Detección y 'Matching' de objetos 3D en una nube de puntos

Calatayud Ferre, Raúl
Pérez Vilaplana, Ignacio

Mayo 2021

Resumen

En el siguiente documento se va a encontrar una implementación para tratar de detectar ciertos objetos en una nube de puntos. Para ello se ha facilitado una escena y una serie de objetos (una hucha, una taza, una planta y un PLC) en formato *pcd*. El objetivo es ser capaz de visualizar esta PCL en pantalla y realizar el proceso necesario para destacar estos objetos en la escena. En primer lugar se plantean los problemas que supone esto así como la lógica que se va a seguir. A continuación se discutirá sobre las diferentes formas de resolverlos aplicando y discutiendo sobre los métodos de la **PCLibrary** aplicados [4]. El código, gráficas e imágenes, nubes de puntos, enunciados de la práctica y esta misma memoria están incluidos en un repositorio de GitHub [1] que se ha ido actualizando durante el proceso de investigación.

Índice

Índice de figuras

Índice de cuadros

1. Introducción

1.1. Contexto

PCL es una librería de C++ desarrollada para el tratamiento de nubes de puntos en N-dimensiones desarrollado por Willow Garage [3] con el objetivo de realizar procesamiento con numerosas técnicas. Esta librería tiene algoritmos para aplicar filtros, estimación de funciones, reconstrucción de superficies, ajustes de modelos, segmentación, entre otros. Además se encuentra liberada bajo licencia BSD, es decir, que es libre para uso comercial e investigativo, y sus códigos fueron desarrollados en lenguaje de programación C++. Se ha recurrido a información [2] y a [5] para completar toda la teoría de la primera parte de esta práctica.

1.2. Estructura del documento

Este documento tiene dos partes claramente diferenciadas e importantes. En un primer lugar se plantearán los problemas y se irán explicando paso a paso los puntos de la práctica. En una segunda parte se aplicarán diferentes métodos y se compararán entre ellos para obtener el mejor de los resultados.

Por otro lado, se ha comentado ya que existe un repositorio en GitHub [1] con todos los archivos. Al descargarlo debería obtener una carpeta con la estructura de archivos que aparece en la Figura [1].

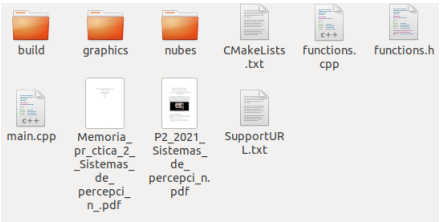


Figura 1: Estructura de los archivos

Para compilarlo tan solo debe acceder desde el terminal a la carpeta, abrir el directorio **build** y ejecutar el comando *make*. Una vez compilado, desde ese mismo directorio se debe introducir la instrucción *./main* para ejecutarlo.

Como se puede ver, existen otras dos carpetas. En **graphics** están guardadas las tablas de comparación de tiempo de ejecución así como las imágenes que se incluyen en este documento. Por otro lado, en la carpeta **nubes** se encuentran los archivos pcd tanto de la escena como de los objetos.

Por último, se incluyen los *.pdf* y *.txt* con esta misma memoria, el enunciado de la práctica y un documento con las direcciones *url* de la documentación buscada.

2. Estado del arte: Descripción de los problemas y puntos planteados

Los diferentes problemas o puntos a abordar en la implementación del pipeline de reconocimiento de objetos en una escena 3D van a ser expuestos en esta sección. Toda, o casi toda la documentación necesaria para implementar los métodos de la PCL se ha encontrado en su pagina web [4]. Los diferentes módulos que se pueden emplear se pueden ver en la Figura [2]

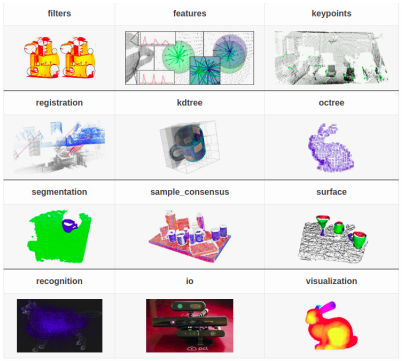


Figura 2: Módulos librería PCL

2.1. Eliminación de los planos dominantes de la escena

Se pretenden eliminar planos y superficies interiores comunes, como paredes, etc, ... Para ello, el módulo de PCL que encaja con estos requisitos es **Segmentation**.

En la Figura [3] se pueden ver los diferentes planos como la pared del fondo y la lateral, las cuáles se deben eliminar, de manera que se queden solo los puntos más relevantes (los pertenecientes a los objetos y alrededores).

2.2. Extracción de *keypoints*

2.2.1. SIFT 3D (Scale Inveriante Feature Transform)

El método SIFT busca puntos característicos a partir de las derivadas parciales para sacar las zonas con mayor varianza en su vecindad y los hace invariantes a la escala, rotación, cambios de iluminación, ruido y pequeños cambios de pose o perspectiva.

2.2.2. SUSAN

Es un detector de keypoints a bajo nivel, se escogen una vecindad esférica y los puntos que quedan dentro de ella se someten a un umbral de iluminación. Los puntos que pasan el corte se agregan al área USAN del núcleo.

