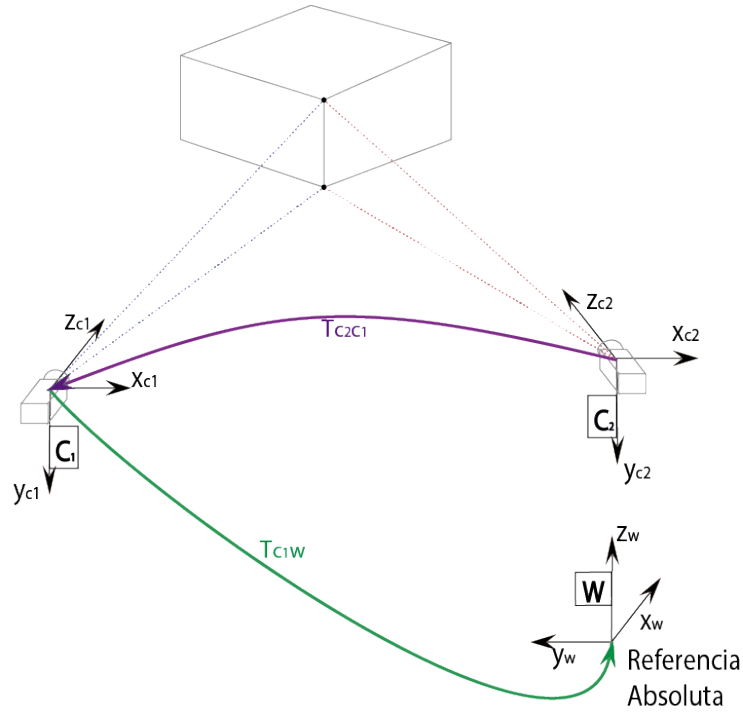


Figura 3.2: Composición de transformaciones

donde

$$R_{c_2c_1}^{-1} = R_{c_2c_1}^T \quad (3.12)$$

Utilizando estas técnicas de geometría podemos, dado un mapa de puntos tridimensional y la posición de una cámara determinar que puntos se proyectarían en el plano de imagen de la cámara y por lo tanto qué ve.

Este modelo proyectivo nos permite estimar la profundidad de un punto a partir de múltiples vistas utilizando triangulación. En el capítulo 4 se detalla el método que hemos utilizado.

3.2. Redes Neuronales Artificiales

Este es el algoritmo de aprendizaje automático seleccionado para resolver el problema de estimación de profundidad a partir de una sola vista, *single-view*. Posee una estructura capaz de representar relaciones complejas de los datos, siempre que disponga de los parámetros y datos suficientes. Si bien son conocidas desde hace décadas, recientemente han experimentado un resurgimiento debido a los últimos avances de la computación que han logrado que el entrenamiento de las redes se pueda realizar en tiempos aceptables,

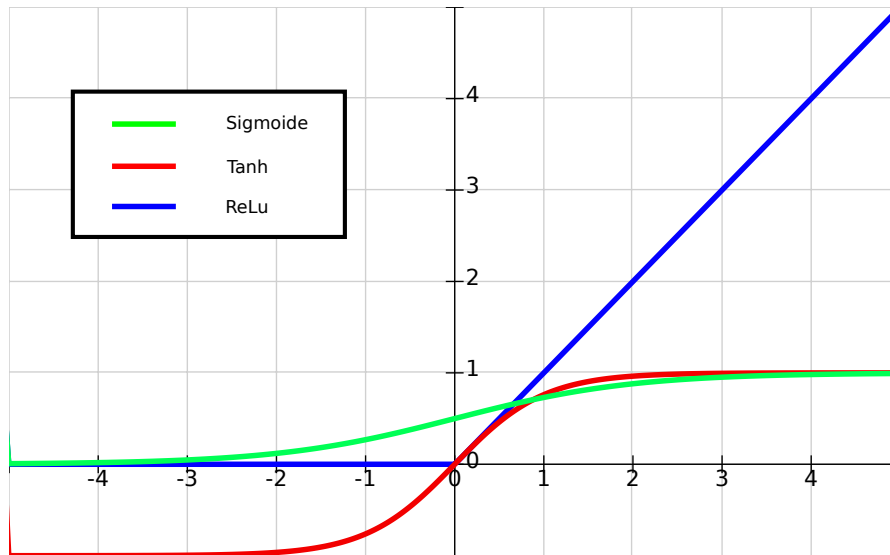


Figura 3.3: Tres funciones de activación de entre las más comunes. *En verde* la función Sigmoido $f(z) = \frac{1}{1+e^{-z}}$, *en rojo* la tangente hiperbólica $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ y *en azul* la función ReLu (*Rectified Linear Unit*) $f(z) = \max(0, z)$.

a la vez que la adquisición de las grandes cantidades de datos que requieren las redes neuronales ha sido cada vez más fácil debido al progreso de las TIC.

3.2.1. Modelo de una RNA

La unidad básica de las redes neuronales es la célula (“*cell*”) o unidad, que trata de simular una neurona biológica al tomar información de entrada de varias fuentes $X = (x_1, x_2, \dots, x_n)^\top$ y producir una salida que representa el procesamiento que ha hecho la neurona a partir de la información de la entrada.

Matemáticamente, la activación de una célula a partir de la información de entrada es un producto escalar (más un término de sesgo o *bias* b),

$$h_{W,b}(X) = f(W^\top X) = f\left(\sum_{i=1}^n w_i x_i + b\right), \quad (3.13)$$

donde $W = (w_1, w_2, \dots, w_n)^\top$ son los pesos propios de la célula y $f(z)$ es la función de activación. Existen diferentes funciones de activación. Algunas de las más utilizadas se muestran en la Figura 3.3. Con la estructura de una neurona definida, puede observarse que se trata en realidad de un modelo lineal al que le es aplicado una función de activación, generalmente no lineal. Por ejemplo, en el caso de la función sigmoideal convierte el modelo en una regresión logística, uno de los algoritmos básicos de aprendizaje automático. Se trata de un modelo que es capaz únicamente de representar relaciones

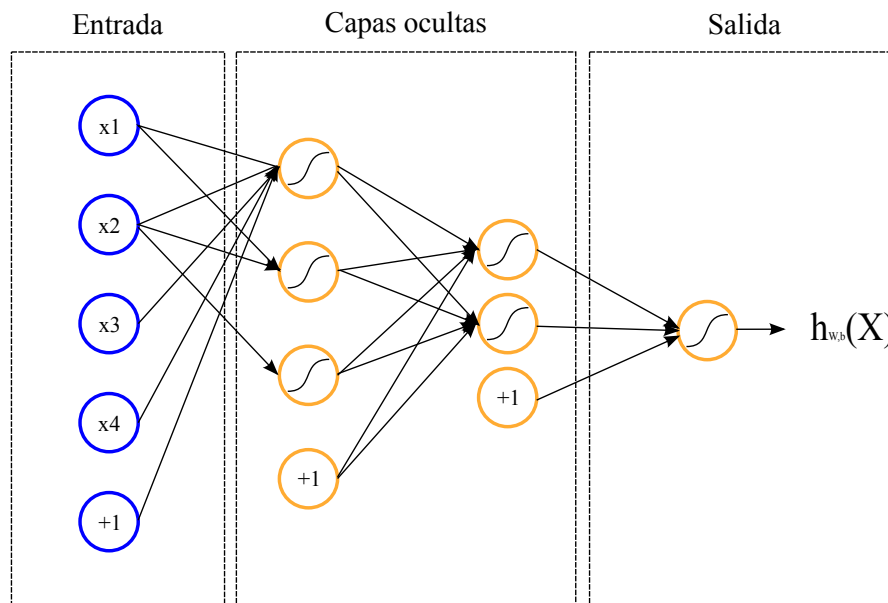


Figura 3.4: Esquema de una red neuronal con dos capas ocultas de tres y dos neuronas respectivamente. Las neuronas con etiqueta “+1” representan el sesgo o bias.

relativamente simples. El poder de las redes neuronales aparece al agrupar un conjunto de neuronas para que trabajen interaccionando unas con otras, de un modo similar a lo que hacen nuestras propias neuronas. Normalmente esta agrupación se lleva a cabo mediante una organización en capas. De forma resumida, todas las neuronas de una capa procesan la misma información de forma paralela. Las neuronas de la capa siguiente reciben como entrada la salida, ya procesada, de la capa anterior. A todas las capas distintas de la entrada y la salida se las llama capas ocultas ya que desde un punto de vista de caja negra, dichas capas procesan pasos internos del algoritmo para la obtención de la predicción a la salida. Un ejemplo ilustrativo de el esquema de una red neuronal se puede ver en la Figura 3.4.

3.2.2. Estructura de las capas de una RNA

En esta sección pretendemos distinguir distintos tipos de capas según según la forma de sus conexiones, entre las cuales destacamos las más relevantes que se han utilizado en este trabajo.

Completamente conectada: (FC- *Fully-Connected*) En esta capa todas las neuronas de la capa actual (capa de la derecha en la Figura 3.5a) están conectadas con todas las neuronas de la capa anterior (capa de la izquierda en la figura). Esto provoca que sea una capa genérica, dado que no está condicionada ni restringida ninguna relación entre las

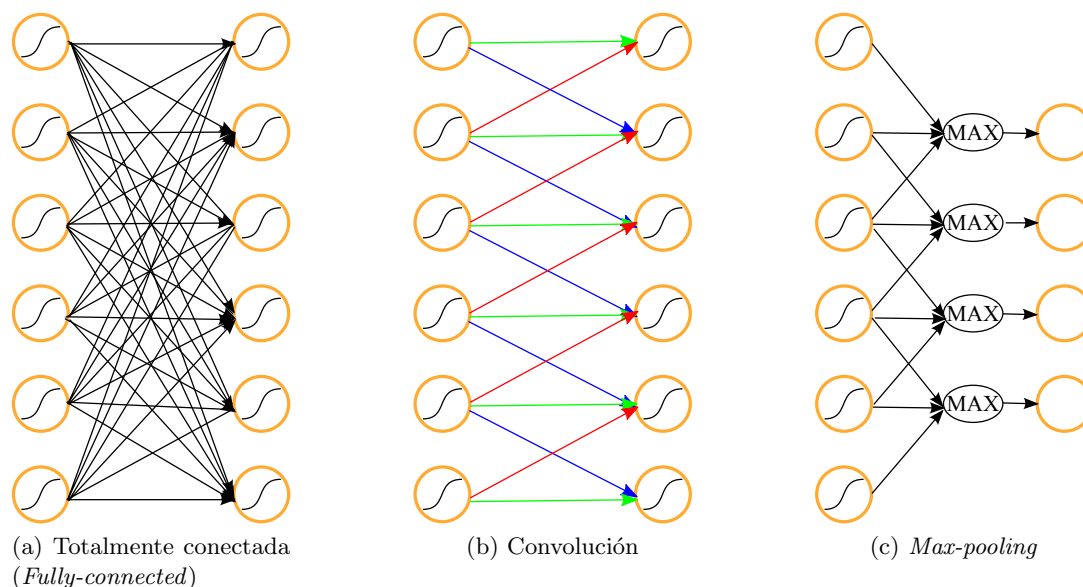


Figura 3.5: Configuración de las conexiones para los tipos de capas más relevantes en una RNA.

