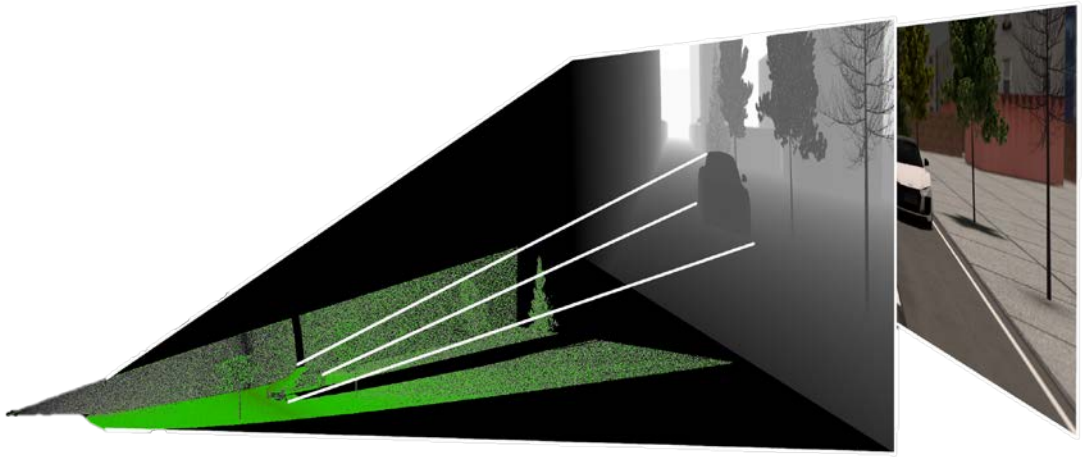
Universidad
Rey Juan Carlos
Escuela Técnica Superior
Ingeniería de Telecomunicación



Visión Artificial

9. Visión 3D

JOSÉ MIGUEL GUERRERO HERNÁNDEZ

EMAIL: JOSEMIGUEL.GUERRERO@URJC.ES

Índice de contenidos

Índice de contenidos

1. Introducción

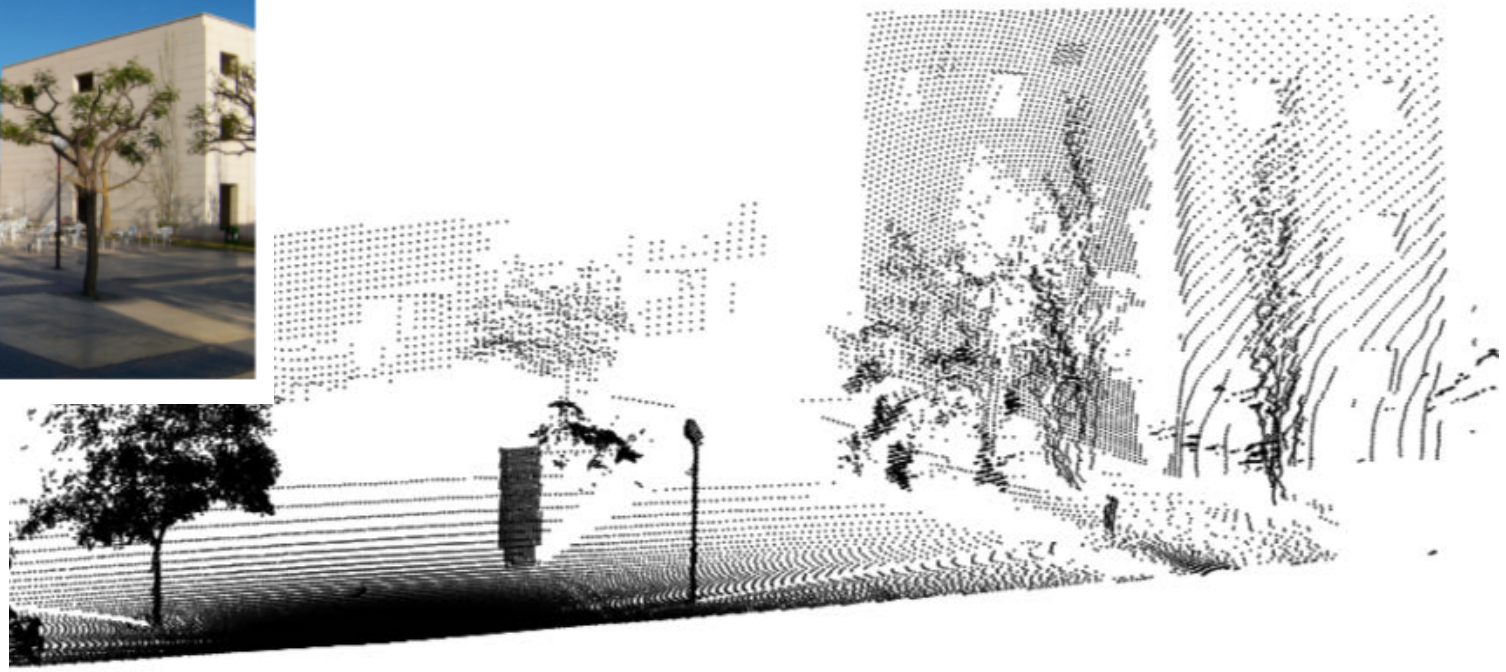
- Los sensores 3D permiten obtener la información de las superficies que componen una escena escaneando puntos sobre estas
- Pueden incluir información de color (o no)
- Características:
  - Alcance
  - Resolución
  - Precisión
- Tecnologías: estéreo, RGB-D, infrarrojos, láser (ToF)
- El resultado es una nube de puntos compuestos por sus coordenadas XYZ y, opcionalmente, color RGB

1. Introducción

- El tipo de información que los sensores 3D capturan de la escena depende en gran medida del tipo de sensor utilizado
- Las principales formas de proporcionar dicha información son las siguientes:
  - **Nube de puntos**
  - **Imagen de profundidad**
  - **Imagen estereoscópica**

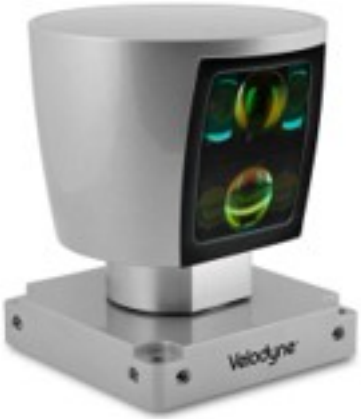
1. Introducción: láser

- **Nube de puntos**: Se proporcionan las coordenadas tridimensionales de puntos del espacio, normalmente en el sistema de referencia del propio sensor. Éste es el formato que devuelven usualmente los **escáneres láser** por tiempo de vuelo y cambio de fase, que actúan en grandes entornos



1. Introducción: láser

- Time of Flight: se emite una señal y se espera hasta que choque con algo y vuelva al origen
- Cuanto mayor longitud de onda, más precisión y alcance
- El láser es dirigido por espejos en 1 o 2 dimensiones
- Posibilidad de montar un laser 2D en un sistema pan-tilt



1. Introducción: láser

- **Escáneres láser**
- Características:
  - Largo alcance (hasta 6 Km.)
  - Precisión (<1mm. Independiente a distancia)
  - Resolución alta
  - Reflectividad o color RGB (no es común)
  - Precio muy alto (5k a >10k €)
  - Peso muy alto (y consumo)

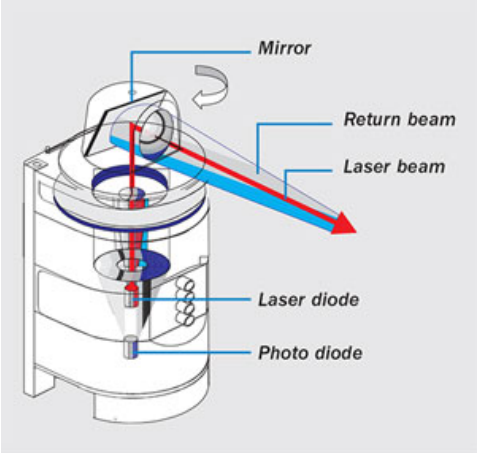
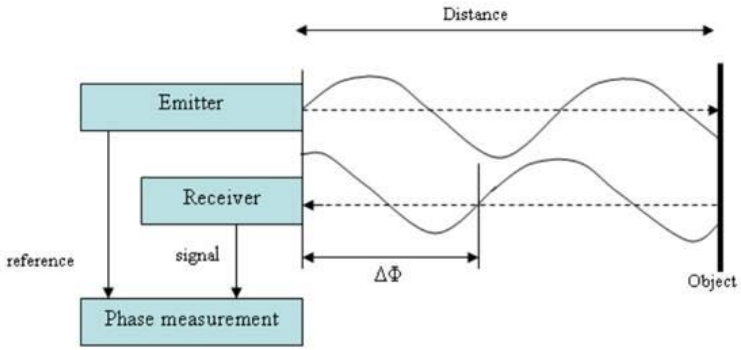


Illustration of LIDAR sensor demonstrating the time of flight principle. *(Courtesy of SICK, Inc.)*

