CHAPTER X

ESTIMACIÓN DE SUELOS NAVEGABLES PARA INTERIORES

JOSE PARDEIRO, JAVIER V. GÓMEZ, DAVID ÁLVAREZ, LUIS MORENO

Robotics Lab, Universidad Carlos III de Madrid.

Este capítulo explica como se adapta a interiores un algoritmo diseñado para segmentar el suelo navegable del entorno y además, gracias a la fusión de sensores, es capaz de superar las limitaciones de los sensores de medida empleados. Para ello se utilizará una cámara RGB-D de bajo coste y se analizarán todos los inconvenientes encontrados tanto por su aplicación a interiores como por limitaciones de hardware. Además se proponen soluciones para aliviar en parte dichos problemas con resultados satisfactorios.

1 Introducción

Actualmente existen dos vertientes diferentes para abordar tareas tales como el modelado del entorno. Tradicionalmente solo existía la opción de utilizar sensores láser, pero de forma reciente están apareciendo unos nuevos modelos de cámara basados en la tecnología RGB-D. Estas últimas cámaras cuentan con el aliciente de un precio bajo y un balance excelente entre posibilidades y precio. Por su parte, los sensores láser incrementan notablemente el coste del proyecto, aportando una mayor precisión que en muchos casos no resulta necesaria. Por tanto, actualmente la decisión de una tecnología u otra reside principalmente en las necesidades y el presupuesto del proyecto.

2 Trabajo previo

En el 2005, el Departamento de Defensa norteamericano organizó un concurso llamado DARPA Grand Challenge que consistía en llevar de forma autónoma un vehículo de un punto de Estados Unidos a otro sin ningún tipo de aporte humano. La Universidad de Stanford ganó dicho concurso gracias a que su algoritmo de segmentación de suelos transitables permitía a su vehículo moverse a una velocidad mayor que los demás participantes (Dahlkamp, 2006).

En dicha tarea utilizaron dos clases de sensores, uno compuesto por sensores láser capaces de detectar la presencia de obstáculos en el trayecto, y otro compuesto por una cámara RGB. Debido a la velocidad alcanzada por el vehículo, el rango de trabajo de los sensores láser era escaso para ser funcional, por lo que optaron por combinarlo con una cámara RGB. La forma de lograr que dicha combinación fuese satisfactoria diseñaron un algoritmo basado en el color. Para ello realizaron una suposición, en la cual asumían que si un suelo transitable según el sensor poseía cierto color, si el que se encontraba inmediatamente después mantenía el color se consideraba una extensión del suelo anterior.

En todos los pasos anteriores se extraían los parámetros de color de la zona transitable y mediante distintos métodos de ajuste reducían el conjunto de datos a funciones estadísticas gaussianas. Tras eso, utilizaban un método estadístico para comprobar si el espacio de color del resto de la imagen captada por la cámara RGB era muy similar al de la función gaussiana, validando así el terreno. Como ejemplo, en la figura 1 se muestra un camino aleatorio el rango abarcado por los sensores láser lo delimitan las líneas azules.

(a) Imagen inicial. (b) Sección detectada por los sensores

Fig. 1. Funcionamiento del algoritmo. Imágenes extraídas de (Dahlkamp, 2006).

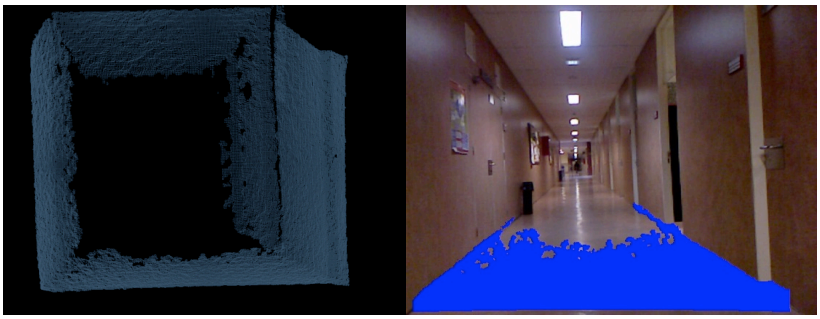
Tras el proceso de validación del terreno, el resultado se corresponde a la figura 2, siendo la zona blanca la considerada como transitable por el vehículo.

Fig. 2. Resultado del algoritmo. Imágenes extraídas de (Dahlkamp, 2006).

Para lograr, además, una mayor fiabilidad en el resultado se agregaron sistemas de aprendizaje y actualización de gaussianas.

3 Algoritmo de segmentación y aprendizaje de suelos transitables

En este algoritmo se utilizará una cámara RGB-D de bajo coste. Estas cámaras están compuestas por una cámara RGB y un conjunto de sensores de profundidad capaces de medir distancias. Aunque dichos sensores son suficientes para detectar la geometría del entorno dentro de su rango, éste es de aproximadamente 5 metros, que puede resultar escaso dependiendo de la aplicación. En el caso de un pasillo, la salida proporcionada por una cámara RGB-D se corresponde a la figura 3.