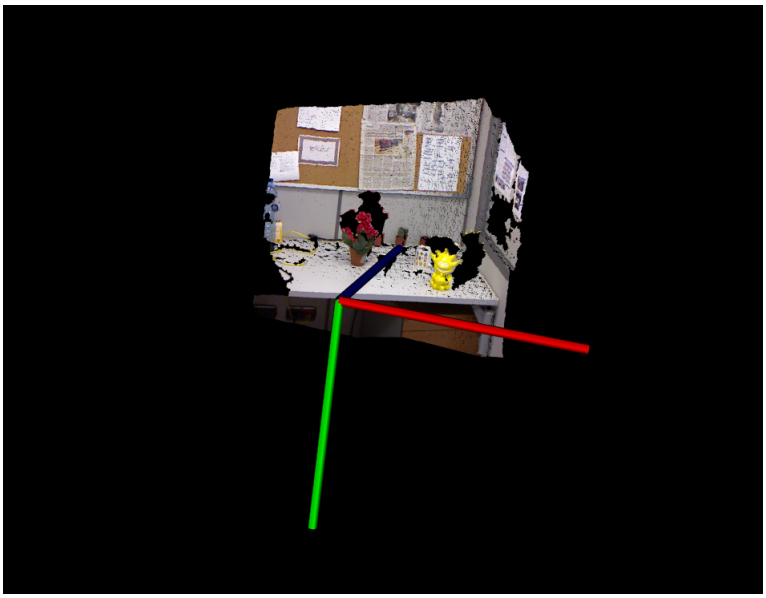


Figura 3: Escena inicial

2.2.3. HARRIS 3D

Se cambian los gradientes de intensidad por el cálculo de las normales en cada uno de los puntos. Este cálculo devuelve la variabilidad del punto con los 3 ejes principales. Se escogen los puntos con la mayor varianza y se someten a un umbral para evitar que haya demasiados. También se divide la figura en clústers y en cada uno de ellos se escoge el punto más significativo.

2.2.4. ISS (Instrinsic Shape Signature)

Este algoritmo realiza una búsqueda a través de la superfície de la figura y selecciona los puntos que tienen una gran variación en la dirección principal

2.2.5. NARF (Normal Aligned Radial Feature)

NARF extrae los keypoints en áreaas donde la superficie subyacente directa es estable y el vecindario contiene cambios importantes en su superfície, por tanto éste método hace que los keypoints se situen en el entorno local de las formas geométricas significativas y no sobre ellas. Éste método tiene muy en cuenta los bordes de los objetos, por tanto la silueta de la figura es muy importante para el cálculo de los keypoints resultantes.

2.2.6. UNIFORM SAMPLING

El algoritmo crea un voxel grid sobre la nube de puntos y calcula el centroide de los puntos que quedan dentro de cada casilla.

2.3. Extracción de descriptores de *keypoints*

2.3.1. PFH (Point Feature Histogram)

Obtiene información de la geometría calculanod las diferencias entre las normales. Se busca una vecindad esférica y se relacionan el punto central con los vecinos y los vecinos entre ellos. Para cada par de puntos se calculan los ángulos entre los vectores de la normal de cada punto. Estas variables angulares se guardan cada una en un histograma y en un cuarto histograma se incluye la distancia euclídea entre los puntos.

2.3.2. FPFH (Fast Point Feature Histogram)

Los cálculos són los mismos que en el método anterior, ssolo que en este en vez de relacionar todos los vecinos entre ellos, solo se unen con los puntos que queden dentro de una vecindad esférica del mismo radio que la vecindad inicial.

2.3.3. 3D Shape Context

Se crea una malla esférica alrededor del punto y se divide en sectores de forma equitativa a lo largo de las dos variables angulares(azimut e inclinación), se calculan los puntos que caen dentro de cada sector y cada punto se multiplica por un peso inversamente proporcional a la densidad local del punto y el volumen del sector.El vector de características contendrá cada uno de los sectores y cada parte del histograma se asociará con un valor y calculada como la suma ponderada de los puntos de ese sector. La normal del punto central se alinea con el "polo norte", no obstante no se conocen la posición de los otros dos ejes, lo que provoca que cualquier posición de la esfera sea válida.

2.3.4. Unique Shape Context

Este método es igual que el 3D Shape Context pero haciendo un análisis para ver como evolucionan los puntos a lo largo de esa superfície y acotando los ejes. El resto de puntos sacan los 2 ejes principales con PCA, siendo el único inconveniente que la normal puede ser tanto positiva como negativa. Para intentar solucionar esto, se cogen las 4 posibles combinaciones de los ejes y así en vez de tener todas las posibles combinaciones, se tienen solo 4.

2.3.5. SHOT (Signature of Histograms of OrienTations)

Se crea una vecindad esférica y se divide en sectores del mismo tamaño y para cada uno de ellos se crea un histograma unidimensional en el que se guarda el ángulo entre la normal de los puntos de un sector con la normal que tiene

el punto que se está estudiando. Luego se comparan las normales de un sector hasta que quede solo una y se mete en un vector descriptor. Para mejorar la robustez ante la densidad de la nube de puntos, se normaliza el descriptor entre el número de puntos de la nube de puntos vecinos.

2.3.6. VFH (Viewpoint Feature Histogram)

Este método se parece al FPFH solo que no se calcula para cada uno de los puntos, sino que se coge el punto central y se eligen el resto de puntos como vecinos. También se coge información del punto de vista desde el que se ha tomado para hacerlo invariante a la posición del objeto

2.3.7. CVFH (Clustered Viewpoint Feature Histogram)

Se divide el objeto en trozos característicos para sacar el descriptor de cada uno y así luego con que se encuentre uno solo ya se considera que se ha encontrado el objeto. De está manera se mejora la robustez ante las oclusiones.