neuronas. Este modelo de capa es idóneo cuando se desconoce la estructura del problema, dado que permiten capturar cualquier tipo de relación que haya en los datos. Sin embargo, es el modelo de capa más caro computacionalmente. Requiere un total de $N \cdot (M + 1)$ conexiones, siendo N el número de neuronas de la capa actual y M el número de neuronas de la capa anterior. Por este coste computacional no puede aplicarse este modelo a datos con un elevado número de dimensiones. Por ejemplo, en el problema de estimación de profundidad, si nuestro objetivo es estimarla a partir de una imagen RGB de 320×240 y queremos que nuestra estimación tenga la mitad de tamaño 160×120 utilizando únicamente una capa completamente conectada requeriríamos un total de $(160 \cdot 120) \cdot (320 \cdot 240 + 1) = 1474579200$ parámetros de la capa, que si trabajamos con precisión simple (32 bits) equivalen a 16,48GiB necesarios únicamente para los parámetros de la capa.

Convoluciones (localmente conectada): Cuando las capas completamente conectadas resultan muy caras, y el problema se presta a ello, se suele optar por capas localmente conectadas. En estas, cada neurona de la capa actual se conecta únicamente a un subconjunto de neuronas de la capa anterior. Es importante resaltar que este subconjunto no se aprende, se fija antes del entrenamiento. En concreto, una de las capas localmente conectadas más extendida es la capa convolucional. La capa convolucional tiene como característica que todas las neuronas de la capa actual se conectan con el mismo número de neuronas de la capa anterior (normalmente, los diferentes subconjuntos tienen una relación espacial) y comparten los pesos W entre las neuronas de la misma capa. En la

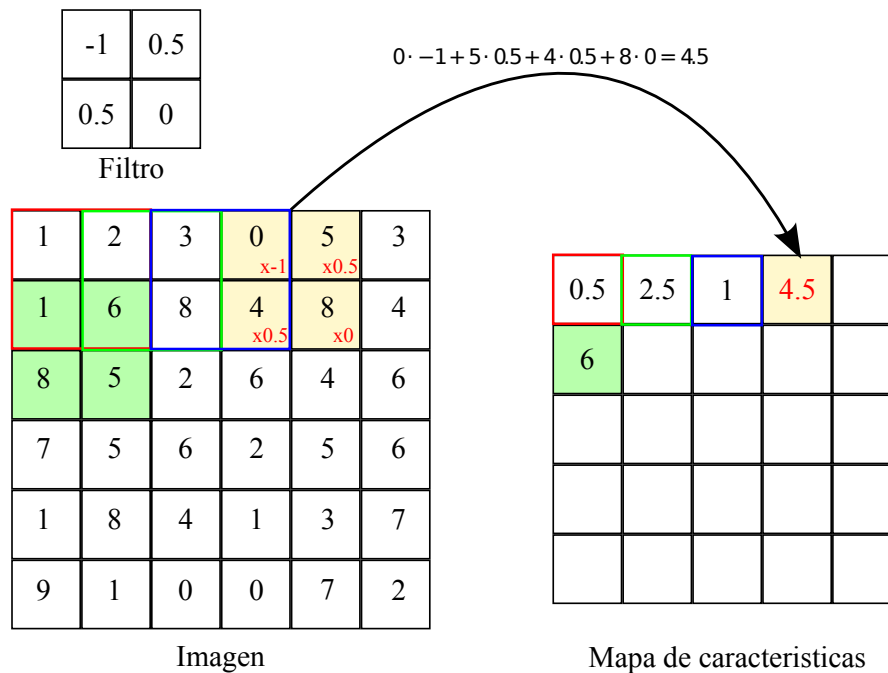


Figura 3.6: Esquema de funcionamiento de una capa convolucional en imágenes.

Figura 3.5b se muestra un ejemplo de una capa convolucional, en el ejemplo las flechas del mismo color representan los mismos pesos w_i .

Pooling (localmente conectada): Otro ejemplo de capa localmente conectada es la capa *pooling*. En este caso cada neurona de la capa actual también está conectada a un subconjunto de neuronas de la capa anterior, con propiedades parejas a la capa convolucional. Sin embargo, en este caso los pesos W son sustituidos por una función de reducción de los datos (e.g. las funciones más comunes son la media y el máximo). En la Figura 3.5c se muestra un ejemplo de un *max-pooling* llevada a cabo con la función máximo. Este modelo se usa entre otros motivos para reducir la dimensionalidad del problema.

3.2.3. Redes Neuronales 2D con imágenes

Dado que nuestro objetivo es trabajar con imágenes es necesario introducir como se aplican algunas de las capas mencionadas en la sección anterior a datos 2D como las imágenes. En concreto los modelos de capa localmente conectados pues son los que se ven afectados.

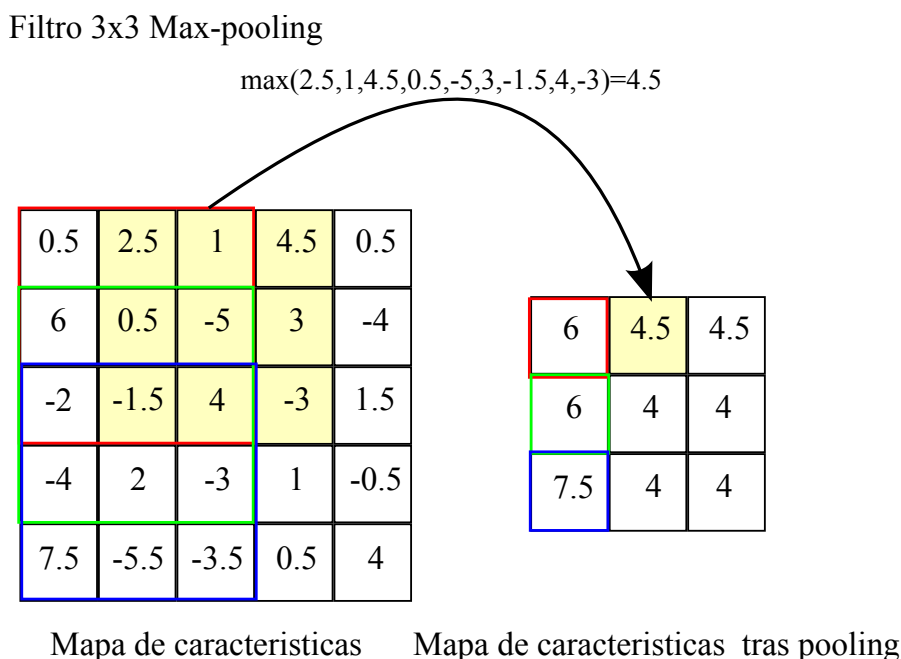


Figura 3.7: Esquema de funcionamiento de una capa *max-pooling* en imágenes.

Convolución: El operador convolución aplica, dadas dos matrices (la imagen y el filtro), las cuales pueden ser de tamaños distintos, el producto escalar de Frobenius (sumatorio del producto elemento a elemento) entre la matriz de menor tamaño (el filtro) y una submatriz de la de mayor tamaño (imagen), de forma que el orden de esta submatriz sea el mismo que el del filtro. Teóricamente se ha de invertir el filtro para que se trate de una auténtica convolución, pero por claridad especialmente en la visualización, esto no se realiza en esta área en particular. De este modo se obtiene un escalar para cada combinación de filtro y ventana de la imagen. Al igual que en el operador de convolución continuo, se produce un desplazamiento relativo entre los dos elementos: el filtro se mueve a lo largo y ancho de la imagen (se está operando en 2D), produciendo para cada posición un escalar distinto ya que la ventana de la imagen cambia. Dichos escalares se almacenan en una nueva matriz donde conservan el orden espacial en el que se han calculado, de modo similar a la convolución unidimensional pero trabajando en las dos dimensiones del problema. Se puede ver un esquema ilustrativo de la operación convolución en la Figura 3.6.

En realidad, las imágenes RGB no son tensores de dos dimensiones, sino de tres (altura, anchura, canal). Por ello, el procesamiento explicado anteriormente es en realidad algo más complicado. Lo que se hace en realidad es el mismo proceso, pero aplicando un filtro de a su vez tres dimensiones. De este modo, el cálculo es exactamente el mismo sólo que con el triple de elementos. Por tanto, dada una imagen y un filtro, al aplicar la conexión convolucional se obtiene una matriz donde se almacena espacialmente los distintos escalares obtenidos por el “inner product” (Frobenius) del filtro por la ventana

local de la imagen. Es decir, se obtiene una nueva imagen 2D, solo que en vez de colores contiene características obtenidas espacialmente por el filtro. Señalar que dichas características son todas obtenidas por el mismo filtro, pero moviéndolo a lo largo y ancho de la imagen, de modo similar a un escáner. Para aumentar el poder de representación de la capa, se pueden emplear tantos filtros distintos como se quiera: se almacenan los resultados de cada filtro a lo largo de una tercera dimensión, obteniendo de nuevo un tensor de tres dimensiones. Cada una de las matrices pertenecientes al tensor se corresponde al “escaneo”, mapa de características, obtenido a partir de la imagen y del filtro correspondiente. Además, al tratarse de nuevo de un tensor de tres dimensiones, implica que es posible volver a aplicar una capa convolucional, permitiendo construir redes convolucionales profundas.