(a) Nube de puntos original (b) Nube de puntos sobre imagen RGB

Fig. 3. Ejemplo de imagen tomada en una cámara RGB-D.

Como se puede ver, el rango de la imagen extraída por la cámara RGB es mucho más lejano que el obtenido por los sensores. Por tanto, una parte importante de este algoritmo consiste en paliar dicha limitación, además de otros problemas como la presencia de brillos u otros derivados de la iluminación artificial, similitud de colores, etc. Nuestra contribución tratará de solucionar todos estos problemas modificando lo menos posible el algoritmo inicial.

El algoritmo empleado está basado en el aprendizaje de los parámetros de color del suelo segmentado con información RGB-D. La estimación del resto de suelo navegable se basa en que dicho suelo mantiene las propiedades de color. Así, el algoritmo es capaz de estimar el mapa del entorno en aquellas zonas que están fuera del rango de los sensores de distancia.

3.1 Secciones que componen el algoritmo

Al comienzo del algoritmo, se trabaja directamente con la nube de puntos obtenida de los sensores de la cámara RGB-D, pues es la mejor base para extraer la información del entorno al ser una representación tridimensional. Debido a que la información que interesa en este caso es estrictamente la del suelo, el primer paso consiste en seccionar la nube para únicamente trabajar con el suelo.

Una vez determinada la zona de la nube que se considera suelo, es preciso traspasar la información de la nube de puntos a píxeles, para así poder relacionarlo con la imagen RGB. Posteriormente, se procede a extraer los parámetros de color de los puntos de la nube pertenecientes al suelo. Debido a que es un flujo muy grande de datos, se procederá a agrupar los datos

utilizando una técnica de clustering y posteriormente transformar esa información en parámetros de una función de distribución probabilística de tipo gaussiana de tres dimensiones, para facilitar los cálculos posteriores.

Con toda esta información guardada se comienza a trabajar con la imagen extraída de la cámara RGB. Para ello se utiliza la transformación desde puntos de una nube a píxeles realizada anteriormente y se extraen los parámetros de color de la imagen RGB.

Para conocer cuáles de esos píxeles pertenecen al suelo transitable se comparan sus parámetros de color con cada gaussiana. Si son parámetros cercanos se considera una extensión de suelo que no alcanza a detectar el sensor de la cámara RGB-D, en caso contrario se discrimina y se considera obstáculo.

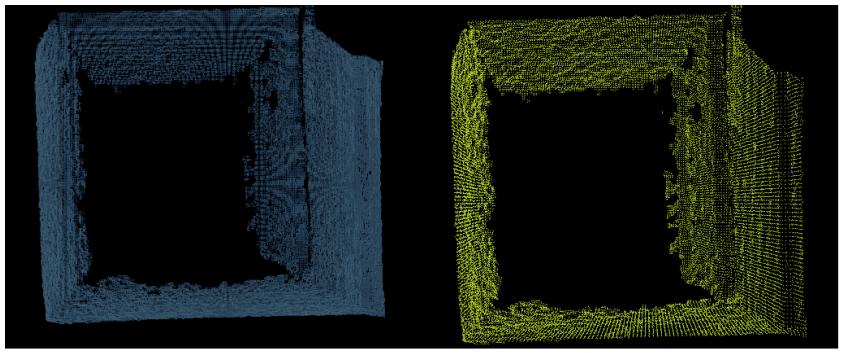
Debido a que es un algoritmo preparado para ser ejecutado en bucle, se han implementado técnicas de aprendizaje para evitar la saturación del sistema almacenando parámetros gaussianos muy similares.

3.2 Filtrado

Se comenzará el algoritmo con un proceso de filtrado. El filtrado consiste en la eliminación de un exceso de datos, tratando de suprimir la información superficial para mantener la característica. Así se consigue un aumento de velocidad en el procesado de información. Para ello, se hará uso de un filtro de tipo voxel grid.

El filtro voxel grid genera una rejilla de cubos de tamaño idéntico en la nube de puntos. Todos los puntos que pertenecen a un mismo cubo (voxel) son sustituidos por su centroide. Al finalizar, la nube filtrada mantiene la forma original, con un número menor de puntos y una densidad más uniforme.

La disminución de puntos depende íntegramente del tamaño del cubo, reduciéndose considerablemente a medida que el tamaño del cubo aumenta. Por contra, al disminuir el número de punto se aumenta el riesgo a perder elementos característicos, por lo que de forma empírica fue necesario encontrar un punto medio entre ligereza y fidelidad con el original. El resultado obtenido se muestra en la figura 4.