2.3.8. ESF (Ensemble of Shape Functions)

El ESF usa una combinación de 3 funciones de forma: distancias, ángulos y áreas entre puntos. Es robusto ante ruido y superficies incompletas. Primero aplica un voxel grid para encajar la figura en un cubo y empieza a realizar iteraciones. En cada iteración se escogen 3 puntos al azar y a partir de ellos calcula una serie de funciones: Calcula la distancia entre los puntos, el área del triangulo que forman y los ángulos entre ellos. De esta forma crea una serie de histogramas dependiendo de si las líneas y áreas anteriores caen dentro de la figura, fuera o mezclado si caen una parte dentro y otra fuera.

2.4. Emparejamientos de los descriptores de la escena y del objeto

Para realizar los emparejamientos entre los distintos descriptores se suele usar algún tipo de distancia entre los keypoints, normalmente es la euclídea. Se comparan dos conjuntos de descriptores, y devuelve los keypoints emparejados, es decir a que punto de una nube se parece mas un punto de la otra. Para esto se suele meter los puntos de la nube más grande en un KdTree y luego se iteran los puntos de la nube más pequeña y se busca el vecino más cercano usando el KdTree. Esto suele devolver emparejamientos buenos y malos, y estos se deben filtrar para eliminar los incorrectos.

2.5. Eliminación de los emparejamientos incorrectos

Para el filtrado y eliminación de los emparejamientos incorrectos en este caso se ha usado RANSAC de la siguiente forma: Se cogen 3 pares de correspondencias aleatorias y se calcula la transformación entre ellas para hacer cuadrar las nubes de puntos y luego se evalúan la cantidad de inliers en un umbral. Este

proceso se realiza n veces y se elige la que mayor número de inliers tiene, es decir, la mejor transformación. Este proceso puede que haga un encuadre bastante preciso de las nubes de puntos, pero es muy difícil que sea la óptima ya que los 3 pares de correspondencias elegidas son aleatorias.

2.6. Cálculo de la transformación entre la PC del objeto en la escena y del objeto

RANSAC ya devuelve una matriz de transformación con rotación y traslación para encajar la nube de puntos del objeto en la nube de puntos de la escena, aunque la transformación no sea la óptima, para el ojo humano puede resultar en un encuadre perfecto o muy preciso.

2.7. Refinamiento de los resultados

Ya que la transformación de RANSAC seguramente no sea la óptima, se ejecuta un refinamiento de la transformación para hacerla lo mas precisa posible, para ello se usa el Iterative Closest Point, que se debe aplicar sobre una transformación ya de por si bastante buena. Su funcionamiento se basa en encontrar el punto más cercano minimizando la distancia entre dos formas y se corrige, teniendo un error resultante. Este proceso se realiza en varias iteraciones hasta que el error resultante baja de un umbral y se elige esa transformación como la definitiva.

3. Experimentación. Descripción y comparación de los métodos empleados para encontrar objetos en la escena

En secciones anteriores se han planteado los problemas que supone cada paso del pipeline. En esta parte se compararan los diferentes métodos para cada punto y se justificará el porqué se ha acabado eligiendo uno u otro. Por supuesto también se van a enumerar los pasos seguidos y se va a hacer una explicación (superficial) del código. Con esto se pretende que el lector sea capaz de entender la lógica que sigue no solo el código de implementación si no también los datos que va a ver a continuación.

3.1. Explicación código para implementar el pipeline

La detección de los objetos dentro de la nube de puntos de la escena pasa por muchos procesos antes de poder visualizar la representación final. El pipeline que se ha seguido es:

1. **Eliminación de los planos dominantes**
2. **Selección de método de extracción de Keypoints y descriptores sobre un objeto concreto a través de una interfaz**
3. **Extraer los Keypoints de la nube de puntos d la escena y del objeto**
4. **Calcular los descriptores de los Keypoints extraídos**
5. **Realizar los emparejamientos entre los Keypoints del objeto con los de la escena**
6. **Eliminar los emparejamientos incorrectos**
7. **Calcular la transformación entre los emparejamientos de la nube de puntos del objeto con la de la escena**
8. **Refinar la transformación calculada para hacerla lo más precisa posible**

3.1.1. Eliminación de planos

Para la *Eliminación de planos* se ha recurrido a la clase **SACSegmentation** de **PCL** para eliminar las paredes, mesa y demás superficies grandes. En el uso de esta clase se pueden seleccionar modelos de superficie, esferas, conos, planos en horizontal y vertical, lineas, etc... También se puede elegir el método empleado para la localización de estas superficies y el *threshold*, que es la distancia al umbral del modelo. Se han hecho pruebas con un modelos de plano de tipo:

- SACMODEL_PLANE

- SACMODEL_NORMAL_PLANE
- SACMODEL_NORMAL_PARALLEL_PLANE

Así pues, también se ha modificado el *threshold*, entre valroes de 0.01 y 0.05. La selección de parámetros elegidos para eliminar los planos y que se podrá ver en los diferentes experimentos ha sido la combinación entre el uso del **SAMODEL_PLANE** y el *Threshold* de valores 0.012 y 0.05. Tras realizar algunas pruebas se ha concluido que la eliminación funciona mejor con estos valores.