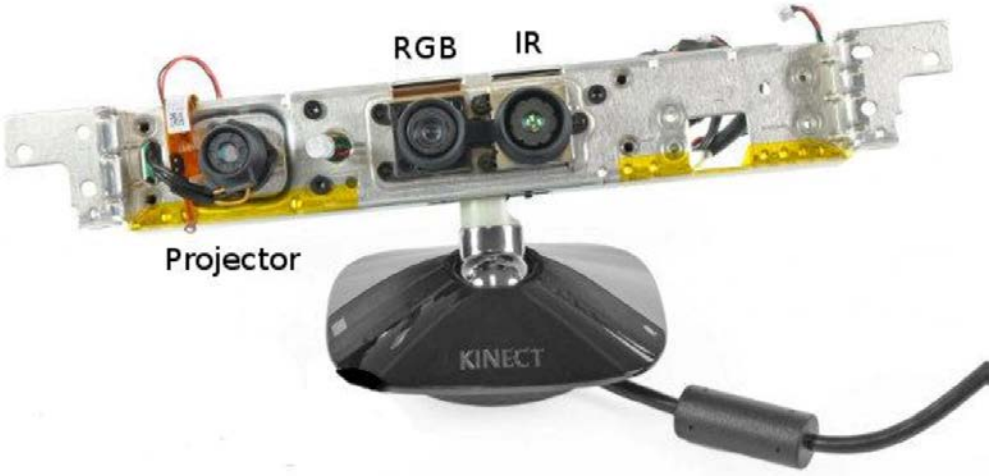


1. Introducción: RGB-D

- **Imagen de profundidad**: La información es una imagen donde cada píxel contiene la distancia al centro óptico de la cámara utilizada por el sensor. En este caso también se obtiene la imagen color, o en escala de grises de la escena captada. Formato habitual proporcionado por **cámaras RGB-D**

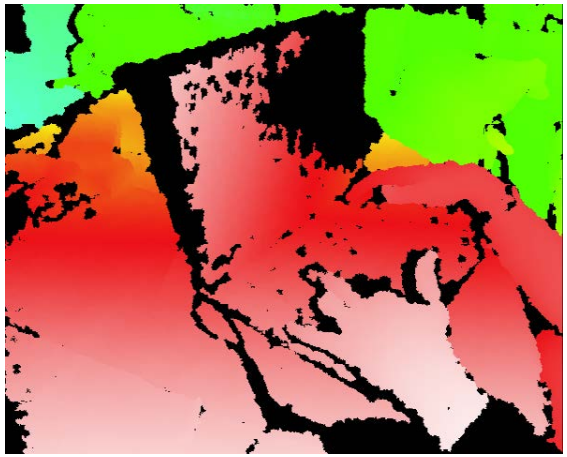
1. Introducción: RGB-D

- Las **cámaras RGB-D** son sensores compuestos por un emisor-receptor de luz IR y una cámara 2D
- El emisor IR proyecta un patrón conocido sobre la escena
- El receptor capta el patrón proyectado y analiza las divergencias con el patrón original para obtener la profundidad de los puntos
- La cámara 2D se utiliza para asignar color RGB a los puntos 3D



1. Introducción: RGB-D

- **Cámaras RGB-D**
- Características:
  - Alcance limitado hasta 10m
  - Baja resolución (640x480)
  - Error de medida elevado (10%)
  - Muy bajo coste
  - Permiten trabajar en la oscuridad

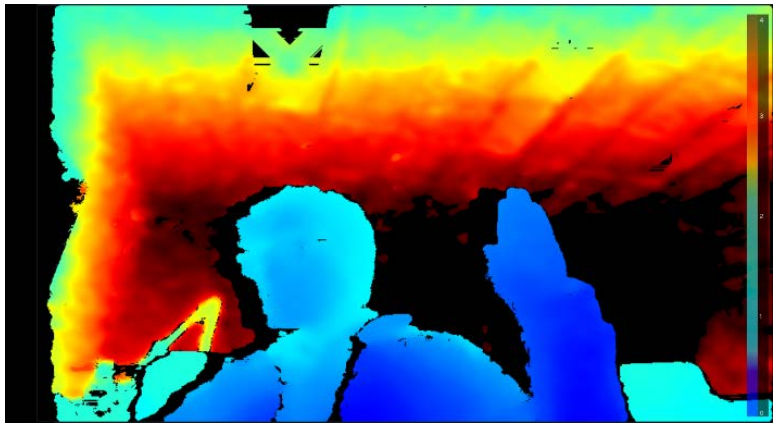


1. Introducción: estereoscopía

- **Imagen estereoscópica**: Se proporcionan dos imágenes en color, cada una correspondiente a un sensor de imagen. Se utiliza normalmente para aplicaciones de visualización en tres dimensiones. Este formato es devuelto por los **sistemas estereoscópicos**

1. Introducción: estereoscopía

- **Cámaras estereoscópicas**
- Características:
  - Nubes de puntos 3D con color (escala de gris o RGB)
  - Alcance limitado (baseline) 5-15m
  - Alta resolución (1280x1024)
  - Error de medida elevado (10%)
  - Problemas con zonas sin texturas
  - Coste medio. Hardware adicional. El cálculo de la disparidad se hace por software



Índice de contenidos

2. Visión estereoscópica

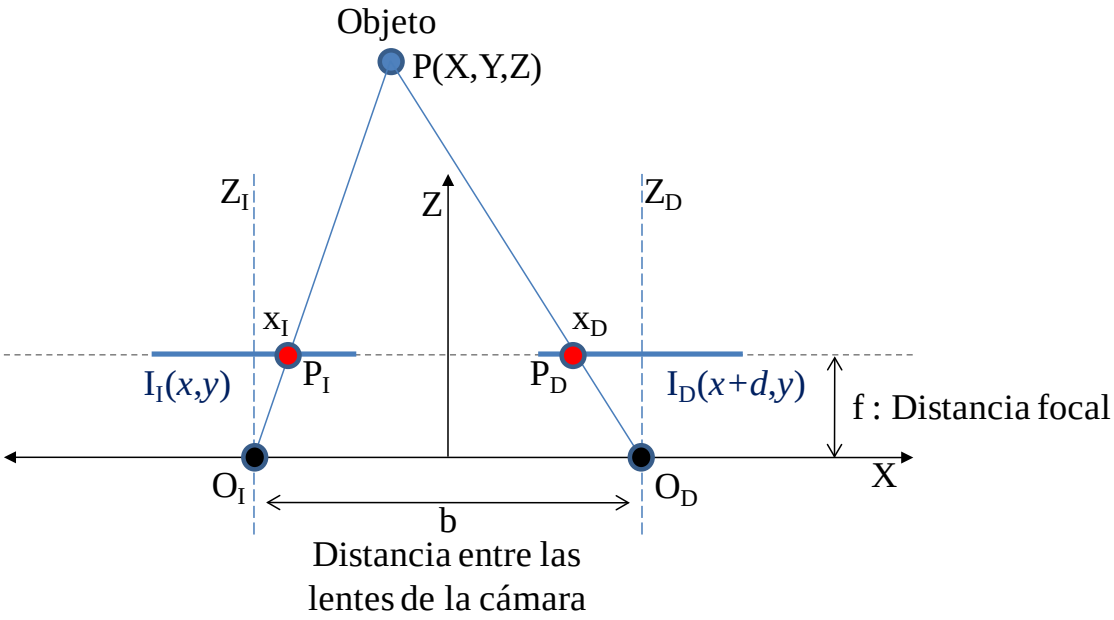
- Las técnicas de **visión estereoscópica** permiten inferir información geométrica (3D) sobre la escena
- Se basan en aprovechar la información procedente de **dos imágenes** distintas para suplir la pérdida de información que supone el proceso de proyección de la escena en la imagen 2D
- Como usamos dos cámaras, los objetos aparecerán en distinta posición en una imagen (izquierda) y en la otra (derecha)
- Para poder realizar la reconstrucción es necesario identificar las **proyecciones de un mismo punto** de la escena en cada una de las imágenes, para después, utilizar la **información geométrica** sobre la disposición relativa de las imágenes y así poder obtener la posición de dichos puntos en el espacio

2. Visión estereoscópica

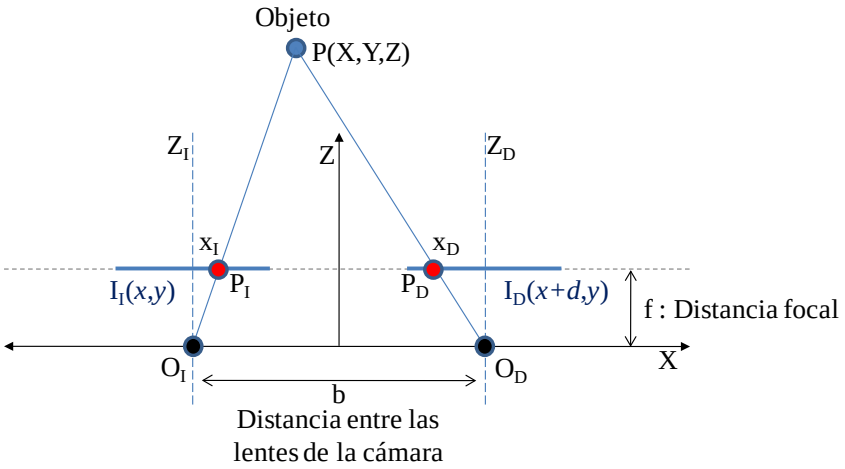
- La **restricción epipolar** liga la geometría de ambas imágenes y resulta clave para facilitar la búsqueda de correspondencias, así como para inferir la posición y orientación de una cámara respecto a la otra cuando el par estéreo no ha podido ser calibrado
- Cuanto más cerca estén los objetos de la cámara, mayor será su cambio relativo
- Este desplazamiento relativo de los objetos en las dos imágenes es lo que se conoce como **disparidad**
- Existe una relación directa entre la **profundidad** y la **disparidad**

2. Geometría de los sistemas estéreo

- El sistema más sencillo es aquel que utiliza dos cámaras 2D en un mismo marco, alineadas entre sí:



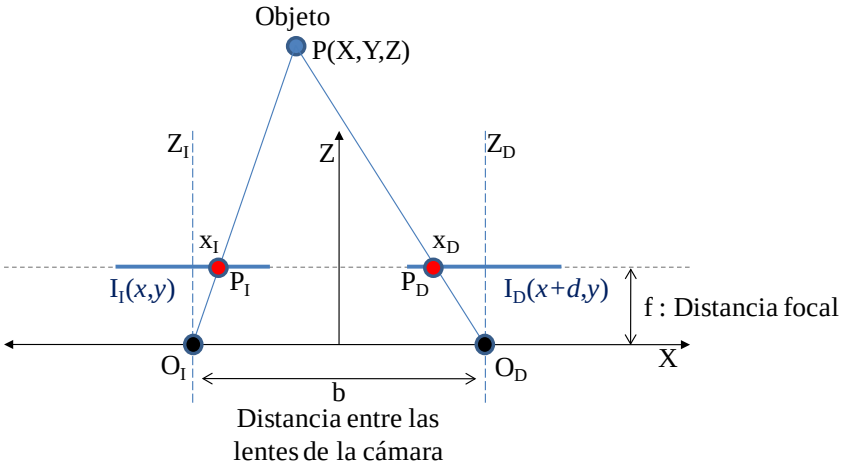
2. Geometría de los sistemas estéreo



donde:

- (Z_I y Z_D) son los ejes ópticos mutuamente paralelos y separados por una distancia horizontal (b) que se denomina línea base
- (I_I e I_D) son los planos imagen y sus centros de proyección O_I y O_D respectivamente
- f es la longitud focal efectiva de cada cámara y la disparidad d se calcula para cada par de puntos emparejados $P_I(x_I, y_I)$ y $P_D(x_D, y_D)$ como $d = (x_I - x_D)$

2. Geometría de los sistemas estéreo

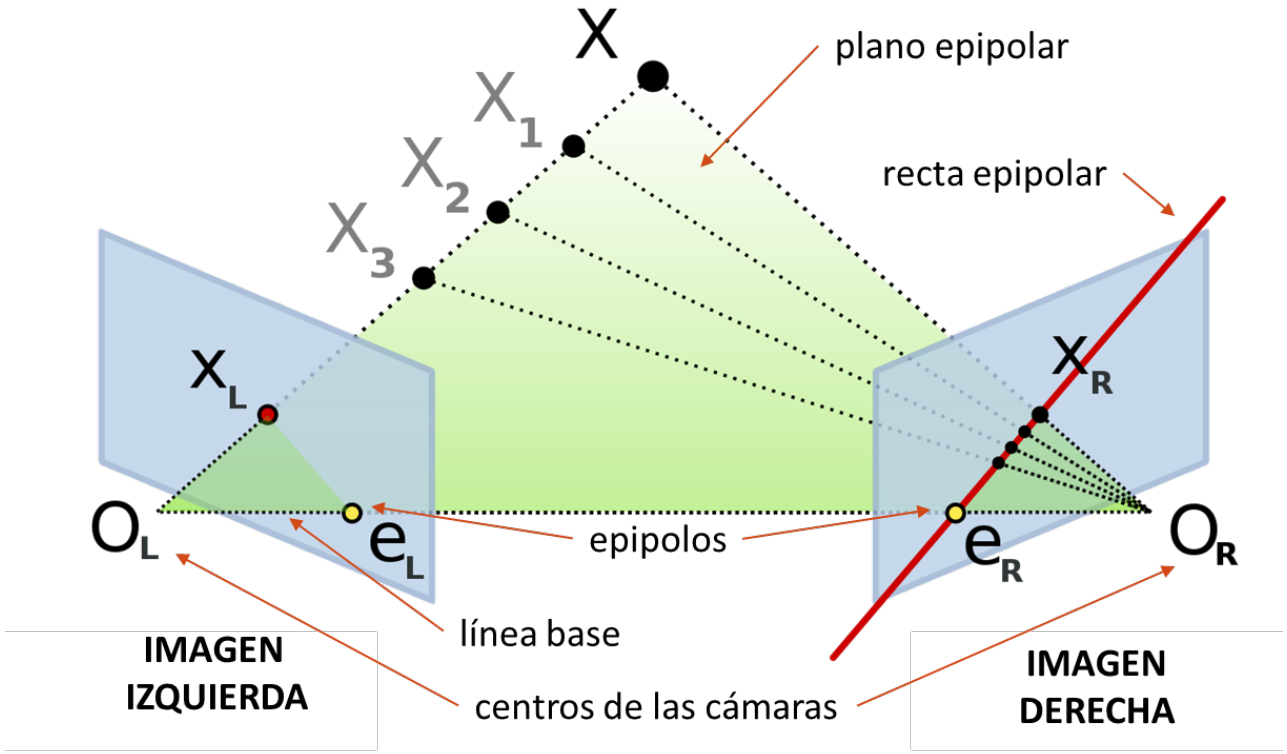


- Resolviendo el siguiente sistema obtenemos:

$$\left.\begin{array}{l} O_I: \dfrac{\frac{b}{2}+X}{Z}=\dfrac{x_I}{f} \\ O_D: -\dfrac{\frac{b}{2}-X}{Z}=\dfrac{x_D}{f} \end{array}\right\} \Rightarrow \left.\begin{array}{l} x_I=\dfrac{f}{Z}\left(X+\dfrac{b}{2}\right) \\ x_D=\dfrac{f}{Z}\left(X-\dfrac{b}{2}\right) \end{array}\right\} \Rightarrow \mathrm{d}=x_I-x_D=\frac{fb}{Z} \Rightarrow Z=\frac{fb}{d}$$

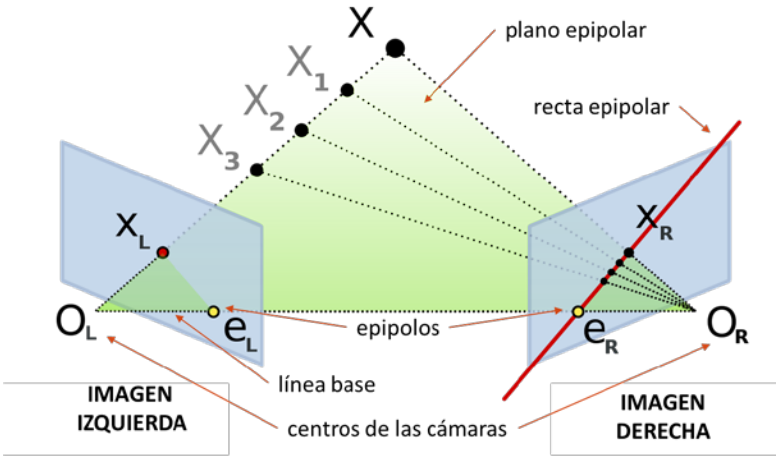
2. Geometría de los sistemas estéreo

- Cuando el sistema no tiene las dos cámaras alineadas, las entidades geométricas son:



2. Geometría de los sistemas estéreo

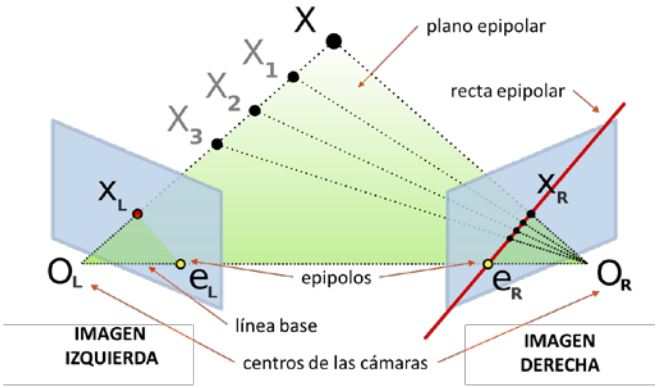
- Las entidades geométricas son:



- **Línea base**: es el segmento 3D que une los centros de proyección de ambas cámaras
- **Plano epipolar**: Es el plano que queda definido por un punto de la escena y los centros de proyección de ambas cámaras. Como consecuencia, los planos epipolares forman un haz de planos cuyo eje es precisamente la línea base

2. Geometría de los sistemas estéreo

- Las entidades geométricas son:



  - **Epipolo**: para cada vista del sistema estéreo, el epipolo es el punto en el que la línea base corta al plano imagen asociado a dicha vista. El epipolo es pues la proyección en una cámara, del centro de proyección de la otra cámara que forma el par estéreo. El epipolo puede caer fuera del trozo del plano imagen que devuelve la cámara cuando la línea base es paralela al plano imagen (caso anterior)
  - **Recta epipolar**: es la recta intersección del plano epipolar con el plano imagen

2. Correspondencia de puntos

- Debemos ser capaces de encontrar para cada punto en una imagen, su correspondencia en la otra imagen
- Se suelen usar dos métodos:
  - **Correlación**: se basa en usar una zona alrededor de un punto para "compararla" con puntos de la otra imagen. Estéreo denso
  - **Características** (esquinas, aristas, etc.): Se identifican características para calcular su profundidad. Estéreo disperso
- En zonas uniformes no se pueden encontrar correspondencias, se usan discontinuidades

2. Correspondencia de puntos

- Correlación:
- La correlación es un proceso que consiste en sumar el resultado de multiplicar píxel a píxel una cierta vecindad en las dos imágenes:
  1. Se define una cierta ventana de tamaño W
  2. Para cada punto (x, y) de la imagen izquierda calculamos el desplazamiento (d_x, d_y) que se produce en la imagen derecha

$$c(x, y, d_x, d_y) = \sum_{k=-W}^{W} \sum_{l=-W}^{W} I_I(x + k, y + l) * I_D(x + k + d_x, y + l + d_y)$$