(a) Nube original (b) Nube filtrada con voxel = 0.03

Fig. 4. Filtrado de la nube de entrada.

3.3 Segmentado de la nube de puntos

El segundo paso del algoritmo se corresponde con la segmentación de la nube en los distintos planos que lo conforman. Para realizar este ajuste de planos, se hará uso del algoritmo RANSAC (Fischler, 1981), abreviación de RANdom SAmple Consensus. Es un método que, de forma iterativa, ajusta los parámetros que definen a un plano que abarque el mayor número de puntos. Para minimizar los errores, RANSAC también evalúa la distancia entre los puntos, discriminando los puntos que superen un límite de distancia.

Matemáticamente, un plano se puede definir como en (1):

$$ax + by + cz + d = 0 \tag{1}$$

Tras obtener los coeficientes (a, b, c y d), el programa los analiza para determinar su posición. Geométricamente, consideramos que la posición de la cámara es paralela al suelo en su eje z y perpendicular en su eje y. Con respecto a la referencia de la cámara, si el valor absoluto del coeficiente a del plano se encuentra comprendido entre 0.9 y 1, la situación del plano se considera vertical. Por ese motivo, las dos posibilidades que existen es que sea algo relacionado con la pared derecha, o con la pared izquierda. Por contra, si es el coeficiente b el que se encuentra entre 0.9 y 1, el plano es considerado horizontal, dejándonos como posibilidades que se encuentre relacionado con el suelo o con el techo.

Aunque se pueda conocer si el plano es vertical u horizontal, es necesario determinar en qué posición exacta se encuentra. Para ello, se analiza el coeficiente d del plano. Si se ha determinado que el plano es vertical y dicho coeficiente tiene un valor positivo, está situado a la derecha, mientras

que si es negativo, lo hace a la izquierda. En los planos horizontales, un coeficiente d positivo indica techo, y uno negativo suelo. La segmentación da como resultado lo mostrado en la figura 5 y la tabla 1.

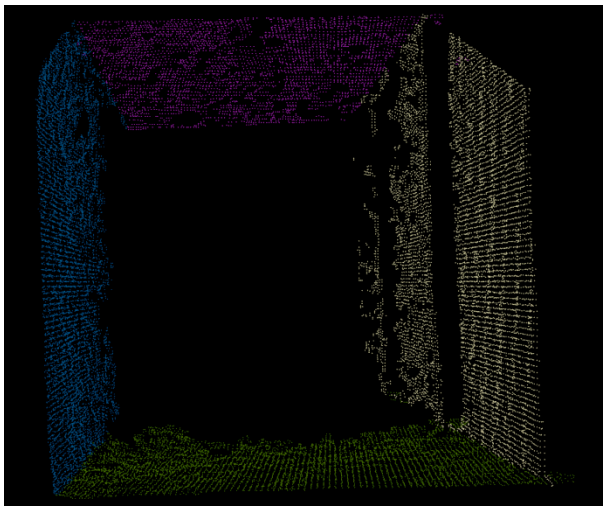


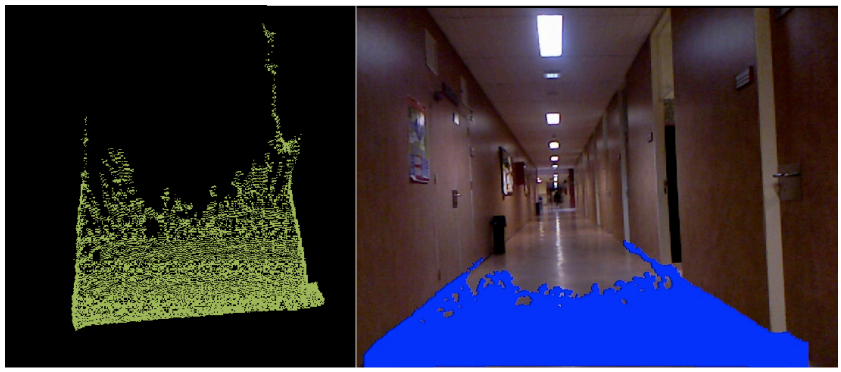
Fig. 5. Ejemplo de nube segmentada.

Tabla 1. Parámetros de los planos de la nube segmentada.

	a	b	c	d
Pared derecha	0.990	0.995	0.124	1.134
Pared izquierda	0.995	-0.030	0.093	-1.204
Techo	-0.047	0.999	0.025	0.920
Suelo	-0.044	-0.999	-0.001	-1.954

3.3.1 Píxeles pertenecientes al plano del suelo

Una vez calculados los coeficientes de los planos de las superficies, es necesario conocer la ubicación en píxel del suelo. Para ello se parte de la nube original, que se encuentra ordenada, y de los coeficientes del plano que devuelve la segmentación de planos. Con todo eso, es posible conocer de una forma aproximada que píxeles pertenecen al suelo. La forma de lograrlo es utilizando la distancia punto a plano, estableciendo una distancia máxima a la que puede estar un punto para ser considerado suelo. De dichos puntos, además, se obtiene la información RGB. El resultado es el obtenido en la figura 6.