Esto proceso al ejecutarlo una vez, eliminaría el plano de mayor superficie que encuentre. Sin embargo, lo que se busca es quedarse con la menor cantidad de superficie (ajena a los objetos) posibles. Por tanto, esto se hace en bucle hasta que se obtiene una nube de puntos más reducida. Se han hecho pruebas reduciendo la nube de puntos hasta un 60 % como se puede ver en la Figura [4] y hasta un 50 % en la Figura [5]. Con cada uno de los porcentajes se han probado los dos valores de *Threshold* comentados. Con porcentajes inferiores no se han obtenido buenos resultados a la hora de calcular KeyPoints, descriptores, etc, ... Por lo tanto, se ha descartado comprobar combinaciones con valores más pequeños.

(a) Eliminación de planos en un 60 % con *Threshold* a 0.05

(b) Eliminación de planos en un 60 % con *Threshold* a 0.012

Figura 4: Eliminación de planos en un 60 %

(a) Eliminación de planos en un 50 % con *Threshold* a 0.05

(b) Eliminación de planos en un 50 % con *Threshold* a 0.012

Figura 5: Eliminación de planos en un 50 %

Como se puede apreciar, los mejores resultados se obtienen con un 50 % de puntos restantes tras la eliminación de los planos. Todas las pruebas que se han

hecho a continuación están basadas en la premisa de que este es el porcentaje que, para el caso de este código, mejor funciona.

3.1.2. Interfaz de selección

En cuanto a la interfaz para seleccionar el método de extracción de Keypoints y descriptores sobre un objeto concreto, se tiene, que se ha implementado menú con 4 opciones iniciales para la selección de un objeto, con otras cuatro opciones a posteriori para escoger el método de extracción de las características y en función de este método, muestra los métodos de descriptores disponibles. Tras la elección del objeto, se carga su nube de puntos junto a la de la escena cargando el archivo **pcd** correspondiente dentro de la carpeta **nubes** del directorio. Los métodos de extracción de keypoints disponibles son **HARRIS**, **SIFT**, **ISS** y **Uniform Sampling**. Para todos estos métodos se puede escoger entre 4 modelos de descriptor. Los disponibles son **PFH**, **FPFH**, **SHOT** y**RSD**, no obstante, para el detector **US** el último descriptor no se implementa.

3.1.3. Extracción de keypoints y descriptores

En cuánto al método de extracción de keypoints. Cada uno de ellos los calcula utilizando un tipo de punto diferente, para crear una *PointCloud*. **PCL** tiene una serie de tipos de puntos para nubes ya declarado. Cada algoritmo trabaja con un tipo diferente. El método **HARRIS** utiliza el tipo **PointXYZI**, mientras que por ejemplo el método **SIFT** utiliza el **PointWithScale**. Estas nubes de puntos calculadas serán las que utilicen más tarde los diferentes métodos de descriptores. De esta manera se requiere que el descriptor reciba con que tipo de punto se ha creado la nube de características. Para ello se intentó crear una función generalizada utilizando *templates* de C++, no obstante la dificultad de programación no permitió que se usara este recurso. Por lo tanto, se ha creado una función que ejecuta cada combinación posible. Teniendo una para **HARRIS** con **PFH**, con, **FPFH**, y así para todos los métodos. El grueso de código escrito es mucho mayor, pero su implementación es mucho más sencilla.

3.1.4. Emparejamientos, rechazo de incorrectos y refinamiento de la matriz de transformación final

Al igual que pasa en el apartado anterior, el método de búsqueda para los emparejamiento también esta sujeto al tipo de nube de puntos de los descriptores. Por lo tanto no se ha podido generalizar con una función si no que se ha incluido en el mismo algoritmo en el que se hace el cálculo de características. Para los emparejamientos iniciales se utiliza la clase **Correspondence** de **PCL** y se le pasa la nube de descriptores.

Más adelante para rechazar los emparejamientos incorrectos se aplica el algoritmo **RANSAC**. La implementación de este en **PCL** se facilita mediante la clase **CorrespondenceRejectorSampleConsensus**, y se le debe indicar el tipo de keypoints utilizado.

Tras rechazar los emparejamientos incorrectos, existe un método de esta clase que proporciona la matriz de traslación-rotación del objeto sobre la escena. Esta será la primera transformación sin refinar que e tendrá.

Una vez aplicada esta transformación sobre el objeto, se le pasará al programa un algoritmo de **Iterative Closest Point**, de manera que se ajusten todos los puntos que sean similares entre la escena y el objeto. Una vez calculados la propia clase, permite realizar una nueva transformación del objeto sobre la escena con esta traslación y rotación actualizadas.

3.2. Experimentos realizados

La importancia del tipo de punto del keypoint, junto a que después de realizar distintas pruebas con el código, se ha comprobado que la mayor diferencia entre un experimento y otro radica en el tipo de detector de keypoints utilizado, los experimentos realizados a partir de este punto se hacen a partir del método de detección de características. Dentro de cada método se mostrarán los resultados de las pruebas con diferentes umbrales en la eliminación de planos y con los distintos descriptores. También, para cada detector se han hecho 4 pruebas distintas, una para cada objeto, ya que el tiempo y los keypoints detectados varía mucho en función de la nube de puntos del objeto que se carga, ya que, por ejemplo, la planta es mucho más grande que el PLC y por tanto contendrá muchos mas keypoints y el procesamiento que necesitará será también más alto.

A continuación se pueden tablas con todas las combinaciones que se han llevado a cabo. Para cada modelo implementado se han tomado de medidas de tiempo para la eliminación de planos, así como para la duración del proceso de detección de keypoints + descriptores + emparejamientos y rechazo de los incorrectos + ICP.