Pooling: Se aplica de forma muy similar a la convolución. Esta operación se suele utilizar únicamente a mapas de características. Para simplificar la explicación vamos asumir que la operación es un *max-pooling*, ya que es la función que más se utiliza. Dada una matriz (mapa de características) y el tamaño de una ventana o filtro el *pooling* consiste en aplicar la función máximo a todos los elementos de una submatriz del tamaño del filtro en el mapa de características. De esta forma se obtiene un escalar para cada combinación filtro y ventana del mapa de características. Se produce un desplazamiento relativo entre los dos elementos: el filtro se mueve a lo largo y ancho de la imagen (dado que opera en 2D), produciendo para cada posición un escalar distinto. Dichos escalares se almacenan en una nueva matriz donde conservan el orden espacial en el que se han calculado. Se puede ver un esquema ilustrativo de un *max-pooling* en la Figura 3.7.

Cuando el *pooling* se aplica a tensores 3D de mapas de características se suele aplicar a cada uno de los mapas por separado, produciendo otro tensor con los mismos “canales” pero de menor altura y anchura al que se le puede aplicar otra vez el operador convolución. Esta operación tiene dos propiedades deseadas. Primero, reduce la dimensionalidad de las salidas de las convoluciones que producen muchos mapas de características de gran tamaño. Segundo, el *max-pooling* espacial resume la activación de características vecinas, que conduce a una invarianza por traslación (local), es decir elimina ruido de alta frecuencia.

La combinación de estas capas repetidamente permite construir redes neuronales convolucionales profundas (CNN). Este tipo de redes neuronales profundas ha demostrado excelentes resultados en diferentes problemas de visión por computador, en reconocimiento de actividades [22, 23], descripción de imágenes utilizando lenguaje natural [23, 24] y reconocimiento de objetos [25], entre otras.

4. Fusión de profundidad

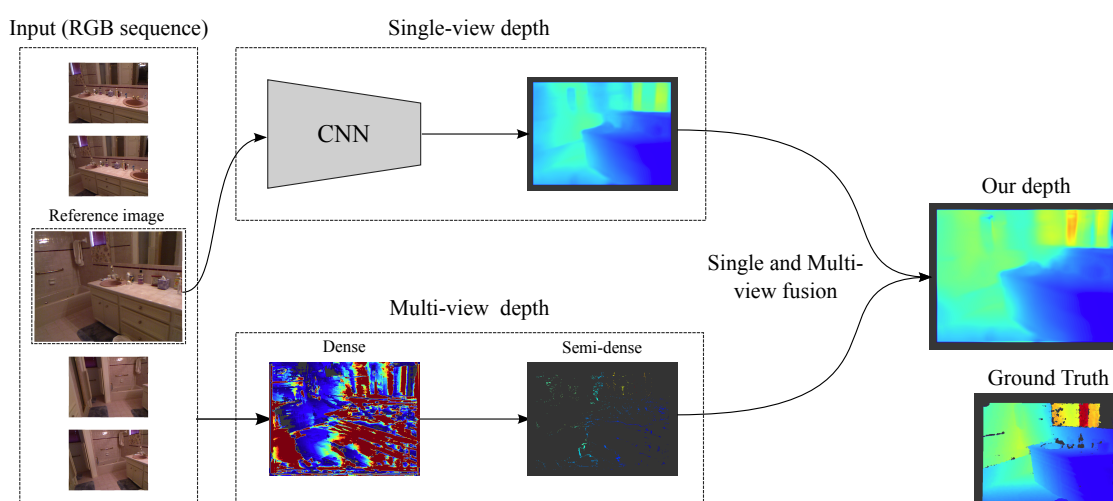


Figura 4.1: Visión general de nuestra propuesta. Nuestra entrada son un conjunto de vistas superpuestas tomadas por una cámara monocular. Estimamos un mapa de profundidad basado en aprendizaje para una sola vista y un mapa de profundidad basado en geometría a partir de varias imágenes, fusionamos ambas fuentes de información para obtener una profundidad que mejora los dos métodos por separado. El color de las imágenes de profundidad ha sido normalizado para una mejor comparación. La imagen de profundidad se representa en una escala continua de colores de azul oscuro para los puntos más cercanos, verde para los puntos intermedios y rojo para los puntos lejanos. Esta figura se ve mejor a color.

Las técnicas basadas en múltiples vistas del estado del arte tienen una fuerte dependencia en secuencias de alto paralaje y escenas con mucha textura. Solo un reducido conjunto de píxeles que cumplen ambas restricciones obtienen una predicción con poco error, la mayoría de puntos tienen un error mayor y no correlado. Por el contrario, los métodos a partir de una única vista basados en redes CNN logran errores razonables en todas las partes de la imagen, pero están correlados de forma local. Nuestra propuesta pretende explotar las mejores propiedades de estos dos métodos. Concretamente, producimos un mapa de profundidad crudo utilizando una red neuronal profunda y fusionamos su estructura con los resultados semi-densos de un método *multi-view* (Figura 4.1).

	Alto gradiente	Bajo gradiente
<i>Multi-View</i>	0.25	0.79
<i>Single-View</i>	0.60	0.18

Tabla 4.1: Error mediano, en metros, de los algoritmos basados en múltiples o una única vista, y en píxeles de alto y bajo gradiente.

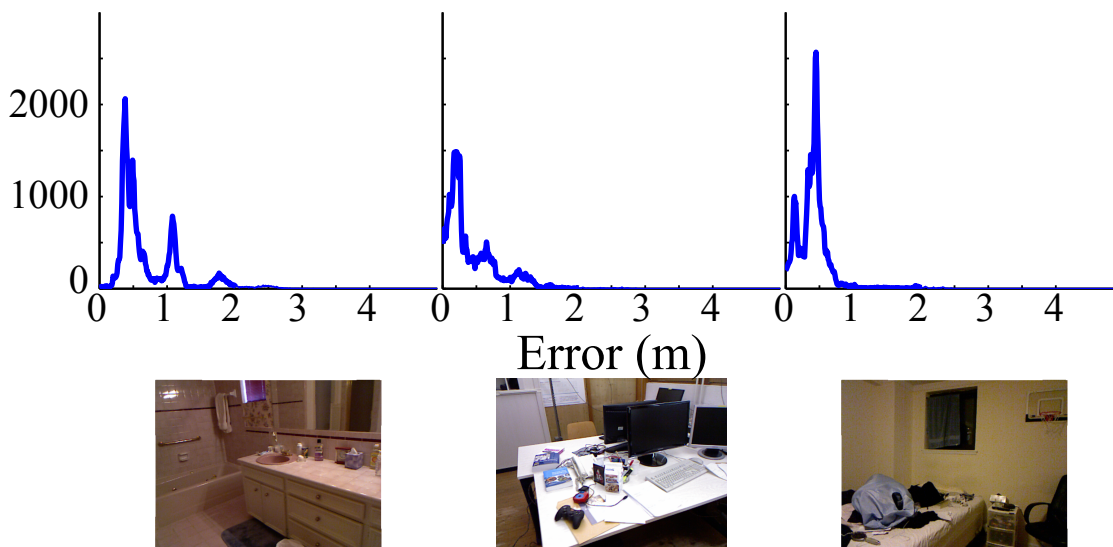


Figura 4.2: Histograma del error de la profundidad estimada, en metros, por el método basado en una vista. Notese las múltiples tendencias del error cada una correspondiendo a una estructura local de la imagen.

Antes de entrar en los aspectos técnicos, motivaremos nuestra propuesta con algunos resultados ilustrativos. La Tabla 4.1 muestra el error mediano de profundidad de los píxeles de alto y bajo gradiente para los dos métodos, uno basado en una imagen y otro basado en múltiples vistas, en una secuencia de alto paralaje. En el caso del método *multi-view*, el error en los píxeles de bajo gradiente aumenta en un factor de 3. Notese que pasa al contrario en el caso del *single-view*: el error en los píxeles de alto gradiente incrementa por un factor de 3.

Además, el error de profundidad estimada por el método *single-view* usualmente tiene una estructura que indica la presencia de correlación local. Por ejemplo, en la Figura 4.2 se muestra el histograma del error de profundidad para el método *single-view* en tres escenas diferentes (dos del conjunto de datos publico NYU y una del TUM). Se puede apreciar que la distribución del error esta agrupada en diferentes tendencias, cada una de ellas correspondiendo a un segmento de la imagen. Este efecto es causado por el uso de las características de alto nivel de las ultimas capas de la red neuronal convolucional (CNN), se extienden sobre docenas de píxeles en la imagen original y por lo tanto sobre zonas de textura homogénea. La diferente naturaleza de los error puede

ser explotada para mejorar ambos métodos individualmente. Sin embargo, la fusión no puede ser implementada basado an un modelo global, dado que requiere deformaciones no rígidas.

En las siguientes secciones detallamos los métodos *multi-view* y *single-view* que usamos en este trabajo así como nuestro algoritmo de fusión.

4.1. Profundidad a partir de múltiples vistas

Para la estimación de profundidad basada en múltiples vistas vistas hemos adoptado una aproximación directa [14], que nos permite estimar un mapa de profundidad denso o semi-denso al contrario que los métodos basados en puntos salientes. Para estimar la profundidad de un *keyframe*¹ \mathcal{I}_k primero seleccionamos un conjunto n de imágenes solapadas $\{\mathcal{I}_1, \dots, \mathcal{I}_o, \dots, \mathcal{I}_n\}$ de una secuencia monocular. Después de eso, cada píxel x_l^k de la imagen de referencia \mathcal{I}_k es mapeado a una profundidad inversa ρ y re-proyectado a todos las imágenes superpuestas \mathcal{I}_o . Utilizando el modelo proyectivo de la sección 3.1.

$$x_l^o = T_{ko}(x_l^k, \rho_l) = \pi \left(K R_{ko}^\top \left(\pi \left(\frac{K^{-1} x_l^k}{\|K^{-1} x_l^k\|} \right) - t_{ko} \right) \right), \quad (4.1)$$

donde $\pi(X)$ representa la deshomogenización de un vector X y T_{ko}, R_{ko} y t_{ko} son respectivamente la matriz de transformación relativa, la rotación y la traslación entre el *keyframe* \mathcal{I}_k y cada uno de las imágenes superpuestas \mathcal{I}_o . K es la matriz de calibración interna de la cámara.