(a) Nube correspondiente al suelo (b) Nube sobre la imagen RGB.

Fig. 6. Ejemplo de píxeles pertenecientes al suelo.

3.4 Análisis de los parámetros de color

En el algoritmo inicial (Dahlkamp, 2006) los cálculos para comparar los parámetros de color son simplificados gracias a su modelado como funciones de distribución gaussianas. Por tanto, se requiere previamente realizar agrupaciones de los datos de color mediante algún método de clustering.

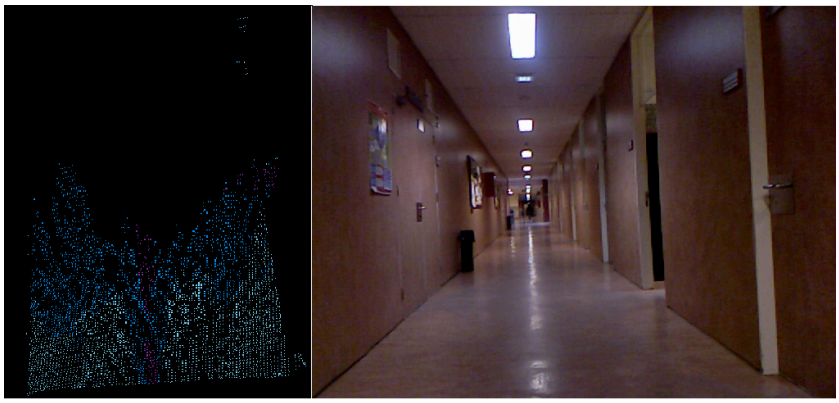
Dentro de la variedad de los algoritmos de clustering, se ha comprobado en situaciones similares que el algoritmo que recibe el nombre de k-means o k-medias (Kanungo, 2002) es adecuado para la aplicación propuesta. En este tipo de algoritmo, se parte de un número k de agrupaciones cuyo fin es etiquetar todos los datos en cada una de las agrupaciones.

Una vez se han obtenido las agrupaciones, es cuando se obtienen los parámetros de las gaussianas. Una distribución gaussiana puede definirse utilizando su media y su varianza. Para calcular su media, μ, y su varianza, σ^2, la fórmulas aplicadas son (2) y (3):

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{2}$$

$$\sigma^2 = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{3}$$

El resultado del clustering corresponde a la figura 7 y la tabla 2.



(a) Clustering sobre la nube del suelo con k=3

(b) Imagen RGB original.

Fig. 7. Ejemplo de clustering.

Tabla 2. Resultado numérico del clustering.

	R	G	B
Media 1	122.773	109.447	122.984
Varianza 1	249.702	278.566	266.490
Masa 1		303	
Media 2	81.911	62.066	62.467
Varianza 2	30.2974	31.2947	77.3858
Masa 2		3221	
Media 3	94.288	78.184	85.672
Varianza 3	35.310	30.985	60.284
Masa 3		309	

3.4.1 Aprendizaje de gaussianas

En este paso se realizará el aprendizaje de gaussianas. A nivel conceptual, se puede hablar de que el sistema aprende terrenos. Con la información tridimensional de la cámara se logra un conocimiento del terreno casi total, pero su rango es demasiado corto como para depender únicamente de ello. Por eso también se realiza el análisis del color. Debido a las pequeñas variaciones que existen en los colores que componen el terreno (el más mínimo brillo o la más pequeña sombra lo alteran), es prácticamente imposible que dos procesos de agrupaciones generen los mismos parámetros gaussianos de forma exacta.

Para cada nuevo frame, se analiza el suelo y se calculan sus gaussianas de color, (μ_T, Σ_T), siendo μ el vector de medias y Σ la matriz de covarianzas. Dichas gaussianas son comparadas con las aprendidas hasta el momento (μ_L, Σ_L) mediante una variación de la distancia de Mahalanobis (Penny, 1996) según la expresión (4).

$$(\mu_L - \mu_T)^T (\Sigma_L + \Sigma_T)^{-1} (\mu_L - \mu_T) \leq d \tag{4}$$

La función del parámetro 'd' es delimitar el nivel de similitud aceptable entre dos gaussianas, pudiendo variar el valor máximo en función de la precisión buscada. En caso de cumplirse dicha expresión, se actualizarán los parámetros de la gaussiana aprendidos, siguiendo (5), (6) y (7):

$$\mu_L \leftarrow \frac{m_L \mu_L + m_T \mu_T}{m_L + m_T} \tag{5}$$

$$\Sigma_L \leftarrow \frac{m_L \Sigma_L + m_T \Sigma_T}{m_L + m_T} \tag{6}$$

$$m_L \leftarrow m_L + m_T \tag{7}$$

Como se puede apreciar, la actualización depende de la masa, es decir, el número de puntos, logrando así que cuanto mayor sea la masa, menos varíe la gaussiana aprendida.