3.2.1. Experimentos con HARRIS

Para el detector HARRIS se puede observar con respecto al tiempo de procesamiento que en este método el modelo de descriptores es el que marca la duración final. Esto se puede observar tanto en las tablas para los cuatro objetos como en la Figura [6]. Aún así se mantiene de forma bastante regular. No varia al cambia el *Threshold* en la eliminación de planos, por lo cual se puede interpretar que en este modelo el *Threshold* más óptimo será el de valor 0.012, ya que no tiene mejoras substanciales con respecto al número de extracción de keypoints y descriptores, y sin embargo es más rápido.

Se remarca que para este extractor, se han obtenido errores al aplicarlo sobre los descriptores **SHOT** sobre la **HUCHA** y el **PLC**.

En conclusión ninguno de los resultados obtenidos ha sido satisfactorio, ya que al mostrarlos por pantalla, la transformación que se hacia sobre el objeto estaba muy alejada del objeto de la escena.

	HARRIS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Buenos	%
TAZA	0,05	PFH	8977	3148	23	23	3	0,00128
	0,012	PFH	5179	3148	23	23	3	0,00128
	0,05	FPFH	5369	3148	23	23	6	0,04555
	0,012	FPFH	4623	3148	23	23	6	0,00276
	0,05	RSD	4870	3148	23	23	5	0,04555
	0,012	RSD	4602	3148	23	23	6	0,00031
	0,05	SHOT	5233	3148	23	23	23	0,04555
	0,012	SHOT	5187	3148	23	23	23	0,00281

Cuadro 1: Método HARRIS para TAZA

	HARRIS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
HUCHA	0,05	PFH	5619	3148	39	39	3	0,00044
	0,012	PFH	5100	3148	39	39	3	0,00044
	0,05	FPFH	5262	3148	39	39	7	0,00848
	0,012	FPFH	5182	3148	39	39	7	0,00075
	0,05	RSD	6105	3148	39	39	4	0,00824
	0,012	RSD	4971	3148	39	39	4	0,00099
	0,05	SHOT	X	X	X	X	X	X
	0,012	SHOT	X	X	X	X	X	X

Cuadro 2: Método HARRIS para HUCHA

	HARRIS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLANTA	0,05	PFH	5486	3148	61	61	3	0,0014
	0,012	PFH	5462	3148	61	61	3	0,0014
	0,05	FPFH	4898	3148	61	61	10	0,02145
	0,012	FPFH	5114	3148	61	61	10	0,00164
	0,05	RSD	5882	3148	61	61	5	0,02762
	0,012	RSD	5356	3148	61	61	5	0,003
	0,05	SHOT	5384	3148	61	61	4	0,00211
	0,012	SHOT	5779	3148	61	61	4	0,01263

Cuadro 3: Método HARRIS para PLANTA

	HARRIS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLC	0,05	PFH	5006	3148	5	5	5	0,4028
	0,012	PFH	5008	3148	5	5	5	0,4028
	0,05	FPFH	4553	3148	5	5	3	0,00017
	0,012	FPFH	4592	3148	5	5	3	0,00588
	0,05	RSD	4675	3148	5	5	5	0,41932
	0,012	RSD	4613	3148	5	5	5	0,07661
	0,05	SHOT	X	X	X	X	X	X
	0,012	SHOT	X	X	X	X	X	X

Cuadro 4: Método HARRIS para PLC

3.2.2. Experimentos SIFT3D

Tras realizar el experimento con SIFT3D queda evidenciado que el tiempo esta marcado por los descriptores. Se puede observar en la Figura [7] que se mantiene similitud con respecto a la de Harris, pero incrementando el coste computacional. Esto solo se debe a que la cantidad de puntos característicos extraída para la escena y el objeto es mucho mayor. Por lo tanto, el coste computacional más elevado queda justificado por el descriptor a de procesar un número mayor de datos.

La cantidad de tiempo de procesamiento queda justificada viendo los resultados obtenidos. En las tablas se puede ver como los emparejamientos correctos extraídos son mayores que en cualquier otro método.

Por otro lado, se ha obtenido buenos resultados de precisión para todos los objetos. Aunque, por supuesto en algunos casos estos resultados son más precisos con otros objetos, destaca mucho la precisión obtenida en la hucha y en la planta.

3.2.3. Experimentos ISS

El método ISS es el más lento de todos los modelos implementados. Extrae una cantidad de Keypoints muy inferior la extraída en ISS, con un coste computacional mucho mayor. Sin embargo, se destaca que obtiene un número de KeyPoints mucho mayor en la escena que en objetos, por lo que se deduce que para objetos más grandes, funciona mejor.

Se han obtenido buenos resultados de este modelo para casi todos los descriptores (exceptuando el SHOT).

3.2.4. Experimentos Uniform Sampling

De este modelo destaca la rapidez de procesamiento y que la cantidad de puntos extraídos es similar a la del modelo **ISS**. Pero, la cantidad de empareja-

	SIFT							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
TAZA	0,05	PFH	18472	11855	510	510	133	0,0003
	0,012	PFH	16944	13876	510	510	112	0,00031
	0,05	FPFH	9800	11855	510	510	162	0,00094
	0,012	FPFH	9322	13876	510	510	165	0,0004
	0,05	RSD	10366	11855	510	510	37	0,00038
	0,012	RSD	9423	13876	510	510	38	0,00022
	0,05	SHOT	9391	11855	510	X	X	X
	0,012	SHOT	8825	13876	510	X	X	X