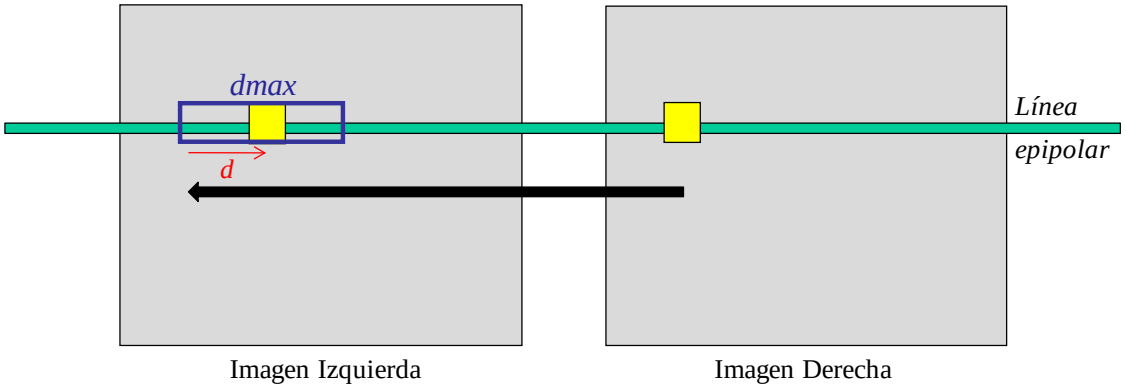
  3. Para cada posible desplazamiento (d_x, d_y) se calcula:

$$(d_x, d_y) = \max\{c(x, y, d_x, d_y)\}$$

  4. Encontramos el desplazamiento que maximiza la anterior función

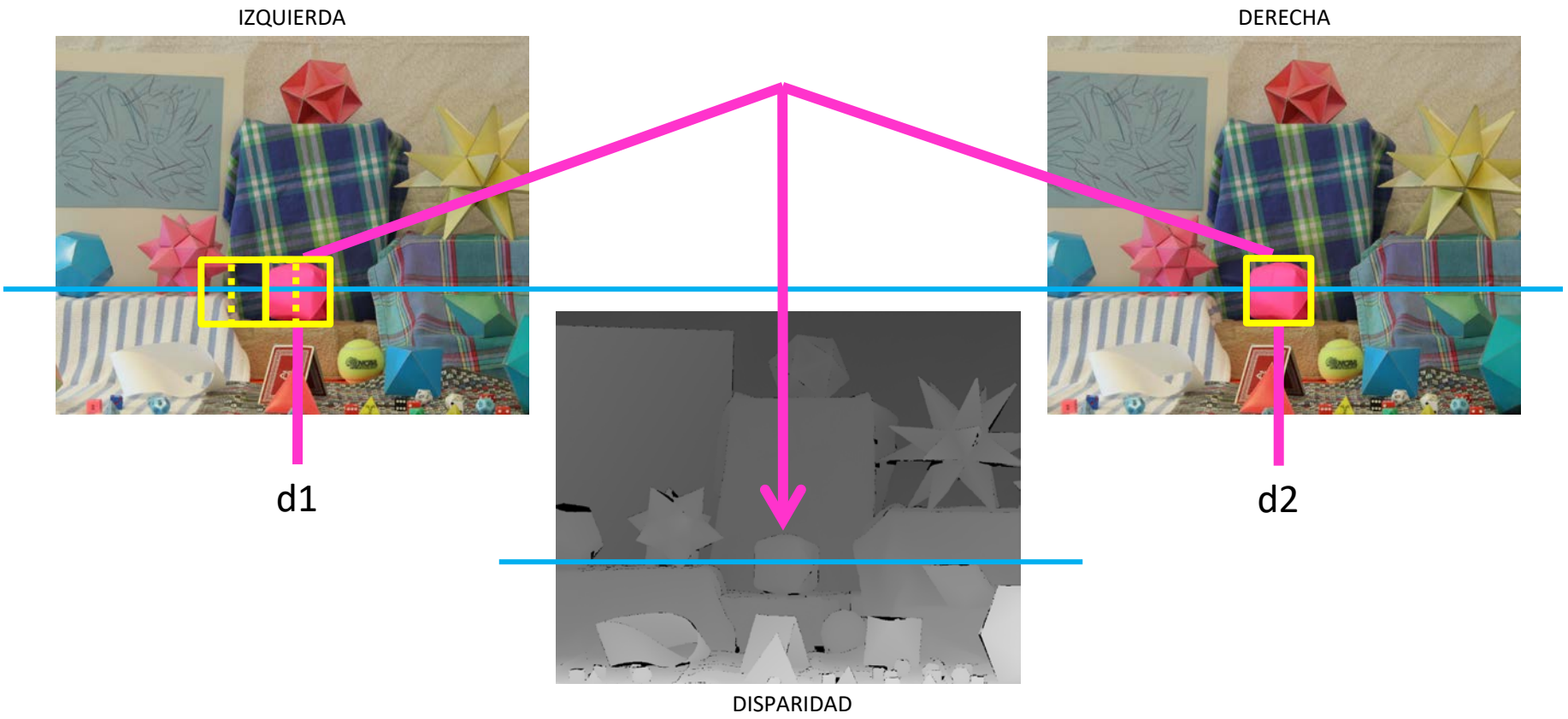
2. Correspondencia de puntos

- Correlación:
- Hemos comentado que para cada píxel buscamos el desplazamiento que se produce en la otra imagen
- Podemos hacer uso de la geometría reducir la búsqueda (geometría epipolar)
- En vez de buscar por toda la imagen, solo buscamos por la línea epipolar correspondiente



2. Correspondencia de puntos

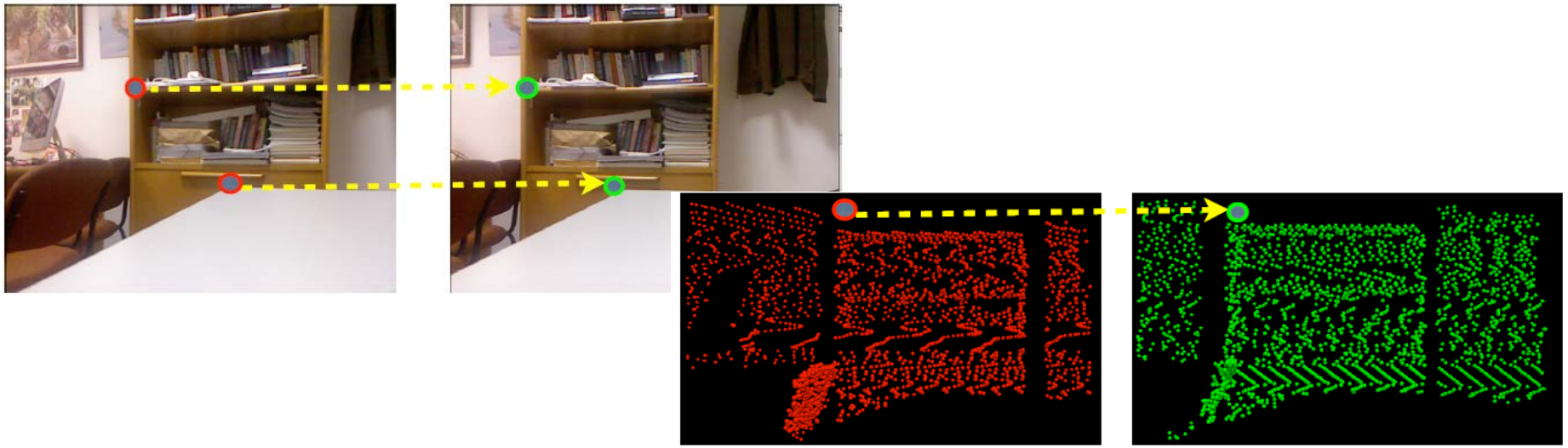
- Correlación:



Índice de contenidos

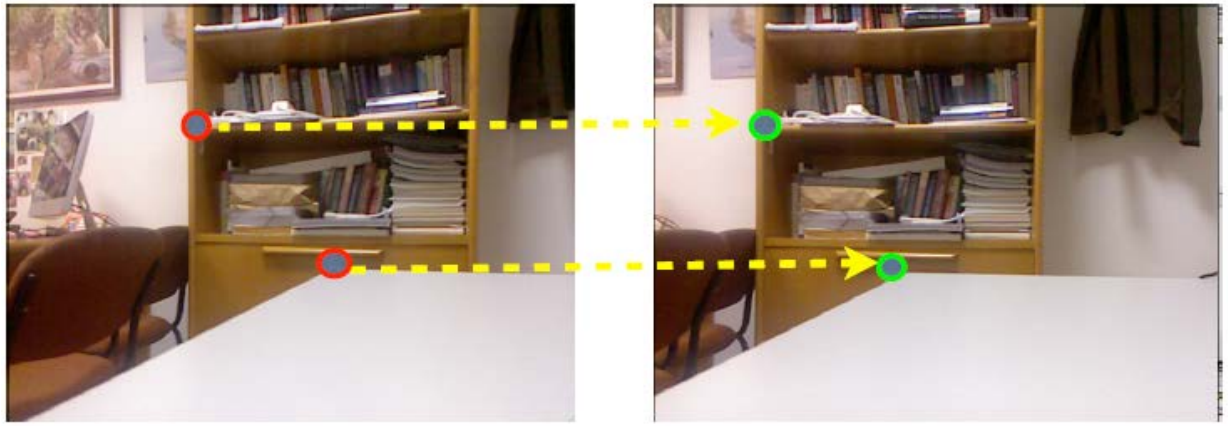
3. Métodos de registro 3D

- Dados dos scans observados desde dos puntos de vista diferentes, en nuestro caso, dos nubes de puntos con su respectivo sistema de referencia local
- Se trata de encontrar la transformación rígida (rotación y traslación) entre los dos sistemas de referencia que alinea puntos correspondientes de las dos nubes (scan matching)