En caso de no cumplir la igualdad, se realizaría otra operación. Para descartar gaussianas causadas por casos aislados o situaciones muy concretas, como un exceso de luz en una zona pequeña, se evalúa su masa. Si es superior al 30% de la masa de la gaussiana superior, se aprendería una gaussiana nueva. El número máximo de gaussianas aprendidas se determina al principio del programa. En caso de que se haya llenado el vector, se compara la masa de la nueva gaussiana que se quiere aprender, y en caso de ser mayor que la gaussiana aprendida con menos puntos se eliminará dicha gaussiana para aprender la nueva. Así se evita mantener como gaussiana aprendida elementos que aparecen en momentos circunstanciales.

3.5 Región de interés

En esta sección se van a analizar los pasos necesarios para poder hallar la región de interés de la nube, entendiendo como región de interés toda la zona que pertenece al suelo. El cálculo de una región de interés es necesa-

rio para poder separar los píxeles de la imagen RGB que pueden pertenecer al suelo de los que es imposible que formen parte de dicho suelo.

A nivel conceptual, este paso es muy sencillo. En este punto tenemos los coeficientes que definen el plano que contiene al suelo, pero en tres dimensiones, por lo que el rango de esta información está limitado. Se comienza definiendo la recta intersección entre el plano del suelo y los dos de las paredes, dando como resultado dos rectas. Una vez con las rectas, se han de proyectar en la imagen RGB bidimensional. El resultado se muestra en la figura 8.

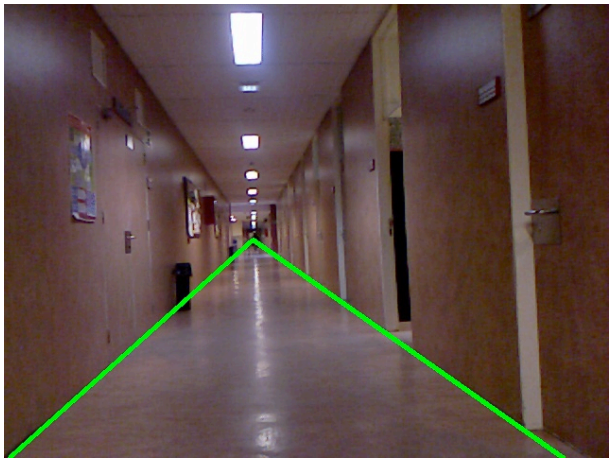


Fig. 8. Ejemplo de región de interés.

3.6 Etiquetado del suelo

En este momento es donde se engloba todo lo realizado anteriormente para poder ampliar el rango de la cámara. Anteriormente, junto a la región de interés, se han etiquetados qué píxeles se encuentran dentro del área que delimitan las dos rectas intersección que forman los planos de las paredes con el suelo y se han extraído sus parámetros de color.

Para poder conocer si un punto de color se engloba dentro de una distribución gaussiana, es necesario calcular la distancia a la que se encuentra de la distribución, y para ello, en estadística multivariante lo más óptimo es usar la distancia de Mahalanobis. Esta distancia proporciona los parámetros más óptimos debido a que no tiene en cuenta solo el valor medio, sino la matriz de covarianzas.

La fórmula para conocer la distancia de Mahalanobis se corresponde a (8).

$$M(x) = \sqrt{(\overline{x} - \overline{y})^T \Sigma^{-1} (\overline{x} - \overline{y})} \quad (8)$$

siendo M la distancia a la que se encuentra y tendiendo a 0 cuanto más cerca se encuentre de la distribución gaussiana. Una vez calculado se obtiene lo mostrado en la figura 9.

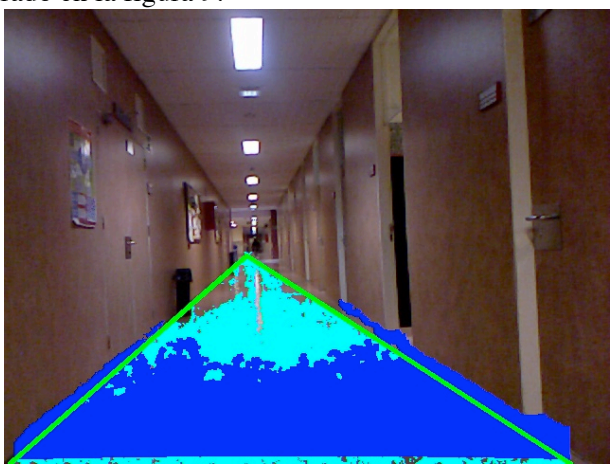


Fig. 9. Ejemplo de etiquetado del suelo con M(x)=2.7.