Definimos el error fotométrico total $C(\rho)$ como la suma de cada uno de los errores fotométricos ϵ_l entre cada píxel (o cada píxel de alto gradiente en caso de querer un mapa semi-denso) x_l^k en la imagen de referencia \mathcal{I}_k y su correspondiente x_l^o en cada una de las imágenes superpuestas \mathcal{I}_o para una profundidad inversa hipotética ρ_l ,

$$C(\rho) = \frac{1}{n} \sum_{o=1, o \neq k}^n \sum_{l=1}^t \epsilon_l(\mathcal{I}_k, \mathcal{I}_o, x_l^k, \rho_l). \quad (4.2)$$

El error $\epsilon_l(\mathcal{I}_k, \mathcal{I}_o, x_l^k, \rho_l)$ para cada píxel x_l^k es la diferencia entre los valores fotométricos del píxel y sus correspondientes

$$\epsilon_l(\mathcal{I}_k, \mathcal{I}_o, x_l^k, \rho_l) = \mathcal{I}_k(x_l^k) - \mathcal{I}_o(x_l^o). \quad (4.3)$$

La profundidad estimada para cada píxel $\hat{\rho} = (\hat{\rho}_1 \dots \hat{\rho}_l \dots \hat{\rho}_t)^\top$ es obtenida por la minimización del error fotométrico total $C(\rho)$:

$$\hat{\rho} = \arg \min_{\rho} C(\rho) \quad (4.4)$$

¹*keyframe* - imagen de referencia en la Figura 4.1

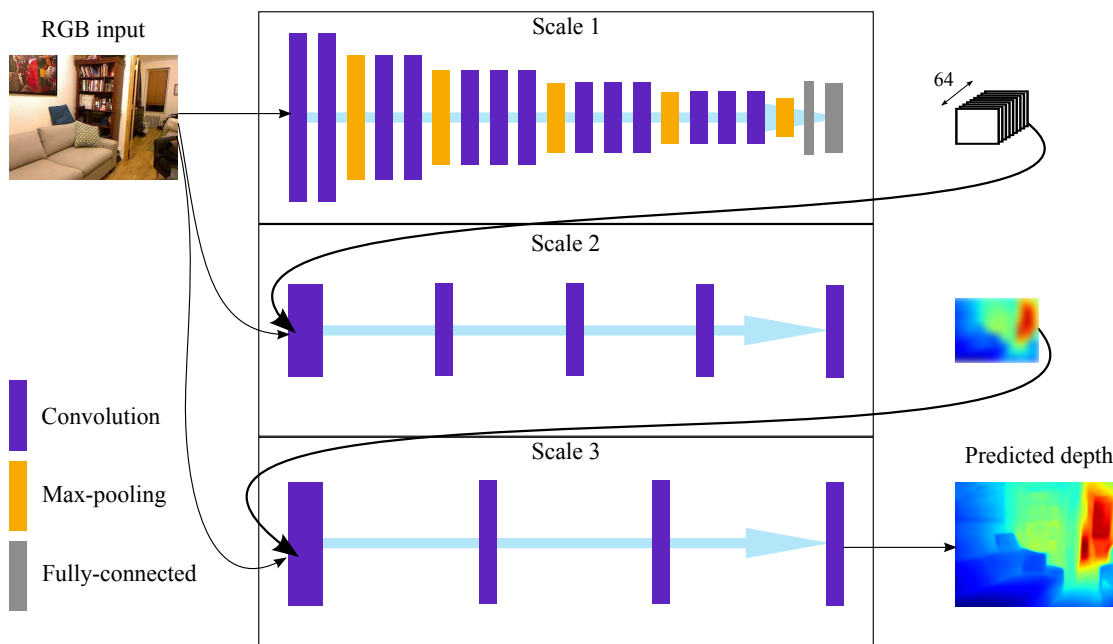


Figura 4.3: Modelo de red neuronal presentado por [19] para estimación de profundidad a partir de una imagen RGB. La imagen de profundidad se representa en una escala continua de colores de azul oscuro para los puntos más cercanos, verde para los puntos intermedios y rojo para los puntos lejanos.

4.2. Profundidad a partir de una única vista

Para la estimación de profundidad a partir de una única imagen utilizamos la red neuronal *convolucional* profunda (*Deep Convolutional Neural Network*) presentada por Eigen *et al.*, [19]. Esta red utiliza tres CNN apiladas para procesar las imágenes en tres diferentes escalas. En este trabajo la entrada de la red es nuestra imagen RGB de referencia, *keyframe*, \mathcal{I}_k . Dado que usamos la estructura de la red y los parámetros aportados por los autores, sin más entrenamiento, la imagen de entrada tiene un tamaño de 320×240 . La salida de la red es la profundidad predicha, la cual denotaremos s . El tamaño de la salida es de 147×109 , que nosotros ampliamos para poder fusionarla con la salida del método *multi-view*.

La CNN de la primera escala extrae características de alto nivel relevantes para estimación de profundidad. Esta CNN produce 64 mapas de características de tamaño 19×14 , que ampliados a 74×55 y junto con la imagen RGB son la entrada de la CNN de la segunda escala. En esta segunda CNN apilada se refinan las características extraídas en la primera con características de intermedias para producir un primer mapa de profundidad de grano grueso de tamaño 74×55 . Este mapa de profundidad es aumentado y junto con la imagen alimenta a la tercera y última CNN apilada que hace un refinamiento local de

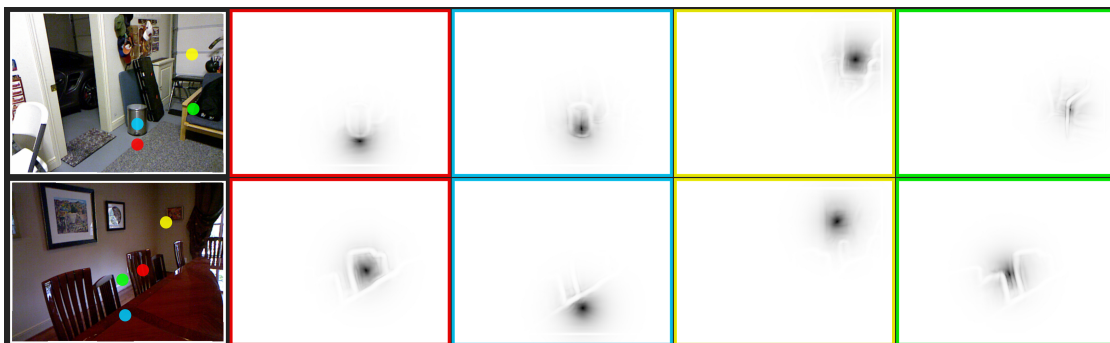


Figura 4.4: Pesos de influencia no normalizados para cada uno de los puntos marcados en la imagen. *Primera columna:* Imagen RGB de entrada con cuatro puntos remarcados en la imagen. *Segunda, tercera, cuarta y quinta columnas:* Muestran la influencia de cada punto remarcado en la imagen RGB. Esta figura se ve mejor en color.

la profundidad. Este último paso es necesario, dado que las *convoluciones* y los *poolings* de los pasos anteriores filtran los detalles de alta frecuencia.

La primera escala se inicializó con dos diferentes redes pre-entrenadas: la *AlexNet* [26] y la VGG de Oxford [27]. Nosotros usamos la versión de la VGG, la más precisa como reportaron los autores. Esta red ha sido entrenada con imágenes de interior del conjunto público de datos *NYUDepth v2* [28]. Dado que ellos utilizaron la partición oficial de los datos para entrenamiento y test, nosotros también. Decidimos trabajar con esta red neuronal porque es el método que mejores profundidades completamente densas consigue a partir de una única imagen. Para ver en detalle el modelo de la red propuesto vease la Figura 4.3. Le recomendamos al lector que para más detalles sobre esta parte de nuestro trabajo consulten el artículo original [19].

4.3. Deformación no rígida

Como ya hemos mencionado, el objetivo de este trabajo es fusionar la salida de cada uno de los métodos previamente descrito, y mantener las mejores propiedades de cada uno de ellos: en el caso del *single-view* una fiable estructura local y una precisa, pero semi-densa, estimación de profundidad del *multi-view*. Denotamos s y m a las estimaciones de profundidad realizadas por los métodos *single-view* y *multi-view* respectivamente. s es predicha como se detalla en la Sección 4.2 y $m = \frac{1}{\rho}$ es la inversa de la profundidad inversa estimada en la Sección 4.1.

La profundidad resultado de la fusión f_{ij} para el píxel (i, j) es calculada como una interpolación pesada de profundidades sobre un conjunto de píxeles en la imagen de

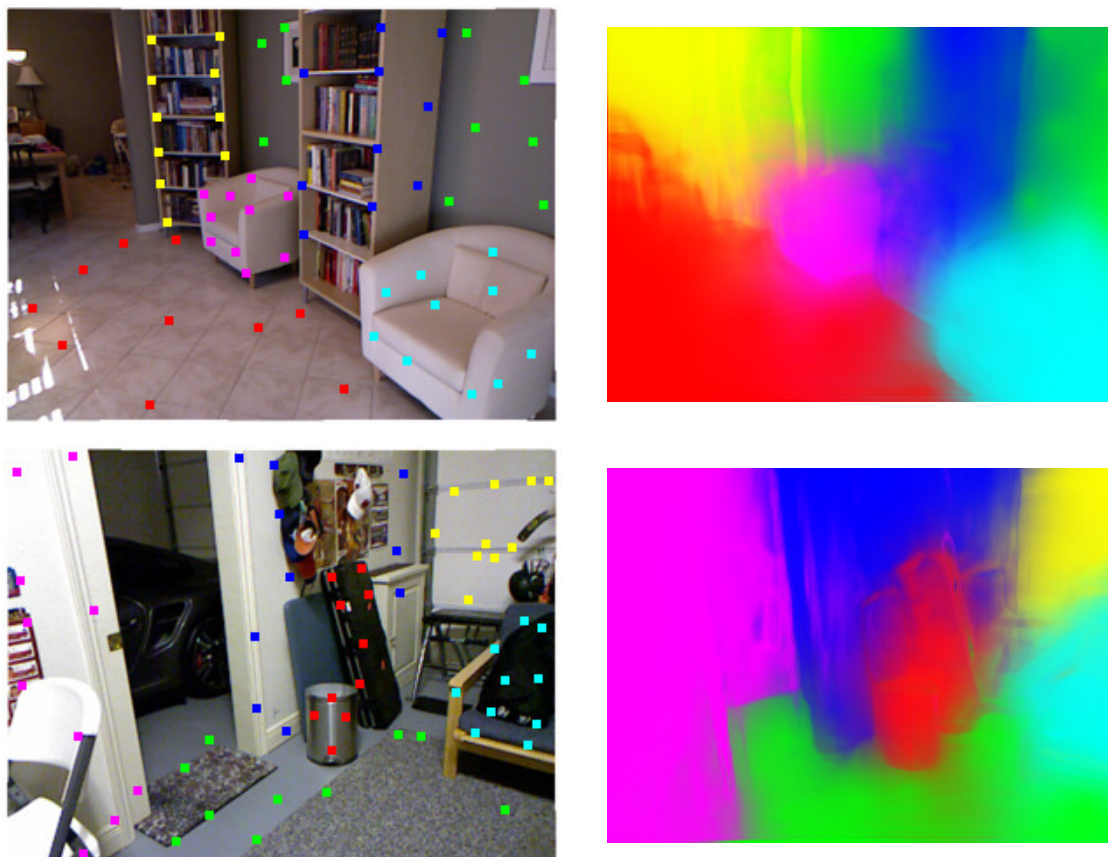


Figura 4.5: Áreas de influencia normalizadas. Notese como se expande alrededor de las estructuras locales dado un conjunto de puntos en Ω . *Primera columna:* Imagen RGB con los puntos Ω etiquetados con diferentes colores. *Segunda columna:* Áreas de influencia calculadas por nuestro método. Esta figura se ve mejor a color.