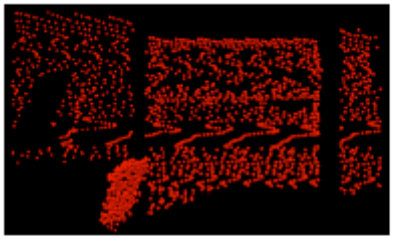
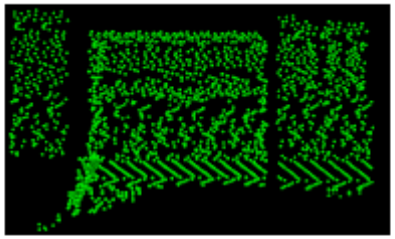
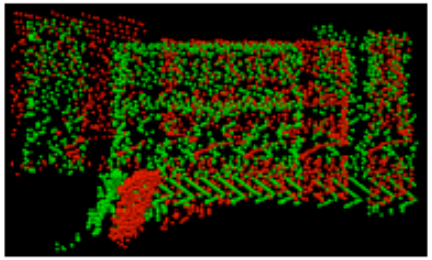
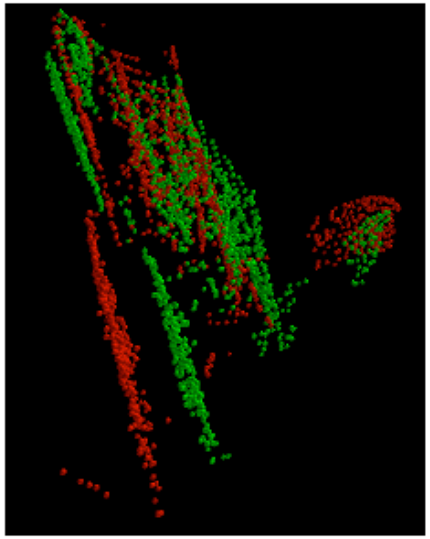
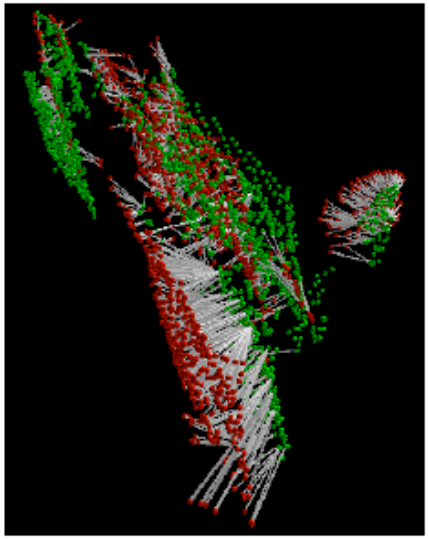
3. Métodos de registro 3D

- **Matching basado en características**:
  - Extraer características (features) distinguibles que permitan ser identificadas en ambos scans
  - A partir del emparejamiento correspondiente entre características de ambos scans (RANSAC), se calcula la transformación rígida que las alinea



3. Métodos de registro 3D

- **Matching como un problema de optimización**:
  - Usar una función de coste que mide la calidad del alineamiento entre puntos de ambos scans
  - Normalmente en un proceso iterativo, se determina la transformación rígida que minimiza dicha función de coste

3. Métodos de registro 3D

- Cálculo de la transformación rígida

- Solución en **forma cerrada**:
  - Aplicable cuando se conocen las correspondencias entre los puntos
  - 4 métodos matemáticos
    - Usando Descomposición en Valores Singulares de una matriz (SVD)
    - Usando matrices ortonormales
    - Usando cuaterniones (quaternions)
    - Usando cuaterniones duales (dual quaternions)

- Solución **iterativa**:
  - Cuando las correspondencias entre los puntos son desconocidas
  - Algoritmo **ICP** (Iterative Closest Points)

3. Métodos de registro 3D

- Dados dos conjuntos de puntos 3D, encontrar la transformación que alinea los conjuntos
- Si los conjuntos han sido capturados con un sensor 3D en dos poses consecutivas del robot, esa transformación coincide con el movimiento realizado por el robot
- **Iterative Closest Points:**
  - Propuesto por Besl, P. & McKay, N. en 1992
  - Método iterativo que permite alinear dos superficies
  - Se asume un solapamiento parcial (total en el caso ideal) y un conocimiento inicial de un alineamiento grueso
  - Permite encontrar la trasformación (en 3D) que mejor alinea los datos de entrada

3. Métodos de registro 3D: ICP

- Inicialmente supondremos que las superficies a alinear vienen caracterizadas por conjuntos de puntos 3D
- Se utiliza una medida de distancia entre los elementos de un conjunto y los del otro
- Inicialmente la distancia euclídea entre dos puntos
- Así, la entrada al algoritmo será:
  - El modelo, o conjunto de puntos de una de las superficies
  - La escena, o conjunto de puntos de la segunda superficie
  - La transformación inicial (odometría, acción, …)
  - Criterio de parada

3. Métodos de registro 3D: ICP

- Algoritmo:
  - Repetir hasta que se cumpla el criterio de parada:
    1. Aplicar la transformación actual al conjunto escena
    2. Para cada punto de la escena transformada, emparejar con el más cercano del conjunto modelo
    3. Estimar la transformación (rotación + traslación) a partir de los emparejamientos de manera que minimice la función de coste utilizando uno de los métodos en forma cerrada

3. Métodos de registro 3D: ICP

- Algoritmo:
  - Sean dos conjuntos de puntos 3D, M (**model set**) y D (**data set**)

$$|M| = N_m \qquad |D| = N_d$$

  - Se trata de encontrar la transformación rígida (R, t) que minimiza la siguiente función de coste:

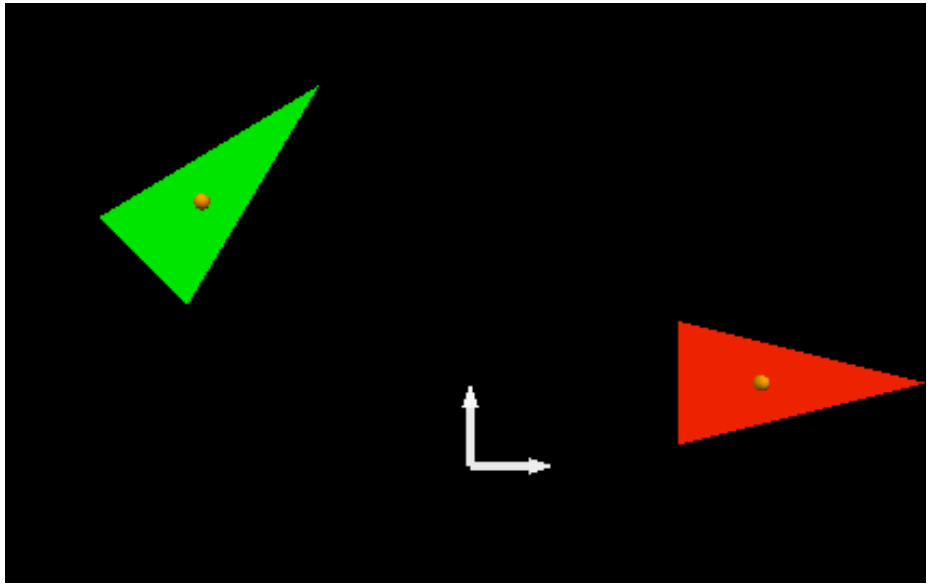
$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left\| m_i - \left(R d_j + t\right) \right\|^2$$

donde $w_{i,j}$ determinan las correspondencias de puntos:

$$w_{i,j} = \begin{cases} 1 \; si \; m_i \; empareja \; con \; d_j \\ 0 \; en \; otro \; caso \end{cases}$$

3. Métodos de registro 3D: ICP

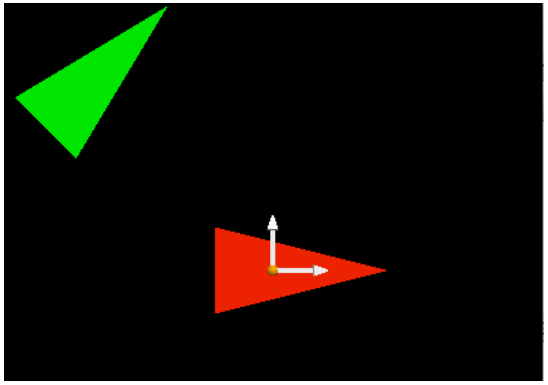
- Generalmente se calcula primero la rotación **R** y después la traslación **t**
- **Idea**: en la solución correcta las 2 nubes de puntos deberían tener el mismo centroide