3.7 Generación de mapa

En este último paso se genera un mapa vista de pájaro del terreno, conociendo las distancias reales a las que se encuentra cada píxel catalogado como transitable.

Para ello se partió del modelo pinhole. Conociendo los parámetros de calibración de la lente y la distancia focal nos era posible conocer cuál era la distancia real de cada punto. El punto negativo que se encontró para este método fue la gran pérdida de precisión a medida que aumentaba la distancia. Si bien en distancias cortas su error era muy pequeño, en distancias largas aumentaba de forma considerable, errando las distancias en más de 20 metros en los puntos más alejados.

Como alternativa, se propone un modelo basado en la relación de la distancia real entre el punto y la cámara y el píxel correspondiente en el que aparece en la imagen RGB. Cada punto etiquetado como suelo es situado dentro del triángulo definido por la región de interés obtenida en la sección 3.5, según figura 10.

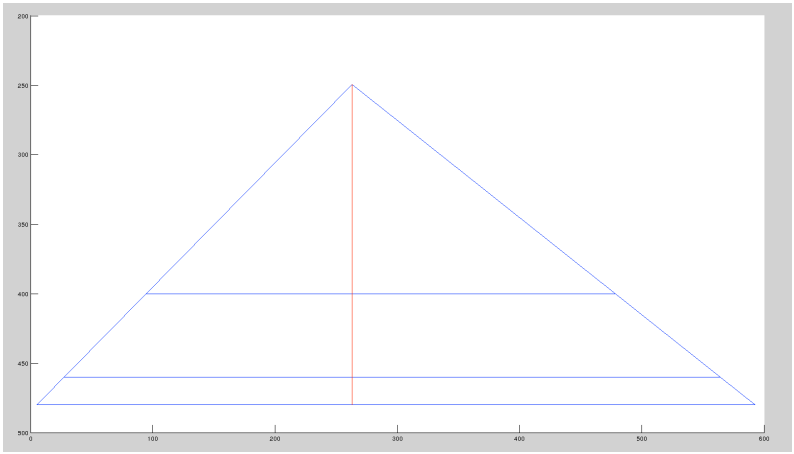


Fig. 10. Ejemplo de triángulos.

Como se puede ver, a medida que el punto a evaluar se encuentra más lejos, el área disminuye, por lo que podríamos establecer la siguiente relación de (9):

$$dist \propto \frac{1}{area} \tag{9}$$

Matemáticamente, esta ecuación lleva a inexactitudes en puntos lejanos, ya que dichos puntos tendrán un área nula, haciendo que la relación tienda a infinito. Dicho error se tendrá que tener en cuenta en el futuro. Se realiza un proceso de ajuste de una curva que relacione la distancia en píxeles con la real (figura 11), acorde a la expresión (9), ya que dicha relación puede ser extraída del suelo navegable segmentado inicialmente. Los parámetros de ajuste se corresponden a (10), y el resultado se corresponde a la figura 12.

$$f(x) = \frac{414.1}{x - 216} \tag{10}$$

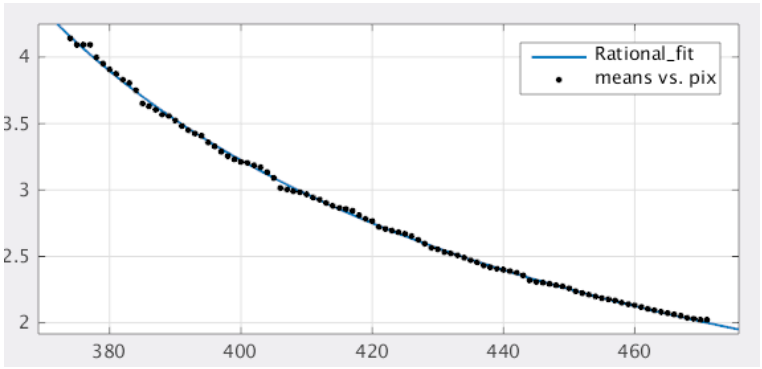


Fig. 11. Ejemplo de curva de ajuste de parámetros.