profundidad estimada a partir de múltiples vistas

$$f_{ij} = \sum_{(u,v) \in \Omega} W_{s_{ij}}^{m_{uv}} (m_{uv} + (s_{ij} - s_{uv})), \quad (4.5)$$

donde Ω es el conjunto de píxeles estimado por el algoritmo *multi-view*. Los pesos de la interpolación $W_{s_{ij}}^{m_{uv}}$ modelan la probabilidad de que cada píxel $(u, v) \in \Omega$ pertenezca a la misma estructura local que el píxel s_{ij} . La interpolación puede ser interpretada de dos maneras. Primero, el gradiente de profundidad $(s_{ij} - s_{uv})$ es añadido a cada profundidad estimada por el método *multi-view* m_{uv} , i.e. creamos un mapa de profundidad para cada m_{uv} con la estructura de s y las ponderamos con los pesos basados en los píxeles. Segundo, modificamos cada profundidad s_{ij} de acuerdo con su la discrepancia pesada entre $(m_{uv} - s_{uv})$.

El ingrediente clave de esta interpolación son los pesos $W_{s_{ij}}^{m_{uv}}$ que necesitan lograr la

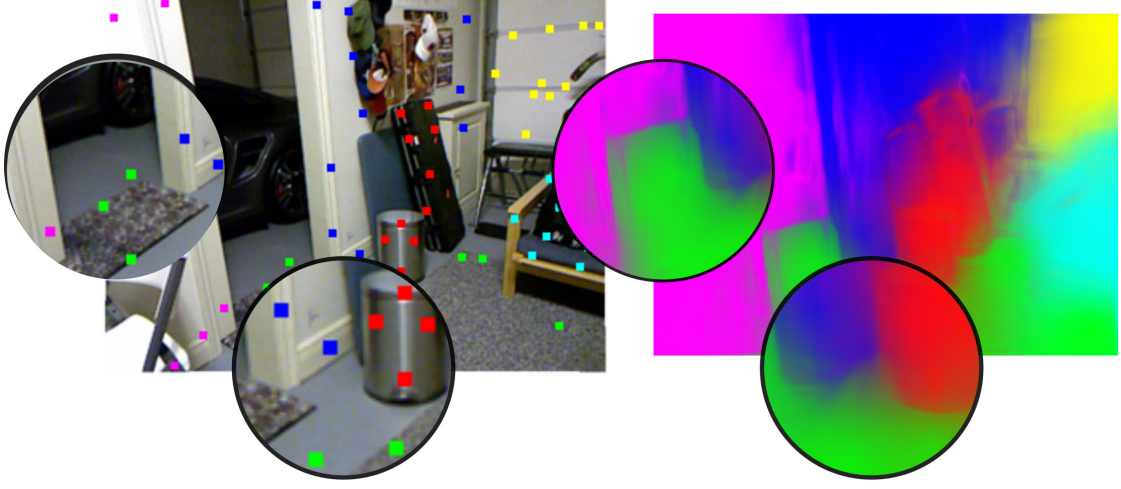


Figura 4.6: Áreas de influencia de forma detallada. Notese como se extiende principalmente en zonas de la misma estructura local. Esta figura se ve mejor a color.

transformación no rígida mencionada anteriormente basada en la estructura local de la imagen. Cada peso es calculado como el producto de cuatro factores. El primer factor

$$\tilde{W}_{1s_{ij}}^{m_{uv}} = e^{-\frac{\sqrt{(i-u)^2+(j-v)^2}}{\sigma_1}}, \quad (4.6)$$

mide simplemente la proximidad basado en la distancia entre los píxeles (ij) y (uv) . El parámetro σ_1 controla el radio de proximidad de cada punto. Los tres factores restantes dependen de la estructura del mapa de profundidad estimado por el método *single-view* s . El segundo factor

$$\tilde{W}_{2s_{ij}}^{m_{uv}} = \frac{1}{|\nabla_x s_{uv} - \nabla_x s_{ij}| + \sigma_2} \cdot \frac{1}{|\nabla_y s_{uv} - \nabla_y s_{ij}| + \sigma_2} \quad (4.7)$$

mide la similaridad de los gradientes de profundidad y asigna mayores pesos a los que son más similares entre ellos. $\nabla_x s_{ij}$ y $\nabla_y s_{ij}$ representan el gradiente de profundidad en las direcciones x e y respectivamente en el píxel (i, j) . σ_2 limita la influencia de un punto para evitar pesos extremadamente altos en puntos con gradiente muy similar o idéntico. Los dos últimos factores fortalecen la influencia entre los puntos que están en el mismo plano y se definen como

$$\tilde{W}_{3s_{vij}}^{m_{uv}} = e^{-|(s_{ij} + \nabla_x s_{ij} \cdot (i-u)) - s_{uv}|} + \sigma_3 \quad (4.8)$$

and

$$\tilde{W}_{4s_{ij}}^{m_{uv}} = e^{-|(s_{ij} + \nabla_y s_{ij} \cdot (j-v)) - s_{uv}|} + \sigma_3, \quad (4.9)$$

donde σ_3 fija un valor mínimo para el peso de cada punto en Ω . Se requiere para evitar el desvanecimiento de los pesos cuando estos son combinados con $\tilde{W}_{1s_{ij}}^{m_{uv}}$ y $\tilde{W}_{2s_{ij}}^{m_{uv}}$.

El producto de estos cuatro factores produce un peso no normalizado para cada píxel en Ω con respecto a todos los píxeles de la imagen

$$\tilde{W}_{s_{ij}}^{m_{uv}} = \tilde{W}_{1s_{ij}}^{m_{uv}} \cdot \tilde{W}_{2s_{ij}}^{m_{uv}} \cdot \tilde{W}_{3s_{ij}}^{m_{uv}} \cdot \tilde{W}_{4s_{ij}}^{m_{uv}}, \quad (4.10)$$

y representa su área de influencia. Los parámetros σ_1 , σ_2 y σ_3 dan forma al área de influencia y tienen que ser seleccionados para balancear la proximidad, los gradientes y planos y para evitar discontinuidades en el resultado final de la fusión. Esta selección se hizo empíricamente en un conjunto pequeño de tres imágenes. Los valores para los parámetros son 15, 0.1 y $1e - 3$ respectivamente. Estos valores se fijaron para todos los experimentos, sugiriendo que no requieren ser estimados para cada imagen.

La Figura 4.4 muestra este área para algunos puntos en dos imágenes diferentes. Notese como la influencia se expande alrededor del punto pero se mantiene dentro de la misma estructura local. Una vez todos los factores han sido calculados para todos los píxeles en Ω , normalizamos los pesos para cada píxel del *single-view* y así todos los pesos sobre un píxel (i, j) suman 1.

$$W_{s_{ij}}^{m_{uv}} = \frac{\tilde{W}_{s_{ij}}^{m_{uv}} - \min_{(g,h) \in \Omega} \tilde{W}_{s_{ij}}^{m_{gh}}}{\sum_{(p,k) \in \Omega} \tilde{W}_{s_{ij}}^{m_{pk}} - \min_{(g,h) \in \Omega} \tilde{W}_{s_{ij}}^{m_{gh}}} \quad (4.11)$$

Los pesos normalizados expanden la influencia local a toda la imagen (ver la Figura 4.5 y la Figura 4.6 para verlo con más detalle). Notese como la influencia se expande a lo largo de los planos incluso si los puntos en Ω no alcanza el final del plano y es cortada cuando la estructura local cambia. Una vez todos los pesos han sido calculados y normalizados, la estimación profundidad fusionada f , para cada punto es la combinación de todos los puntos seleccionados en Ω , como se muestra en la Ecuación 4.5.

4.4. Selección de puntos de bajo error

Hasta ahora, hemos asumido que todos los puntos en la estimación semi-densa de profundidad del método *multi-view* tienen bajo error. Esto se logra fácilmente en secuencias de alto paralaje usando estimadores robustos – funciones de coste robustas o esquemas RANSAC. Sin embargo, es problemático en secuencias con geometría de bajo paralaje en las que también nos centramos en este trabajo. En este caso, las profundidades estimadas a partir de múltiples vistas pueden contener grandes errores que pueden propagarse a nuestra fusión y que por lo tanto necesitamos filtrar. Inesperadamente, seleccionar los píxeles de alto gradiente no es suficientemente robusto para eliminar los puntos con gran error en profundidad y por ello hemos desarrollado un algoritmo en dos pasos que tiene en cuenta la información fotométrica y geométrica en el primer paso y el mapa estimado por la red neuronal, el método *single-view*, en el segundo paso.

En el primer paso pre-selecciona un porcentaje fijo de los mejores candidatos –el mejor 25 % en nuestros experimentos– basado en el producto de un coeficiente fotométrico

y otro geométrico. Por un lado, el criterio fotométrico puntúa la calidad de las correspondencias utilizando información de la imagen. Aplicamos una versión modificada del *mejor ratio al segundo vecino*. Calculamos las dos mejores correspondencias (de acuerdo con la Ecuación 4.3), y calculamos este coeficiente en función de el ratio entre la distancia de los dos mínimos (un ratio pequeño quiere decir un emparejamiento fiable) y el gradiente de la función a lo largo de la epipolar (i.e. si la función presenta una forma clara de V alrededor del mínimo quiere decir que el emparejamiento es espacialmente fiable). Por otro lado, el criterio geométrico simplemente proyecta la fiabilidad de la estimación de profundidad en la imagen, resultando en puntuaciones bajas para los puntos de bajo paralaje.