Cuadro 5: Método SIFT para taza

	SIFT							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
HUCHA	0,05	PFH	22300	11855	1490	1490	234	0,00037
	0,012	PFH	19763	13876	1490	1490	298	0,001
	0,05	FPFH	11609	11855	1490	1490	393	0,00069
	0,012	FPFH	11158	13876	1490	1490	351	0,00044
	0,05	RSD	11409	11855	1490	1490	96	0,00073
	0,012	RSD	10731	13876	1490	1490	67	0,00074
	0,05	SHOT	10717	11855	1490	1490	393	0,00069
	0,012	SHOT	9635	13876	1490	X	X	X

Cuadro 6: Método SIFT para HUCHA

	SIFT							
	Treshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLANTA	0,05	PFH	21018	11855	1963	1963	432	0,00016
	0,012	PFH	19367	13876	1963	1963	472	0,00066
	0,05	FPFH	10707	11855	1963	1963	412	1E-05
	0,012	FPFH	10609	13876	1963	1963	440	1E-05
	0,05	RSD	11805	11855	1963	1963	81	0,00096
	0,012	RSD	11201	13876	1963	1963	126	0,0001
	0,05	SHOT	10306	11855	1963	X	X	X
	0,012	SHOT	102027	13876	1963	X	X	X

Cuadro 7: Método SIFT para PLANTA

	SIFT							
	Treshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLC	0,05	PFH	17527	11855	194	194	86	0,00029
	0,012	PFH	16238	13876	194	194	107	0,00013
	0,05	FPFH	8964	11855	194	194	87	6E-05
	0,012	FPFH	9083	13876	194	194	102	0,00188
	0,05	RSD	8986	11855	194	194	29	0,00013
	0,012	RSD	9138	13876	194	194	17	6E-05
	0,05	SHOT	8680	11855	194	X	X	X
	0,012	SHOT	8697	13876	194	X	X	X

Cuadro 8: Método SIFT para PLC

	ISS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
TAZA	0,05	PFH	2295	2353	24	24	6	0,00046
	0,012	PFH	2081	1938	24	24	4	0,00039
	0,05	FPFH	2422	2353	24	24	6	0,01123
	0,012	FPFH	2072	1938	24	24	12	0,00015
	0,05	RSD	2673	2353	24	24	5	0,00073
	0,012	RSD	2379	1938	24	24	4	0,00038
	0,05	SHOT	2684	2353	24	24	4	0,00041
	0,012	SHOT	2224	1938	24	24	4	0,00017

Cuadro 9: Método ISS para TAZA

	ISS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
HUCHA	0,05	PFH	2489	2353	84	84	15	0,00084
	0,012	PFH	2601	1938	84	84	15	0,00029
	0,05	FPFH	2467	2353	84	84	30	0,00059
	0,012	FPFH	2538	1938	84	84	9	0,00058
	0,05	RSD	2298	2353	84	84	7	0,00027
	0,012	RSD	2500	1938	84	84	5	0,0004
	0,05	SHOT	2846	2353	84	84	23	0,00298
	0,012	SHOT	3251	1938	84	84	14	0,00063

Cuadro 10: Método ISS para HUCHA

	ISS							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLANTA	0,05	PFH	2534	2353	111	111	16	0,0001
	0,012	PFH	2755	1938	111	111	6	0,00153
	0,05	FPFH	2414	2353	111	111	13	3E-05
	0,012	FPFH	3219	1938	111	111	17	0,00115
	0,05	RSD	2505	2353	111	111	6	0,00085
	0,012	RSD	2739	1938	111	111	4	0,00135
	0,05	SHOT	3018	2353	111	111	12	0,02439
	0,012	SHOT	3195	1938	111	111	7	0,00116

Cuadro 11: Método ISS para PLANTA

	ISS							
OBJETO	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLC	0,05	PFH	2004	2353	11	11	5	7E-05
	0,012	PFH	2048	1938	11	11	9	7E-05
	0,05	FPFH	1980	2353	11	11	6	7E-05
	0,012	FPFH	2074	1938	11	11	11	0,00018
	0,05	RSD	2136	2353	11	11	4	0,00131
	0,012	RSD	2090	1938	11	11	3	0,00012
	0,05	SHOT	2048	2353	11	11	9	0,00016
	0,012	SHOT	2072	1938	11	11	3	0,00012

Cuadro 12: Método ISS para PLC

mientos correctos es menor y muestra un accuracy peor de forma general.

Cabe destacar que para este modelo no se ha podido implementar adecuadamente el descriptor **RSD** ni **SHOT**. También destacar que se ha obtenido un muy buen resultado con la hucha.

	US							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
TAZA	0,05	PFH	463	1481	32	32	4	0,00093
	0,012	PFH	133	1939	32	32	6	0,00074
	0,05	FPFH	112	1481	32	32	7	0,00011
	0,012	FPFH	162	1939	32	32	4	0,00033
	0,05	RSD	X	X	X	X	X	X
	0,012	RSD	X	X	X	X	X	X
	0,05	SHOT	204	1481	32	X	X	X
	0,012	SHOT	198	1939	32	X	X	X

Cuadro 13: Método UNIFORM SAMPLING para TAZA

	US							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
HUCHA	0,05	PFH	325	1481	72	72	10	0,00042
	0,012	PFH	351	1939	72	72	4	0,00474
	0,05	FPFH	309	1481	72	72	5	0,00079
	0,012	FPFH	324	1939	72	72	7	0,00038
	0,05	RSD	X	X	X	X	X	X
	0,012	RSD	X	X	X	X	X	X
	0,05	SHOT	834	1481	72	X	X	X
	0,012	SHOT	659	1939	72	X	X	X

Cuadro 14: Método UNIFORM SAMPLING para HUCHA

3.3. Comparación de los métodos en velocidad y precisión del reconocimiento

Se ha hecho una comparación de lo que tarda cada uno de los métodos de extracción de KeyPoints en función del descriptor que se escoja (por duplicado para cada uno de los modelos de eliminación de planos).