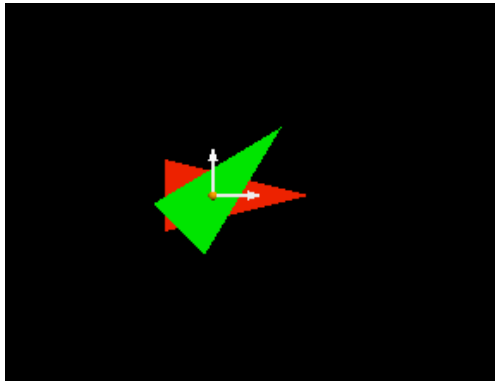
3. Métodos de registro 3D: ICP

- Modificar las 2 nubes de puntos restándoles sus respectivos centroides

$$D' = \{d'_i = d_i - d_m\}_{1,..,N}$$



$$M' = \{m'_i = m_i - c_m\}_{1,..,N}$$

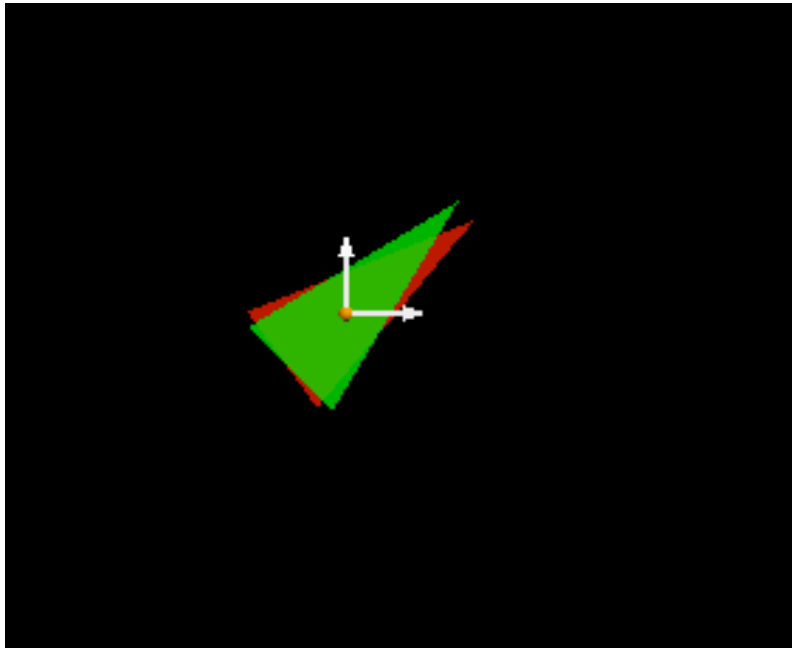


- De forma que el problema se reduce a minimizar la función de error:

$$E(R,t) \propto \sum_{i=1}^{N} \left\| m'_i - R d'_j \right\|^2$$

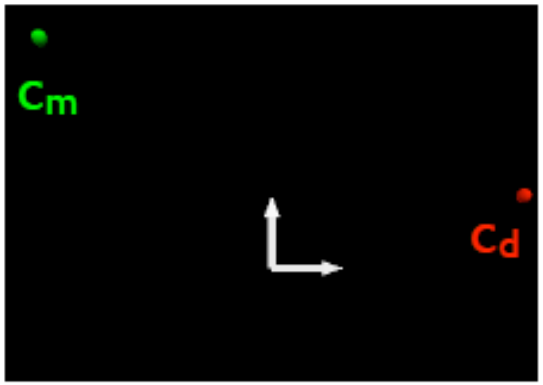
3. Métodos de registro 3D: ICP

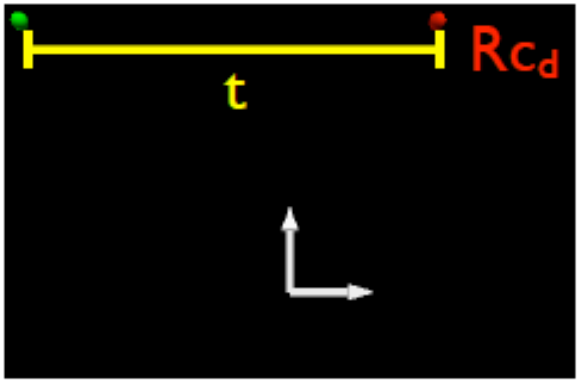
- Calcular la rotación R usando una solución en forma cerrada (por ejemplo SVD, Singular Value Decomposition, Descomposición en valores singulares)

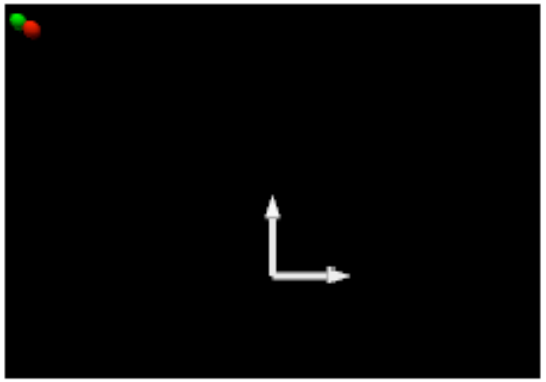


3. Métodos de registro 3D: ICP

- Calcular la traslación $t = c_m - Rc_d$







3. Métodos de registro 3D: ICP

- **Descomposición en Valores Singulares de una matriz (SVD)**
  - Descompone una matriz como producto de matrices ortogonales (direcciones, rotaciones) y diagonales (escalado)
  - Toda matriz H puede expresarse como:

$$H = U_{nxn} S_{nxm} V^T_{mxm}$$

    donde U_{nxr} y V^T_{mxr} son matrices ortogonales, esto es $U^T U = I_{nxn}$, $V^T V = I_{mxm}$
  - S es diagonal con r elementos positivos en la diagonal, siendo r el rango de la matriz y el resto cero
  - Los elementos de la diagonal S se denominan valores singulares de H
  - La columnas de U y V se denominan direcciones principales de salida y entrada, respectivamente

3. Métodos de registro 3D: ICP

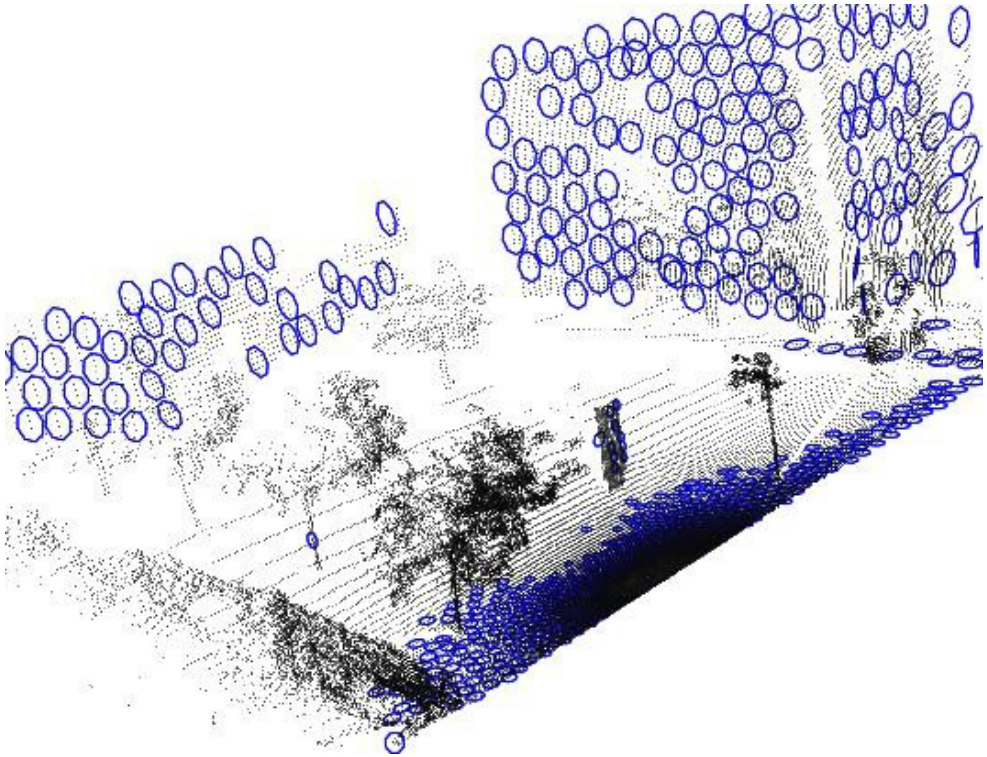
- Análisis:
  - Método iterativo: costoso. Cuello de botella en la búsqueda de emparejamientos. Solución: utilizar árboles de búsqueda KD
  - Minimiza el error de alineamiento
  - Muy sensible a outliers: puntos que están solamente en uno de los dos conjuntos
  - Puede extenderse a conjuntos de diferente tipo: punto-punto, punto-plano, plano-plano, punto-línea, punto-superficie, etc. Lo que necesitamos es definir la distancia entre ellos

3. Métodos de registro 3D: ICP

- ICP para robótica:
  - En robótica móvil, el uso de ICP presenta el inconveniente de la gran cantidad de outliers que suelen presentar los datos
  - Existen (y cada día aparecen nuevas soluciones) métodos para minimizar el impacto de los outliers
  - Además tenemos graves problemas con el cambio de escalas (frecuencia de muestreo de objetos a diferentes distancias)
  - En pocos casos se puede aplicar ICP sin conocer una transformación inicial aproximada pero, ¿qué ocurre si no disponemos de odometría?
  - Solución, utilizar información extra de las superficies que encontremos, por ejemplo, diferenciar las superficies planas