En la segunda fase usamos la estructura de la reconstrucción predicha por el método *single-view* y aplicamos RANSAC para estimar una transformación lineal libre de espurios entre las predicciones de los métodos *multi-view* y *single-view* utilizando únicamente los puntos pre-filtrados en la primera fase. Esto consigue reducir más el número de predicciones espurias del algoritmo *multi-view*. El resultado es un pequeño conjunto de puntos de bajo error que utilizamos para la deformación no rígida descrita en la sección anterior. Como hemos dicho antes, este algoritmo presenta mejores resultados en nuestros experimentos que un test de compatibilidad geométrica, especialmente en las secuencias de bajo paralaje del conjunto de datos de NYU.

5. Resultados experimentales

5.1. Evaluación de la fusión

En esta sección evaluamos el algoritmo y comparamos su rendimiento contra dos métodos del estado del arte: método directo a partir de múltiples vistas usando regularización TV (implementado siguiendo los trabajos [4, 32]) y el método *single-view* para estimar profundidad utilizando la red neuronal de [19]. Hemos seleccionado dos conjuntos de datos con diferentes propiedades. El primero es el NYUv2 Depth Dataset [28], un conjunto de datos general pensado para llevar a cabo evaluaciones de segmentación semántica de imágenes y por lo cual contiene secuencias de bajo paralaaje y escenas de poca textura. Hemos analizado los resultados en seis secuencias del conjunto de test oficial (i.e. la red neuronal para estimación de profundidad a partir de una imagen no ha sido entrenada en estas secuencias) seleccionadas para incluir diferentes tipos de habitaciones. El segundo conjunto de datos es el TUM RGB-D SLAM Dataset [31], un conjunto de datos orientado a SLAM¹ y que presenta un sesgo que beneficia a los métodos basados en múltiples vistas. En este caso, hemos evaluado en dos secuencias seleccionadas de forma aleatoria.

Hemos corrido nuestro algoritmo en una versión reducida de las imágenes de tamaño 320×240 , dado que es el tamaño de entrada de la red neuronal, que utilizamos como método de *single-view*, dada por los autores. También corremos el algoritmo *multi-view* de estimación de profundidad con imágenes de este tamaño, y aumentamos los resultados de nuestra fusión a 640×480 para compararnos con el *ground truth*, canal de profundidad tomado por la cámara Kinect.

Nuestro objetivo es evaluar la precisión de las estimaciones de profundidad, por lo que asumimos que las poses de la cámara son conocidas para la estimación del método *multi-view*. En el conjunto de datos TUM RGB-D SLAM Dataset [31] usamos el *ground truth* de la pose de las cámara dado en los datos. En el caso de las secuencias del NYUv2 Depth Dataset estimamos las poses utilizando la *RGB-D Dense Visual Odometry* de Gutierrez-Gomez *et al.* [33]. Estas poses se mantienen fijas y son usadas para la construcción de mapas a partir de múltiples vistas. Como ya hemos mencionado los parámetros de la

¹SLAM - Simultaneous Localization and Mapping

		MEAN ERROR (m)			GAIN w.r.t TV	GAIN w.r.t [19]
Sequence		TV	Eigen[19]	Ours(auto)	Ours(auto)	Ours(auto)
NYUDepth v2	bathroom_18	1.612	0.723	0.627	61.12 %	13.33 %
	bedroom_13	0.918	0.421	0.342	62.73 %	18.80 %
	dining_32	0.991	0.342	0.399	59.86 %	-16.39 %
	kitchen_32	0.941	0.583	0.581	38.29 %	0.37 %
	living_25	0.828	0.440	0.394	52.47 %	10.55 %
	living_30a	1.472	0.572	0.487	66.91 %	14.87 %
TUM	fr1_desk	0.605	0.437	0.358	40.82 %	18.07 %
	fr1_room	0.613	0.216	0.202	67.04 %	6.48 %

Tabla 5.1: Media del error para la regularización TV de la estimación *multi-view*, la estimación de profundidad *single-view* de [19] y nuestra fusión. Las dos ultimas columnas muestran nuestra mejora con respecto a los dos métodos individuales respectivamente.

		MEAN ERROR (m)		GAIN w.r.t TV	GAIN w.r.t [19]
Sequences		Ours(auto)	Ours(man)	Ours(man)	Ours(man)
NYUDepth v2	bathroom_18	0.627	0.232	85.61 %	67.93 %
	bedroom_13	0.342	0.167	81.79 %	60.33 %
	dining_32	0.399	0.323	67.39 %	5.43 %
	kitchen_32	0.581	0.424	54.91 %	27.20 %
	living_25	0.394	0.289	65.12 %	34.36 %
	living_30a	0.487	0.388	73.62 %	32.15 %
TUM	fr1_desk	0.358	0.229	62.14 %	47.47 %
	fr1_room	0.202	0.131	78.63 %	39.35 %

Tabla 5.2: Error medio de la estimación de nuestra fusión con selección de puntos automática y de nuestra fusión con selección de puntos manual. Las dos últimas columnas muestran la mejora de nuestra fusión con selección de puntos manual con respecto a la regularización TV de la estimación *multi-view* y la estimación de profundidad *single-view* de [19] respectivamente.

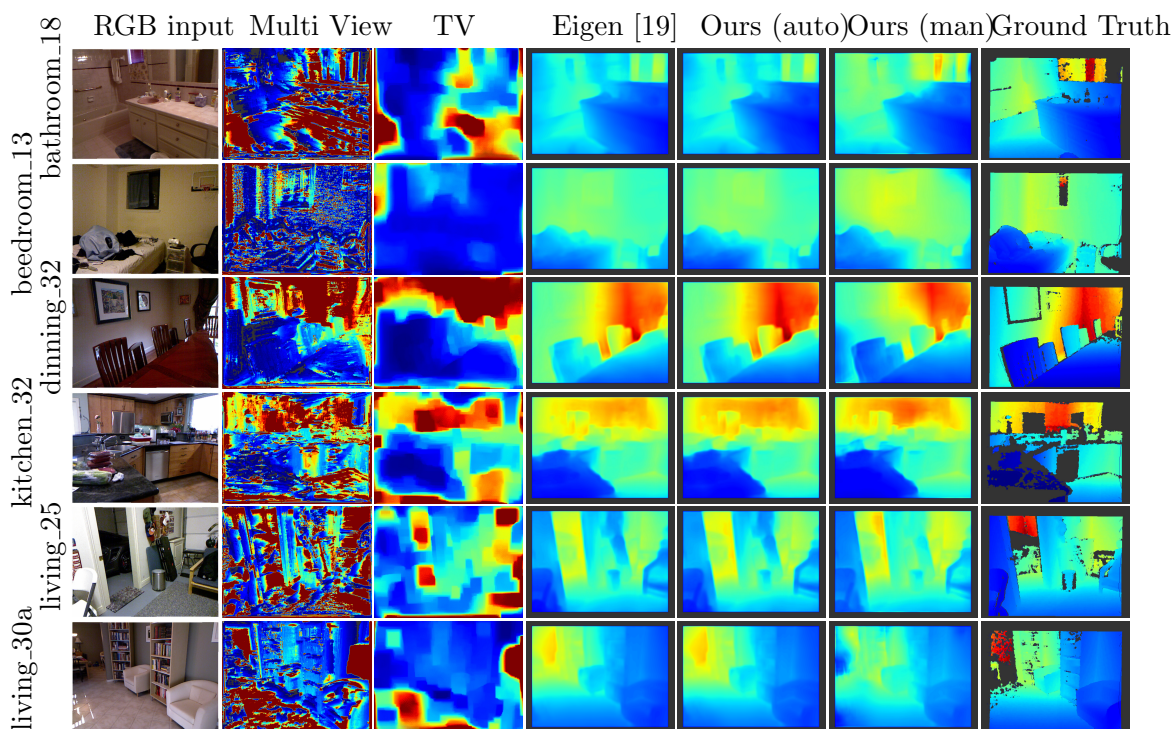


Figura 5.1: Imágenes de profundidad del conjunto de datos NYUDepth v2 [28]. El rango de colores ha sido normalizado por filas para facilitar la comparación entre diferentes métodos. *Primera columna:* Imagen de referencia RGB, *segunda columna* estimación ruidosa del *multi-view*, *tercera columna:* estimación regularizada con la norma TV del *multi-view*, *cuarta columna:* estimación de profundidad del *single-view* [19], *quinta columna:* nuestra fusión de profundidad con selección automática de puntos, *sexta columna:* nuestra fusión de profundidad con selección manual de puntos, y *séptima columna:* *ground truth* tomado por una cámara Kinect. La imagen de profundidad se representa en una escala continua de colores de azul oscuro para los puntos más cercanos, verde para los puntos intermedios y rojo para los puntos lejanos. La figura se ve mejor en color.

fusión fueron fijados experimentalmente con un conjunto pequeño de secuencias a parte. La selección del algoritmo de Gutierrez-Gomez *et al.* se realizó tras hacer una evaluación de los métodos más relevantes del estado del arte, para más detalles ver la Sección 5.2.

Para evaluar los métodos, calculamos el error medio en profundidad en todos los píxeles de la imagen de los que disponemos de *ground truth* para cada método. Los resultados están resumidos en la Tabla 5.1. Nuestro método mejora a la regularización TV en ambos conjuntos de datos, obteniendo una mejora media sobre el 50%. Como esperábamos, la regularización TV funciona mejor en las secuencias del TUM y consigue errores menores, pero en términos de mejora no parece haber grandes diferencias entre ambos conjuntos de datos. Nuestra fusión también mejora las reconstrucciones hechas por el método *single-view*. Sin embargo, en este caso la mejora no es tanta y esta cerca