En las Figuras [6], [7], [8] y [9] se pueden ver las gráficas de tiempo para cada uno de los métodos. Se puede observar que, salvo excepciones, el método de extracción de características que emplea más tiempo es el **ISS** y el de descriptores

	US							
	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLANTA	0,05	PFH	452	1481	108	108	7	0,00057
	0,012	PFH	435	1939	108	108	6	0,0025
	0,05	FPFH	506	1481	108	108	7	0,00105
	0,012	FPFH	537	1939	108	108	4	0,00073
	0,05	RSD	X	X	X	X	X	X
	0,012	RSD	X	X	X	X	X	X
	0,05	SHOT	828	1481	108	X	X	X
	0,012	SHOT	934	1939	108	X	X	X

Cuadro 15: Método UNIFORM SAMPLING para PLANTA

	Threshold	Método	(ms)	KP E	KP O	Matches	Correctos	%
PLC	0,05	PFH	62	1481	12	12	5	0,00051
	0,012	PFH	61	1939	12	12	5	0,0005
	0,05	FPFH	59	1481	12	12	6	0,00019
	0,012	FPFH	183	1939	12	12	12	0,07754
	0,05	RSD	X	X	X	X	X	X
	0,012	RSD	X	X	X	X	X	X
	0,05	SHOT	161	1481	12	X	X	X
	0,012	SHOT	125	1939	12	X	X	X

Cuadro 16: Método UNIFORM SAMPLING para PLC

es el **PFH**. Sin embargo, la mayor cantidad de mejores resultados se ha obtenido con estos dos modelos. Eso no quiere decir que para todos los objetos esta combinación o el uso de uno de estos dos métodos haya generado un resultado perfecto, si no que, la mayoría de buenas soluciones que se han conseguido son producto del uso de uno de los dos, o de la combinación de ambos.

Figura 6: Comparación tiempo para HARRIS

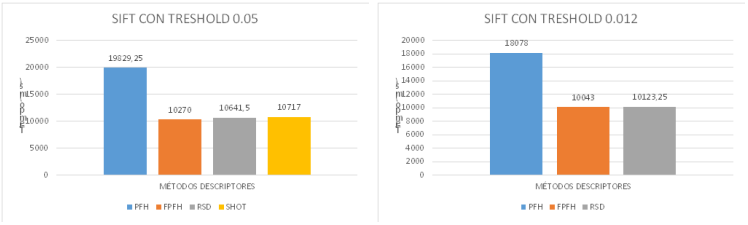


Figura 7: Comparación tiempo para SIFT

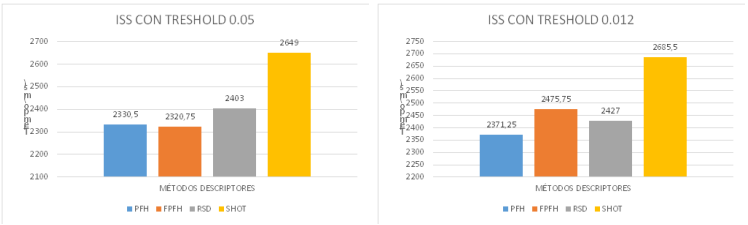


Figura 8: Comparación tiempo para ISS

3.4. Resultados

Tras todos los experimentos los mejores resultados para cada objeto son:

1. **TAZA - SIFT - RSD -Threshold = 0.012** [10]
2. **HUCHA - US - FPFH -Threshold = 0.012** [11]
3. **PLANTA - SIFT - PFH -Threshold = 0.05** [12]

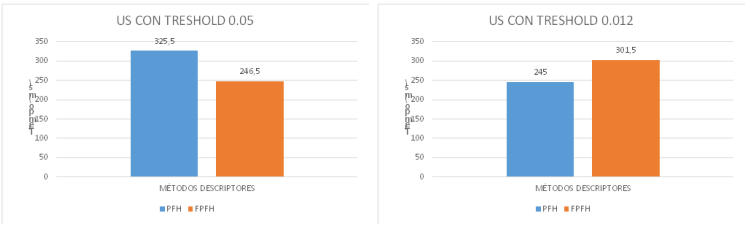


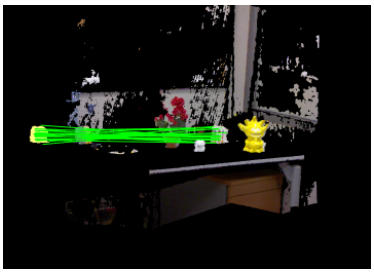
Figura 9: Comparación tiempo para US

4. **PLC - ISS - PFPH -Threshold = 0.05** [13]

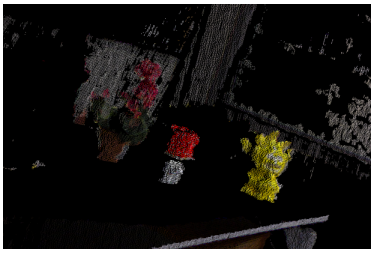
Aunque es cierto que, de los métodos con mejores resultados, la mitad, son con un valor u otro de *Threshold*, lo cierto es que se han obtenido más buenos resultados con el valor de índice mayor (0.05) que con el de índice menor (0.012). De hecho, estos son, junto a otras dos muestras, los únicos dos buenos resultados que se han obtenido con un umbral tan bajo. Mientras que, para un umbral mayor, se ha obtenido un mayor número de buenos resultados (aunque para dos de los objetos se obtenga una mejor precisión con un umbral de 0.012). Se puede extraer la conclusión de que llegado a cierto porcentaje de la eliminación de puntos (a través de la eliminación de planos dominantes), deja de ser tan importante el umbral al que se somete el PointCloud.