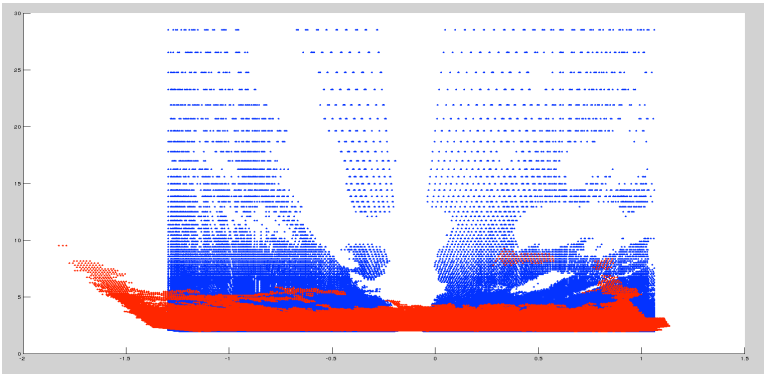


Fig. 12. Ejemplo de mapa generado.

Pese a que intuitivamente se esperan contornos más definidos, la lente de la cámara realiza una proyección con forma de tetraedro, lo cual provoca que los elementos parezcan deformados. Con ello se ha pasado de 5 metros de rango a 30, logrando así un aumento del rango en un 600%, dando por finalizado el algoritmo.

4 Resultados experimentales

Trabajar en el sistema dinámico conlleva varios problemas. El primero viene derivado de la utilización de iluminación artificial en el interior. La iluminación artificial tiene un funcionamiento de 50Hz, por lo que la luz no es constante. Aunque eso es prácticamente inapreciable para el ojo humano, cabe la posibilidad de que la iluminación de distintos frames cap-

tados sea muy distinta. Eso causa problemas en el etiquetado, ya que un frame oscuro tras varios claros hace que las gaussianas aprendidas sean diferentes a la nueva y empeore el funcionamiento general. Por tanto, la iluminación natural es más recomendable si se busca un funcionamiento mejor.

Por otro lado, los suelos brillantes generan multitud de brillos que pueden provocar a su vez problemas en el etiquetado de suelo, como se verá en las pruebas siguientes. Estos problemas surgen especialmente si no hay brillos en el rango de la cámara, pero sí en el de la cámara RGB, ya que una vez que se aprenda una gaussiana que represente los brillos el error cesará.

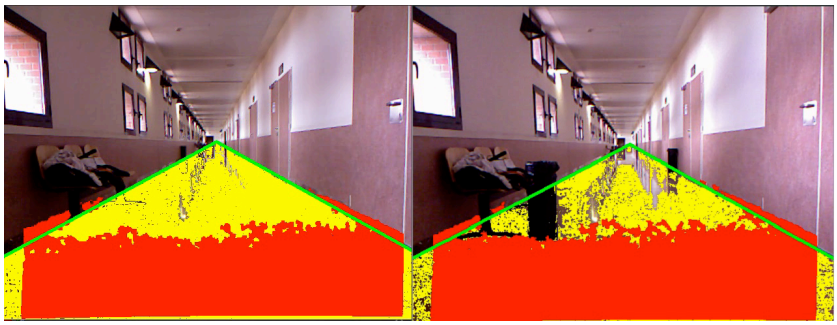
La actualización de gaussianas inicialmente fue causa de errores, por dos motivos que iban de la mano. Primero, la condición utilizada para determinar si dos gaussianas eran iguales resultó ser excesivamente permisiva. Eso hacía que al actualizar la gaussiana acabase resultando una gaussiana tan dispersa que abarcase todas las demás. Por tanto, de forma empírica se ajustó la condición para actualizar, y se tomó la decisión de saturar la gaussiana una vez llegada a una masa de 5000 puntos, impidiendo así su crecimiento excesivo.

Finalmente, la distancia de Mahalanobis es preciso actualizarla en función del terreno para su correcto funcionamiento. En las pruebas posteriores se podrá comprobar como una distancia de Mahalanobis que resulta apta en un caso no lo era para otro tipo de suelo.

El algoritmo ha sido programado en el lenguaje C++ utilizando las librerías PCL (Rusu, 2011) y OpenCV (Baggio, 2012) para facilitar el tratamiento tanto de la nube de puntos obtenida por la cámara RGB-D como el de la imagen RGB. Para la generación del mapa se ha utilizado la herramienta matemática Matlab.

4.1 Pruebas en distintos pasillos

Una consideración muy importante corresponde a la distancia de Mahalanobis. Por problemas relacionados con el espacio de color RGB, una distancia correcta en un lugar no tiene por qué ser válido para otro. Como ejemplo, se mostrará el resultado de la distancia de Mahalanobis calculada como correcta en la fase estática y el resultado con la distancia más adecuada en este caso. Se obtiene como resultado la figura 13.



(a) Resultado con M(x)=2.7 (b) Resultado con M(x)=2

Fig. 13. Ejemplo de segmentación en otro suelo.

En este caso es posible apreciar como evita tanto los objetos que se encuentran en medio como los brillos producidos por la luz y los reflejos. Si de esta información se genera el mapa de vista de pájaro, el resultado es la figura 14.