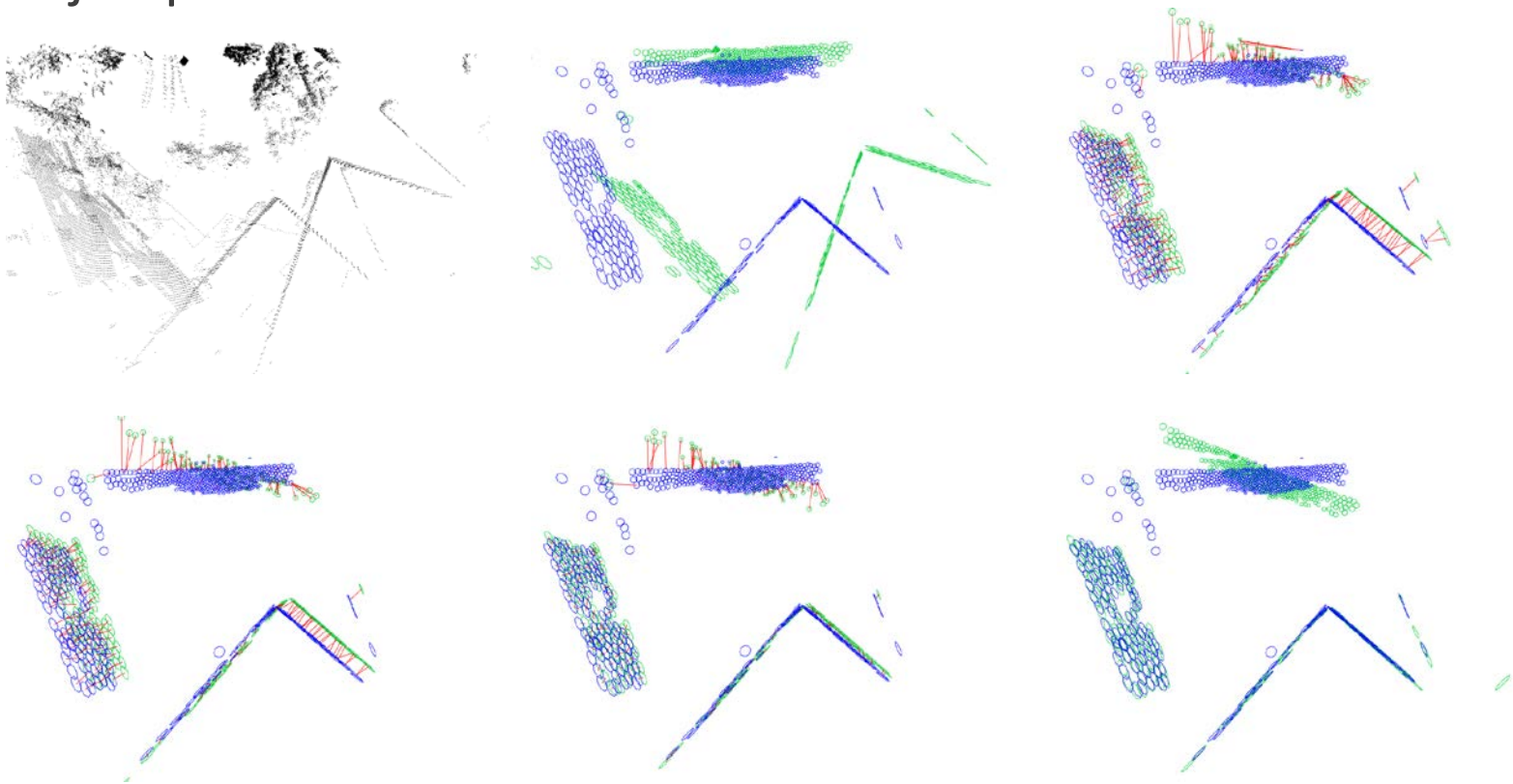
3. Métodos de registro 3D: ICP

- Otra opción es utilizar ICP en lugar de con puntos, con parches planares



3. Métodos de registro 3D: ICP

- Ejemplo:



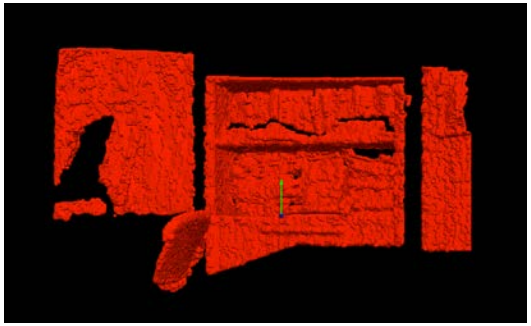
Índice de contenidos

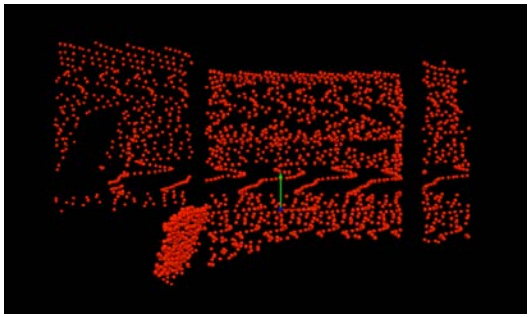
4. Variantes del ICP

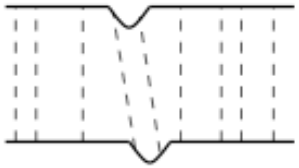
- Hay múltiples variantes que afectan a los distintos pasos del algoritmo:
  - **Seleccionar** puntos en uno o ambos conjuntos
  - **Emparejar** (matching) puntos entre los dos conjuntos
  - **Ponderar** las correspondencias entre puntos
  - **Eliminar** correspondencias incorrectas (outliers)
  - **Métrica** de error asociado a las correspondencias
  - **Minimizar** el error

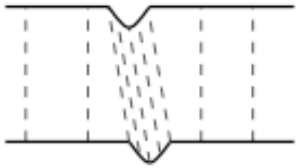
4. Variantes del ICP

- Selección de puntos:
  - Todos los puntos disponibles
  - Muestreo uniforme
  - Muestreo aleatorio con diferentes muestras en cada iteración
  - Con alto gradiente (color o intensidad)
  - Muestreo con la mayor distribución de normales
    - Muestreo uniforme en cada subespacio angular de normales
    - Facilita la convergencia en escenas con características pequeñas y dispersas



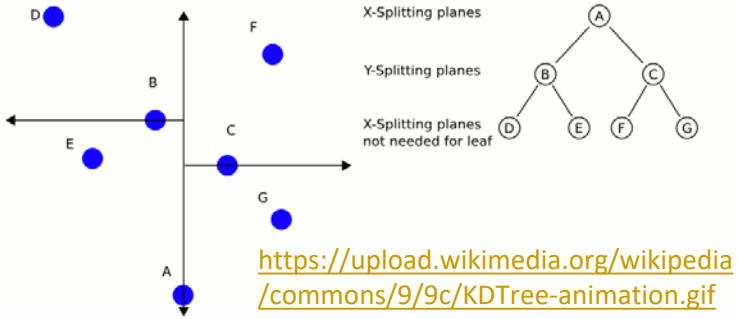




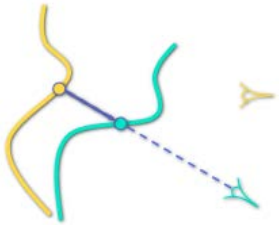


4. Variantes del ICP

- Emparejamiento de puntos:
  - Encontrar el punto más cercano en el otro conjunto (Closest Point). Para acelerar la búsqueda, se utiliza un Kd-tree
  - Encontrar la intersección del rayo en la dirección de la normal, con el conjunto modelo (Normal Shooting)
  - Proyectar el punto sobre el conjunto modelo desde el punto de vista de la cámara del conjunto modelo (Reverse Calibration)



https://upload.wikimedia.org/wikipedia/commons/9/9c/KDTree-animation.gif



4. Variantes del ICP

- Ponderación de correspondencias:
  - Pesos **constantes**
  - Asignar pesos en función de la **distancia** entre los puntos:

$$Weight = 1 - \frac{Disp(p_1, p_2)}{Dist_{max}}$$

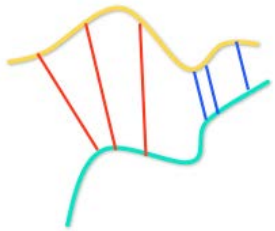
  - Ponderar en función de la "compatibilidad" de **normales**:

$$Weight = n_1 \cdot n_2$$

  - Ponderar en función del **color**
  - Ponderación basada en la **incertidumbre** de la medición (ruido del sensor)
    - Por ejemplo, con la Kinect las mediciones entre [60cm..1,5m] son más fiables que mediciones > 3m

4. Variantes del ICP

- Eliminación de correspondencias (outliers):
  - Eliminar correspondencias con distancia entre sus puntos superior a un determinado **umbral**

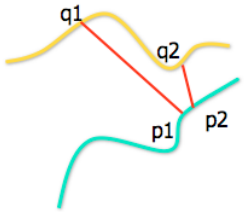


  - Eliminar un determinado porcentaje (por ejemplo 10%) de los peores emparejamientos, según alguna métrica (por ejemplo distancia entre puntos)
  - Eliminar emparejamientos cuya distancia entre sus puntos sea superior a un determinado múltiplo (por ejemplo, 2.5) de la desviación típica de las distancias