Por otro lado, **PFH** y **FPFH** son los descriptores con los que mejores resultados se obtiene, aunque para el caso de la taza el **RSD** ha mostrado buenos resultados. Su tiempo de procesamiento es alto, pero lo compensa con una mayor precisión. Para la extracción de KeyPoints, el modelo **SIFT** y **ISS** son los que más precisión devuelven. Sin embargo pasa como con los Descriptores, y para el caso de la hucha, el modelo **US** proporciona un resultado más exacto.

Se ha observado que los resultados que se muestran mantienen una relación con respecto al valor de **ACCURACY** o **PRECISIÓN %** mostrado en las tablas de antes. Todos ellos generan un valor que oscila entre 10e-05 y 10e-06, precisiones más cercanas al 0 que el resto. Se puede emplear como indicador de buen resultado a menor sea el valor devuelto.

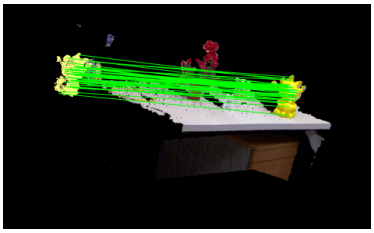


(a) Emparejamientos para la taza con SIFT+RSD a 0.012 Threshold



(b) Transformación final para taza

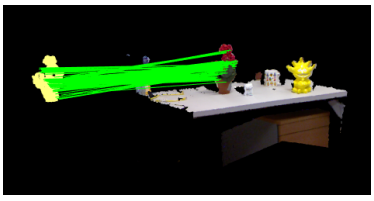
Figura 10: Mejor resultado para la TAZA



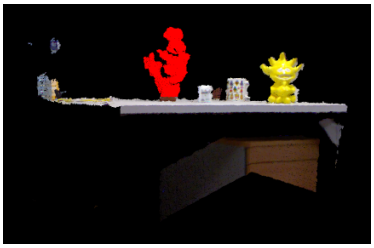
(a) Emparejamientos para la hucha con US+FPFH 0.012 Threshold

(b) Transformación final para la HUCHA

Figura 11: Mejor resultado para la HUCHA

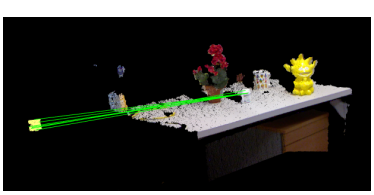


(a) Emparejamientos para la planta con SIFT+PFH a 0.05 Threshold



(b) Transformación final para la PLANTA

Figura 12: Mejor resultado para la PLANTA



(a) Emparejamientos para el plc con ISS+FPFH a 0.05 Threshold

(b) Transformación final para el PLC

Figura 13: Mejor resultado para PLC

4. Conclusiones

PCL es una excelente herramienta que cuenta con diferentes métodos para el tratamiento de nube de puntos, yal poseer una forma modular, puede ser implementado en plataformas de bajo rendimiento; además de tratarse de una biblioteca liberada bajo licencia BSD, lo cual conlleva un gran crecimiento en los algoritmos desarrollados. Otro aspecto es que cuenta con el respaldo de grandes compañías del área de la robótica y la visión artificial, situación que garantiza su evolución y perfeccionamiento.

Con respecto a los resultados de esta práctica. La eliminación de planos deja de ser efectiva tras alcanzar cierto umbral (50 % en este caso) ya que la velocidad de procesamiento debido al menor número de puntos no compensa la pérdida de información. Llegado a estos límites de umbral, el *Threshold* aplicado suele ser poco relevante, aunque se han obtenido mejores precisiones con un valor más alto (0.05). Por otro lado, tras comparar la cantidad de emparejamientos correctos, tiempos de procesamiento y precisión de los diferentes modelos de extracción y descripción, se concluye que el método que mejor funciona para extraer buenos resultados en el reconocimiento de objetos en escenas 3D con el pipeline implementado y los parámetros de PCL cargados es **SIFT + FPFH**. Si se implementa esta combinación a todos los objetos, los resultados (generales) son los mejores.

Referencias

[1] Raúl Calatayud e Ignacio Pérez. *$deteccion_objetos_3D_nubePuntos$*. 2021. URL: `https://github.com/nachoperezzv/deteccion_objetos_3D_nubePuntos.git`.

[2] A. Felipe Calvo Salcedo. "Procesamiento de nubes de puntos por medio de la librería PCL". En: *Programa de Ingeniería Electrónica, Universidad Tecnológica de Pereira, Pereira, Colombia* (Diciembre 2012).

[3] WILLOW GARAGE. *What is PCL?* 2012. URL: `http://pointclouds.org/about.html`.

[4] Radu Bogdan Rusu y Steve Cousins. *3D is here: Point Cloud Library (PCL)*. 2011.

[5] Federico Tombari. "Registration with the Point Cloud Library. A modular Framework for Aligning in 3-D". En: *IEEE Robotics Automation Magazine* (Diciembre, 2015).