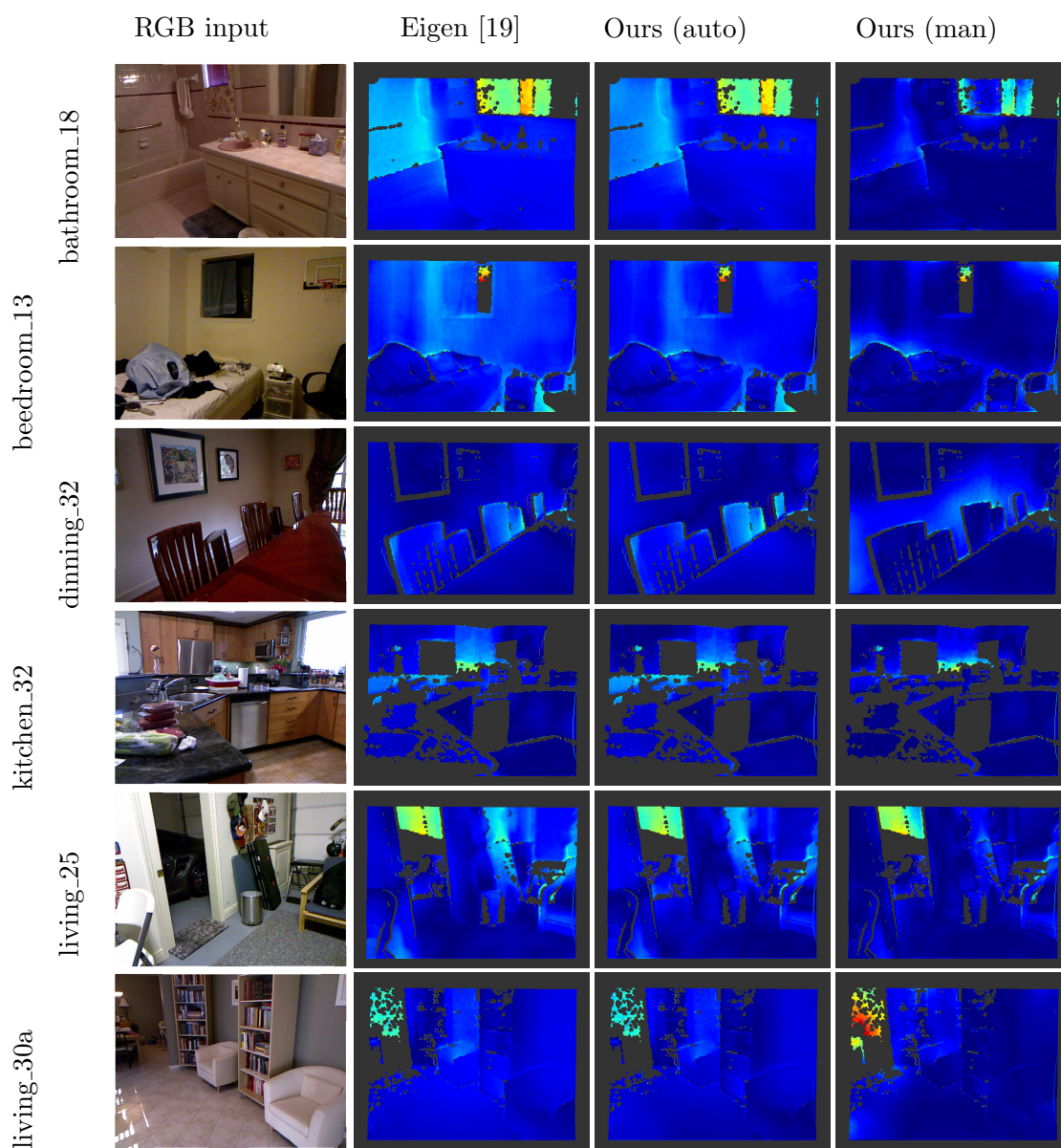


Figura 5.2: Imágenes de error en secuencias del conjunto de datos NYUDepth v2 [28]. El rango de colores ha sido normalizado por filas para facilitar la comparación entre diferentes métodos. *Primera columna*: Imagen de referencia RGB, *segunda columna*: estimación de profundidad del *single-view* [19], *tercera columna*: nuestra fusión de profundidad con selección automática de puntos y *cuarta columna*: nuestra fusión de profundidad con selección manual de puntos. La figura se ve mejor a color.

del 10% en media. Los dos métodos funcionan de forma similar en ambos conjuntos de datos, pero salvo en una secuencia, nuestro método siempre es mejor o igual de bueno

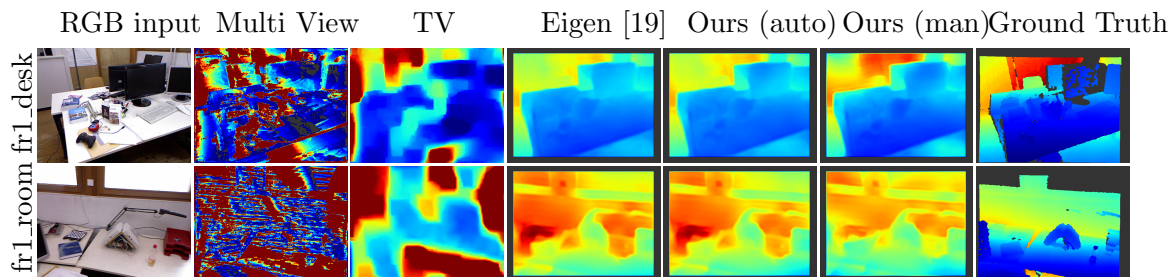


Figura 5.3: Imágenes de profundidad del conjunto de datos TUM Dataset [31].. El rango de colores ha sido normalizado por filas para facilitar la comparación entre diferentes métodos. *Primera columna:* Imagen de referencia RGB, *segunda columna* estimación ruidosa del *multi-view*, *tercera columna:* estimación regularizada con la norma TV del *multi-view*, *cuarta columna:* estimación de profundidad del *single-view* [19], *quinta columna:* nuestra fusión de profundidad con selección automática de puntos, *sexta columna:* nuestra fusión de profundidad con selección manual de puntos, y *séptima columna:* *ground truth* tomado por una cámara Kinect. La imagen de profundidad se representa en una escala continua de colores de azul oscuro para los puntos más cercanos, verde para los puntos intermedios y rojo para los puntos lejanos. La figura se ve mejor en color.

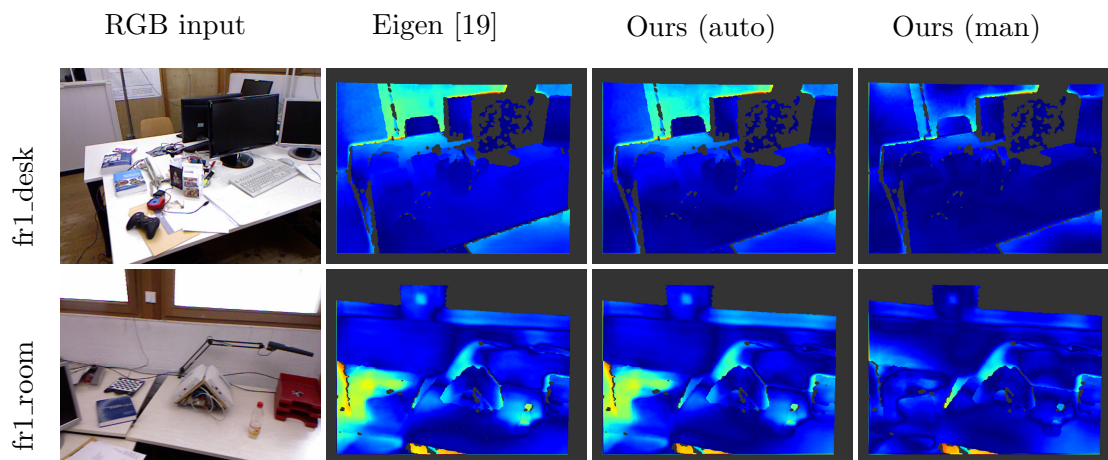


Figura 5.4: Imágenes de error en secuencias del conjunto de datos TUM Dataset [31].. El rango de colores ha sido normalizado por filas para facilitar la comparación entre diferentes métodos. *Primera columna:* Imagen de referencia RGB, *segunda columna:* estimación de profundidad del *single-view* [19], *tercera columna:* nuestra fusión de profundidad con selección automática de puntos y *cuarta columna:* nuestra fusión de profundidad con selección manual de puntos. La figura se ve mejor a color.

que la reconstrucción profunda del *single-view*.

La Tabla 5.2 muestra los errores de profundidad cuando el conjunto de puntos seleccionados del *multi-view* no contiene espurios. Seleccionamos los puntos utilizando el

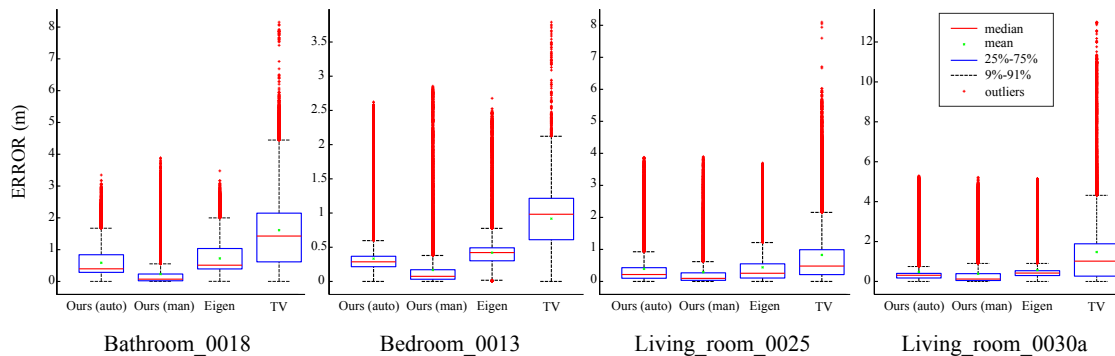


Figura 5.5: Diagrama de cajas y bigotes para la distribución de errores por píxel de cuatro secuencias en las que nos hemos evaluado. De derecha a izquierda comparamos nuestro método con selección automática de puntos, nuestro método con selección manual, el método *single-view* de Eigen *et al.* [19] y la regularización TV para la estimación *multi-view*.

ground truth, los datos del canal de profundidad de la Kinect, y mantenemos únicamente los puntos cuyo error de profundidad es menor a 10cm. Los resultados para todas las secuencias superan cualquier otro método que hayamos probado, consiguiendo mejoras al rededor del 70 % y el 38 % con respecto a la regularización TV y a [19], respectivamente. Aunque esperado, estos resultados resaltan el impacto de los espurios y la necesidad de una buena selección de puntos. Además nos da a conocer una cota superior y demuestra que todavía queda mucho margen de mejora en la última parte de nuestro algoritmo.