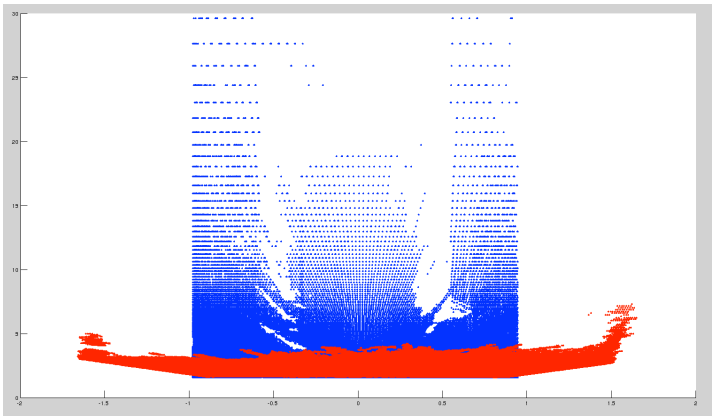
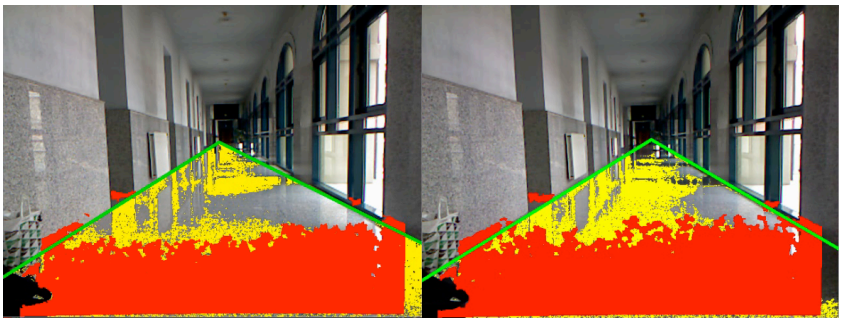


Fig. 14. Mapa de vista de pájaro del pasillo.

Usando la misma distancia de Mahalanobis en otro pasillo aparece un resultado similar, por lo que se mostrará tanto el resultado con la distancia anterior como con la correcta. El resultado se corresponde a la figura 15.



(a) Resultado con M(x)=2 (b) Resultado con M(x)=2.7

Fig. 15. Ejemplo de segmentación en suelo con mucho brillo.

Un resultado mucho mejor, que además evita los objetos que aparecen en medio. Por tanto, la elección correcta de una distancia de Mahalanobis es imprescindible al cambiar de superficie, que se debe ajustar de un modo experimental.

5 Conclusiones y trabajo futuro

Aunque el algoritmo original demostró ser muy potente, su aplicación a entornos de interior genera numerosas limitaciones.

Por un lado, la cámara solo funciona en interiores. Debido a la tecnología basada en infrarrojos, su uso en exteriores es limitado o casi nulo debido a las interferencias de la luz solar. Además, en interiores con iluminación artificial surge otra limitación, y es la frecuencia a la que funciona la luz artificial. Al funcionar a 50Hz, pese a que es un caso que no suele ocurrir, se puede captar un frame en el que todo esté muy oscuro debido a que se ha capturado la parte nula de la onda. Mediante el uso de otros espacios de color como el HSV o LAB, este problema puede ser paliado en parte.

El rango de actuación de los sensores de la cámara es una limitación intrínseca que se ha aliviado para zonas concretas (como el suelo).

Como trabajo futuro se propone el estudio de una versión paralelizada de este algoritmo para acelerar la tasa de refresco.

Otra limitación que se encuentra es su uso exclusivo en pasillos. Es un código que solo trabaja en terrenos ideales, necesitando superficies relativamente lisas, y sobre todo, la localización de pared derecha, izquierda y suelo para poder funcionar correctamente. En el algoritmo inicial, las carreteras se distinguen muy bien del resto del terreno. Sin embargo en inte-

riores es muy frecuente disponer de suelos y paredes de colores muy similares.

Referencias

Baggio, DL, et al. 2012. Mastering OpenCV with Practical Computer Vision Projects, Packt Publishing: Birmingham, UK.

Dahlkamp, H, et al. 2006. Self-supervised monocular road detection in desert terrain. Proceedings of Robotics: Science and Systems, Piladelphia, USA.

Fischler MA, Bolles RC. 1981 Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24: 381-395.

Kanungo, Tapas. 2002. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. IEEE Transactions on Pattern Analysis and Machine Inteligente, 24(7): 881–892.

Penny, KI. 1996. Appropriate Critical Values When Testing for a Single Multivariate Outlier by Using the Mahalanobis Distance. Applied Statistics 45(1): 73-81.

Rusu RB, Cousins S. 2011. 3D is here: Point Cloud Library (PCL). In Proc. Int. Conference on Robotics and Automation (ICRA) 2011.