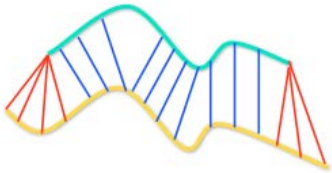
4. Variantes del ICP

- Eliminación de correspondencias (outliers):
  - Eliminar emparejamientos que no son consistentes con otros emparejamientos vecinos
    - Dos emparejamientos (p_1,q_1) y (p_2,q_2) son inconsistentes si

$$|Dist(p_1, p_2) - Dist(q_1, q_2)| < Umbral$$

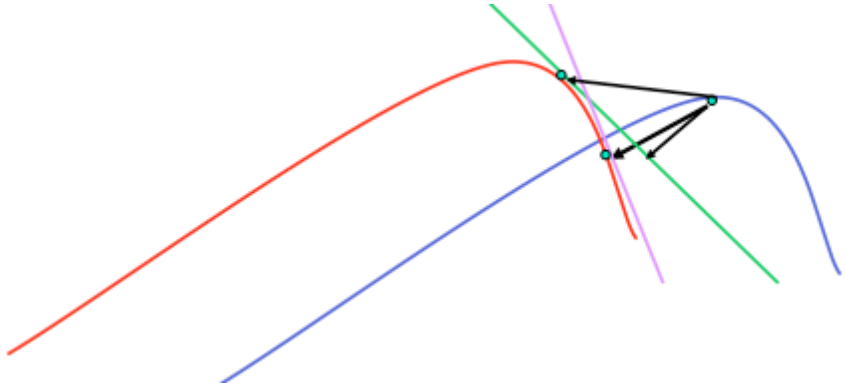


  - Eliminar correspondencias que contienen puntos en los límites del conjunto destino
    - Aplicable cuando el solapamiento entre los conjuntos no es completo



4. Variantes del ICP

- Medición del error de emparejamiento:
  - Métricas punto-a-punto que consideran la suma de distancias al cuadrado entre puntos correspondientes
    - También pueden tener en cuenta la diferencia de color
  - Distancia punto-al-plano que considera la suma de distancias al cuadrado entre cada punto con el plano más cercano (perpendicular a la normal del punto destino)



4. Variantes del ICP

- Minimización del error:
  - Generar **repetidamente** emparejamientos usando la transformación actual, y encontrar una nueva transformación que minimiza la métrica de error
  - Minimización iterativa combinada con **extrapolación** en el espacio de las transformaciones para acelerar la convergencia
  - Realizar la minimización iterativa con distintas **perturbaciones** en las condiciones iniciales, y seleccionar el mejor resultado
  - Realizar la minimización iterativa usando distintos subconjuntos de puntos seleccionados aleatoriamente, y elegir el resultado óptimo usando una métrica robusta (**least median of squares**)

Índice de contenidos

5. Cámara Kinect

- Comercializada desde noviembre de 2010
- Desarrollada por Primesense y utilizada por Microsoft para la consola Xbox 360
- Reconoce gestos, comandos de voz, objetos e imágenes
- Se ha extendido su uso a la comunidad científica
- APIs
  - OpenNI (Primesense)
  - SDK (Microsoft)
  - OpenKinect (comunidad abierta)

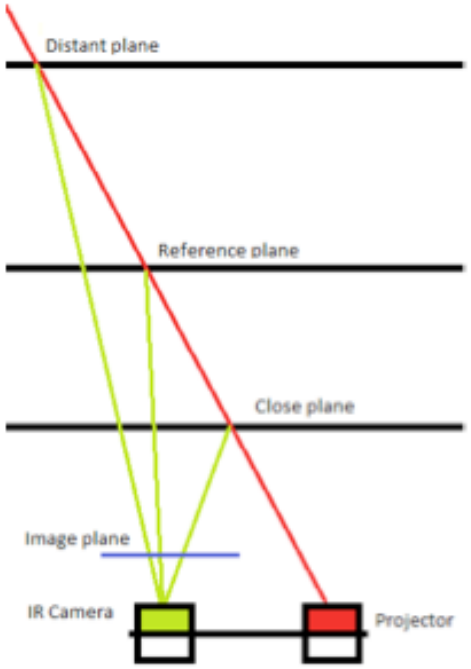
5. Cámara Kinect

- Sensores:

- Cámara RGB *(1)*
  - 32 bits de color
  - 640x480 pixels
  - 57º x 47º campo visual
  - 30 Hz
- Sensor de profundidad
  - proyector infrarrojo *(2)*
  - cámara de infrarrojos *(3)*
    - 16 bits de profundidad
    - 320x240
    - 30 Hz
  - 1,2 a 3,5 metros
- Motor de inclinación *(4)*
  - *Tilt* (hacia arriba o abajo) hasta 27º
- Micrófonos multiarray *(5)*
  - 4 canales a 16 bit de audio
  - 16 kHz

5. Cámara Kinect

- Cálculo de la profundidad:
  - Se proyecta un patrón de puntos con un láser infrarrojo
  - A partir de la luz infrarroja que llega rebotada a la cámara de infrarrojos se obtiene la profundidad a la que se encuentra cada pixel
  - El patrón de puntos está memorizado para una profundidad conocida. Al poner objetos delante, el patrón aparece distorsionado y así se puede calcular la profundidad
  - Como hay menos puntos en el patrón de IR que el mapa de profundidad, algunas partes de este mapa son interpoladas



https://youtu.be/uq9SEJxZiUg

https://www.freepatentsonline.com/20100118123.pdf

5. Cámara Kinect

- Ventajas:
  - Bajo coste (100€)
  - Proporciona 30 fps con resolución 640x480
  - Capaz de detectar profundidad en zonas sin textura
  - Puede funcionar a oscuras
- Inconvenientes:
  - Alcance limitado (50cm - 4m)
  - Falla en zonas que reflejan mal la luz infrarroja
  - No funciona en entornos de exterior
  - Suele fallar en los bordes de los objetos

Índice de contenidos

6. Point Cloud Library (PCL)

- Librería de procesamiento de nubes de puntos 3D
- Software de libre distribución (licencia BSD), desarrollado por la empresa Willow Garage
- Incluye algoritmos de:
  - Filtrado
  - Extracción de características
  - Reconstrucción de superficies
  - Alineamiento
  - Ajustes de modelo
  - Segmentación
  - Visualización
- Implementada en C++
- Plataformas de desarrollo Linux, MacOS, Windows y Android
- Soporta de forma nativa las interfaces 3D de OpenNI

6. Point Cloud Library (PCL)

- Estructuras básicas
- Tipo **PointT**
  - Existen distintos tipos de puntos predefinidos:
    - PointXYZ, PointXYZRGB, PointNormal, etc.
  - Se pueden añadir tipos definidos por el usuario

    http://pointclouds.org/documentation/tutorials/adding_custom_ptype.php#adding-custom-ptype
- Tipo **PointCloud**
  - Permite almacenar una nube de puntos
    - int width
    - int height
    - std::vector<PointT> points
  - Nube de puntos organizada (height = número de filas) o desorganizada (height = 1)

6. Point Cloud Library (PCL)

- Tipo **PointCloud**, ejemplo:

```
pcl::PointCloud<pcl::PointXYZ> cloud;
std::vector<pcl::PointXYZ> data = cloud.points;
// Nube organizada
cloud.width = 640; cloud.height = 480;
// Nube desorganizada
cloud.width = 307200; cloud.height = 1;
```

http://pointclouds.org/documentation/tutorials/

6. Point Cloud Library (PCL)

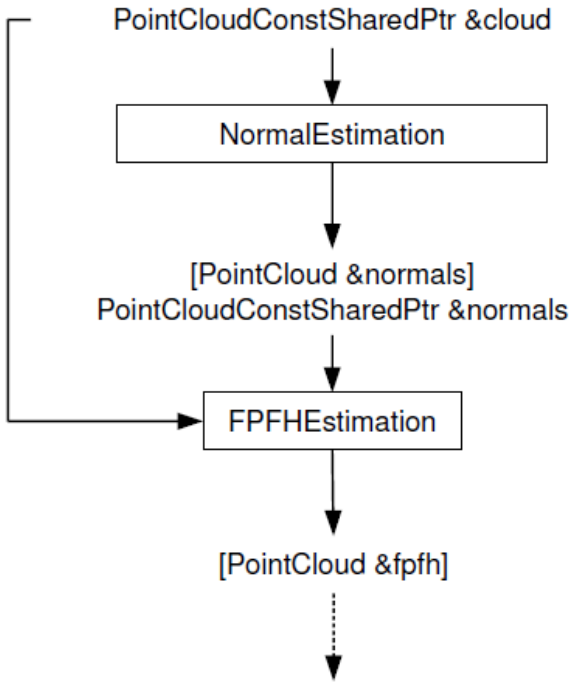
- Librerías de PCL:
  - **libpcl_filters**
    - Filtros de datos: reducción de la resolución (downsampling), eliminación de outliers, etc.
  - **libpcl_features**
    - Extracción de características 3D: normales de superficies, puntos frontera, descriptores SIFT, NARF…
  - **libpcl_io**
    - Entrada y Salida: escribir y leer ficheros en formato PCD (Point Cloud Data)
  - **libpcl_segmentation**
    - Segmentación: extracción de clusters, ajustes de modelo mediante métodos de consenso, etc.

6. Point Cloud Library (PCL)

- Librerías de PCL:
  - **libpcl_surface**
    - Reconstrucción de superficies
  - **libpcl_registration**
    - ICP
  - **libpcl_keypoints**
    - Extracción de puntos clave que puede utilizarse para decidir dónde extraer descriptores de características
  - **libpcl_range_image**
    - Dar soporte a mapas de profundidad creados a partir de nubes de puntos

6. Point Cloud Library (PCL)

- Procesamiento pipeline en PCL:
  1. Crear el objeto de procesamiento (filtro, estimador de características, segmentación, etc)
  2. Utilizar *setInputCloud* para pasar la nube de puntos de entrada al módulo de procesamiento
  3. Actualizar algunos parámetros
  4. Invocar al correspondiente módulo de procesamiento (*compute*, *filter*, *segment*, etc) para obtener el resultado



Índice de contenidos

7. Ejemplo real

- Proyecto KinectFusion
  - Un proyecto para la Kinect que fusiona el mundo real y el virtual
  - Generar modelos 3D en tiempo real: objetos, personas y habitaciones enteras
  - Motor de física realista que permite escanear objetos para ser manipulados con realismo

https://youtu.be/quGhaggn3cQ