Finalmente, presentamos los resultados de algunas imágenes de cada una de las secuencias de cada conjunto de datos. La Figura 5.1 y la Figura 5.3 muestran las imágenes de profundidad obtenidas para los conjuntos de datos NYUDepth v2 y TUM, respectivamente. La mejora con respecto a la regularización del método *multi-view* es clara visualmente dado que la estructura de la profundidad es mucho más consistente. Las mejoras con respecto a la estimación *single-view* es más sutil y se ve mejor en las correspondientes imágenes de error de profundidad de la Figura 5.2 y la Figura 5.4. Normalmente, la mejora proviene de una mejor localización de algunas estructuras locales. Por ejemplo, las paredes están más oscuras en las imágenes de error en profundidad (ver las escenas bathroom_18 o dining_32 en la Figura 5.2). El efecto es más evidente cuando los puntos del *multi-view* son seleccionados manualmente, pero es el mismo efecto obtenido con la selección automática. Este mejor alineamiento de estructuras locales reduce el error como se puede ver el diagrama de cajas y bigotes de cada secuencia mostrado en la Figura 5.5.

	RPE RMSE (m)		
	Fovis	Gutierrez-Gómez [33]	RGB-D SLAM
fr1_360	0.0859	0.0821	0.103
fr1_desk2	0.0505	0.0385	0.102
fr1_room	0.0590	0.0426	0.219
fr2_desk	0.0134	0.0117	-
fr2_large_loop	0.1406	0.0677	-
fr2_pioner_slam	0.1241	0.1549	-

Tabla 5.3: Evaluación de diferentes algoritmos de estimación de la trayectoria de la cámara en el TUM-dataset [31]. RPE (*Relative Pose Error*) es el error relativo en metros.

5.2. Evaluación y selección de odometría RGB-D

Desafortunadamente el conjunto de datos NYUv2 no dispone de la pose de la cámara durante sus secuencias, y dado que es necesaria para calcular la profundidad *multi-view* se ha escogido el algoritmo presentado en [33] para calcularla. Se ha escogido este algoritmo tras una revisión del estado del arte en *RGB-D SLAM* y *RGB-D Odometry* siendo el trabajo de D. Gutierrez-Gómez [33] el que obtiene mejor resultado. Utilizamos como métrica de evaluación el *RPE* en metros. *RPE* es el error relativo de movimiento en metros de la estimación de cada uno de los métodos.

En la tabla 5.3 se comparan tres algoritmos: la odometría Fovis [34], la odometría de D. Gutierrez-Gómez y el RGB-D SLAM [35]. Como puede observarse el algoritmo de [33] obtiene mejores resultados en todas las secuencias salvo en una. Concretamente, su error con respecto al RGB-D SLAM esta por debajo de la mitad. Con respecto a Fovis la diferencia es más pequeña, aun así en todas las escenas obtiene mayor precisión, excepto la escena fr2_pioner_slam donde Fovis consigue un error menor.

5.3. Evaluación de tiempos

En esta sección se hace un pequeño análisis de tiempos del algoritmo que hemos presentado. Merece la pena destacar que el objetivo de este proyecto ha sido mejorar la calidad de las reconstrucciones por lo que el algoritmo no ha sido optimizado. Seguramente pequeñas modificaciones en el código podrían acelerar considerablemente su rendimiento. Los datos que se dan a continuación son cotas superiores.

La evaluación de tiempos del sistema se a realizado en una GPU Nvidia Titan X con 12 GiB de memoria soportada por una CPU Intel Core i7-6850K.

La red neuronal ocupa alrededor de 5GiB en memoria por lo que cargarla en memoria consume gran parte de su tiempo de ejecución. El tiempo de realizar la primera

estimación son 42s, sin embargo una vez la red esta cargada en la gráfica el tiempo se reduce. Tras hacer 100 estimaciones el tiempo medio por estimación es de 1,99s. Aproximadamente 1,6s por estimación con la red cargada.

El método *multi-view* consume alrededor de 4s. Y el tiempo consumido por la fusión crece linealmente con el número de puntos seleccionados del método *multi-view*. Con 60 puntos consume 6s con 330 puntos 13,3s y con 600 puntos 20,2s. La cantidad de puntos depende de la escena y de la calidad de la estimación realizada por el método *multi-view*.

5.4. Herramientas y tecnología utilizada

Durante el trabajo se han utilizado diferentes tecnologías. A continuación se citan las tecnologías más relevantes que hemos utilizado para cada una de las partes del proyecto.

Multi-view: Para el método *multi-view* se ha utilizado C++. Se ha aprovechado gran parte del código y funciones aportado por [8].

Single-View: Para el método *single-view* se ha utilizado la red que propusieron los autores de [19] y también los parámetros de la red. La red esta entrenada utilizando Theano [29] y hemos utilizado el mismo *Framework* con interfaz en Python. Además al comienzo del proyecto se probó diferentes modelos de redes utilizando Caffe [30], otro *Framework* de redes neuronales. En el tiempo que utilizamos Caffe, también se implementaron capas de redes neuronales en Python y C++. Todas las redes que hemos utilizado han sido ejecutadas en GPU utilizando CUDA.

Fusión y experimentos: Tanto el algoritmo de fusión, la selección de puntos como los experimentos han sido implementados en Matlab.

6. Conclusiones

En este trabajo hemos presentado un algoritmo de estimación densa de profundidad que fusiona 1) la estimación de profundidad directa a partir de múltiples vistas, y 2) la estimación de profundidad hecha por una red neuronal profunda entrenada con imágenes RGB-D. Nuestra aproximación selecciona los puntos más preciso de la reconstrucción *multi-view* y los fusiona con la estimación de profundidad densa realizada por la red neuronal. Cabe remarcar que los errores de profundidad en la estimación de la red no dependen de la configuración geométrica sino en el contenido de la imagen y por lo tanto la transformación que requieren es no rígida y varía localmente. La estimación de este alineamiento es la contribución principal de este trabajo y el aspecto más novedoso de esta investigación.

Nuestros experimentos muestran que nuestra propuesta mejora sobre otros métodos del estado del arte (Eigen et al. [19] en *single-view* y la regularización TV sobre un método de estimación directo en *multi-view*). Contrario a otras aproximaciones del estado del arte, la información *single-view* que utilizamos esta completamente basada en los datos y no hace asunciones sobre la escena. El trabajo futuro, como sugieren los resultados, intentara mejorar la selección de puntos de la estimación directa y la fusión entre las dos profundidades, por ejemplo utilizando métodos iterativos.

Debido a originalidad del algoritmo y a la mejora que presenta con respecto a otros métodos del estado del arte, el contenido de este trabajo esta actualmente bajo revisión para ser publicado en la conferencia internacional del IEEE *International Conference on Robotics and Automation (ICRA)*. El titulo del articulo es “*Deep Single and Direct Multi-View Depth Fusion*” y esta incluido en el Anexo A.

Bibliografía

- [1] B. Benfold and I. Reid, “Guiding visual surveillance by tracking human attention,” in *Proceedings of the 20th British Machine Vision Conference*, September 2009.
- [2] M. Meilland, C. Barat, and A. Comport, “3d high dynamic range dense visual slam and its application to real-time object re-lighting,” in *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pp. 143–152, IEEE, 2013.
- [3] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Advances in Neural Information Processing Systems*, pp. 2366–2374, 2014.
- [4] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense tracking and mapping in real-time,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2320–2327, IEEE, 2011.
- [5] G. Graber, T. Pock, and H. Bischof, “Online 3d reconstruction using convex optimization,” in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 708–711, IEEE, 2011.
- [6] J. Stühmer, S. Gumhold, and D. Cremers, “Real-time dense geometry from a hand-held camera,” in *Joint Pattern Recognition Symposium*, pp. 11–20, Springer, 2010.
- [7] A. Concha, W. Hussain, L. Montano, and J. Civera, “Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping,”
- [8] A. Concha and J. Civera, “Using superpixels in monocular slam,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 365–372, IEEE, 2014.
- [9] V. Hedau, D. Hoiem, and D. Forsyth, “Recovering the spatial layout of cluttered rooms,” in *2009 IEEE 12th international conference on computer vision*, pp. 1849–1856, IEEE, 2009.
- [10] P. Pinies, L. M. Paz, and P. Newman, “Dense mono reconstruction: Living with the pain of the plain plane,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 5226–5231, IEEE, 2015.

- [11] P. Piniés, L. M. Paz, and P. Newman, “Too much TV is bad: Dense reconstruction from sparse laser with non-convex regularisation,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 135–142, IEEE, 2015.
- [12] A. Concha, W. Hussain, L. Montano, and J. Civera, “Incorporating scene priors to dense monocular mapping,” *Autonomous Robots*, vol. 39, no. 3, pp. 279–292, 2015.
- [13] D. F. Fouhey, A. Gupta, and M. Hebert, “Data-driven 3d primitives for single image understanding,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3392–3399, 2013.
- [14] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.
- [15] R. Mur-Artal, J. Montiel, and J. D. Tardos, “ORB-SLAM: a versatile and accurate monocular SLAM system,” *Robotics, IEEE Transactions on*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [16] J. Ens and P. Lawrence, “An investigation of methods for determining depth from focus,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 2, pp. 97–108, 1993.
- [17] P. Sturm and S. Maybank, “A method for interactive 3d reconstruction of piecewise planar objects from single images,” in *The 10th British machine vision conference (BMVC’99)*, pp. 265–274, The British Machine Vision Association (BMVA), 1999.
- [18] A. Saxena, M. Sun, and A. Y. Ng, “Make3D: Learning 3D scene structure from a single still image,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2009.
- [19] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658, 2015.
- [20] F. Liu, C. Shen, and G. Lin, “Deep convolutional neural fields for depth estimation from a single image,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5162–5170, 2015.
- [21] R. Y. Tsai, “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off the shelf tv cameras and lenses,” *IEEE Journal of Robotics and Automation*, pp. 323–344, 1987.
- [22] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, 2013.
- [23] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” *arXiv preprint arXiv:1411.4389*, 2014.

- [24] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” *arXiv preprint arXiv:1412.2306*, 2014.
- [25] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, “Convolutional-recursive deep learning for 3d object classification,” in *Advances in Neural Information Processing Systems*, pp. 665–673, 2012.
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- [27] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [28] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, “Indoor segmentation and support inference from RGBD Images,” in *ECCV*, 2012.
- [29] Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, vol. abs/1605.02688, May 2016.
- [30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
- [32] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison, “Applications of legendre-fenchel transformation to computer vision problems,” *Department of Computing at Imperial College London. DTR11-7*, vol. 45, 2011.
- [33] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. Guerrero, “Inverse depth for accurate photometric and geometric error minimisation in rgb-d dense visual odometry,”
- [34] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, “Visual odometry and mapping for autonomous flight using an rgb-d camera,” in *International Symposium on Robotics Research (ISRR)*, pp. 1–16, 2011.
- [35] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, “An evaluation of the RGB-D SLAM system,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1691–1696, IEEE, 2012.