Estado del arte: SLAM visual

Felipe Pérez Molina

Máster de Visión Artificial
Universidad Rey Juan Carlos
Comunidad de Madrid, España

1 Introducción

[Pendiente]

2 Localización con mapa conocido

El primer tipo de localización es aquella que hace uso de un mapa para poder orientarse y así ubicarse en el entorno. La definición de mapa puede ser muy variada, podemos encontrarnos desde información detallada del entorno hasta la localización de unas balizas o marcadores reconocibles.

El sistema visual hará uso de los sensores que tiene incorparado y estimará su posición en base a una función de probabilidad o en base a cálculos geométricos.

2.1 Localización visual basada en geometría

Esta técnica hace uso de la información visual que te aporta el sensor, un módelo de la cámara y la información del entorno en el que se mueve el dispositivo (como por ejemplo un marcador).

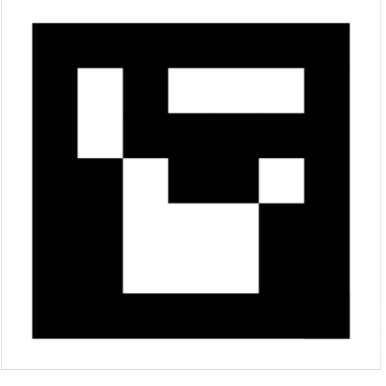

Figure 1: Ejemplo de marcador visual de la librería de ArUco.

2.1.1 Modelo de cámara Pin Hole

El modelo de la cámara Pin Hole se basa en la idea de que todos los rayos de luz pasan por el mismo punto (foco) e impactan en un plano formando una imagen. La distancia que separa el plano focal y el plano imagen toma el nombre de distancia focal, de la que depende el tamaño de los objetos observados en la imagen, siendo más pequeños cuanto mayor sea la distancia focal. La hipótesis que se realiza es asumible ya que las cámaras actuales son capaces de concentrar los rayos de luz en un solo punto debido a sus pequeñas lentes.

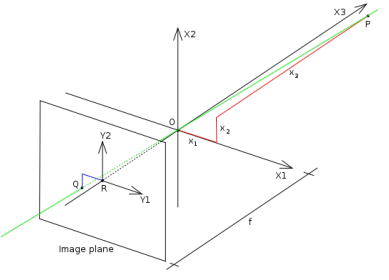

Figure 2: Modelo de cámara Pin Hole.

La figura [2] muestra los principales elementos que componen el modelo:

1. Un sistema de coordenadas 3D con origen en 0. Los 3 ejes del sistema de coordenadas son referidos como X1, X2 y X3. El eje X3 apunta en dirección a la camara y es referencia como el eje óptico o eje principal. El plano que intersecciona con los ejes X1 y X2 (plano principal) se encuentra en frente de la cámara.

2. El plano de la imagen donde el mundo 3D se proyecta en dirección al centro óptico. Siendo éste paralelo a los ejes X1 y X2 localizados a una distancia "f" desde el origen O en la dirección negativa del eje X3. Dicha distancia es conocida como la distancia focal.

3. Un punto R en la intersección del eje óptico y el plano de la imagen. Este punto es reperido como el punto principal o el centro de la imagen.

4. Un punto P en algún lugar del entorno con coordenadas(x_1,x_2,x_3).

5. La línea de proyección del punto P a la cámara.

6. La proyección del punto P al plano de la imagen, denotado como Q Este punto es dado por la intersección del la línea de proyección(verde) y el plano de la imagen.

7. Finalmente, hay un sistema de coordenadas 2D en el plano de la imagen, con origen R y con los ejes Y1 y Y2 que son paralelos a X1 y X2, respectivamente.

A pesar del pequeño tamaño de las lentes, las imágenes captadas no son perfectas. Existen pequeñas deformaciones que no son apreciables para el ojo humano, por lo que obtener los parámetros intrínsecos de la cámara juega un papel esencial en cada uno de los algoritmos

para obtener unas medidas correctas. La obtención de dichos parámetros recibe el nombre de calibración, calculando la matriz de calibración K [1].

$$K = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Los parámetros f_x y f_y representan la distancia focal en los ejes X e Y, mientras que u_0 y v_0 la posición del centro óptico en la imagen. Esta matriz permite realizar la proyección de un punto 3D en el mundo a un punto 2D en el plano imagen de nuestra cámara, sólamente necesitamos multiplicar dicho punto por la matriz K.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2}$$

Ahora bien la localización 3D o pose se define como la posición 3D de la cámara y su orientación respecto cierto sistema de referencia externo, y se suele expresar como una matriz 3x4 (4x4 si trabajamos en coordenadas homogéneas) que es el resultado de la multiplicación entre una matriz de rotación (R) 3x3 por un vector de traslación (t) 3x1.

$$RT = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{3}$$

Quedando la proyección finalmente como:

$$P_{uv} = K \cdot RT \cdot P_{XYZ} \tag{4}$$

2.1.2 Perspective-n-Point

Una vez explicado como se proyectan los puntos del mundo 3D al plano 2D de una cámara, podemos pasar al calculo de la localización de la cámara dado un conjunto *n* de puntos 3D y sus proyecciones 2D. Dicho problema fue presentado por Fischler y Bolles [2], denotándolo como Perspective-n-Point [3]:

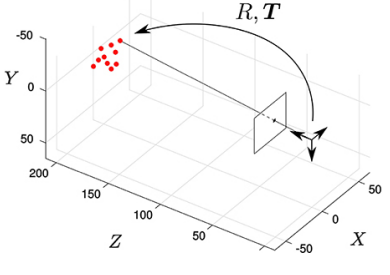

Figure 3: Perspective-n-Point

Dependiendo de la cantidad de puntos que tengamos existen varias soluciones:

- Dos puntos o menos: soluciones infinitas.

- Tres puntos: ocho posibles soluciones de las cuales, la mitad están frente a la cámara (Haralick et al., 1994 [5]).
- Cuatro puntos: si los cuatro puntos son coplanares solo existe una solución, en cualquier otro caso hay dos soluciones [Horaud et al., 1989 [11]].
- Cinco puntos: es posible encontrar una solución pero ésta dependerá del método que utilicemos para minimizar el error. La técnica más usada es la de los mínimos cuadrados, pero también existen otras con una mejor eficiencia y precisión [4].

3 Localización sin mapa conocido

En la anterior sección he explicado algunos de los algoritmos del estado del arte para los casos en los que poseemos un mapa, con una serie de puntos conocidos XYZ que se proyectarán en la cámara y a través de algoritmos como Perspective-n-Point podemos averiguar la posición y orientación del sensor.

En esta sección explicaré algunos de los algoritmos más famosos en el caso de que no tengamos un mapa.

3.1 Odometría visual

La odometría visual es el proceso de determinar la posición y la orientación del sensor analizando una secuencia de imágenes captadas por la cámara. La odometría es un término que se utiliza generalmente en el campo de la robótica en el que se estima la posición del robot con ruedas durante la navegación. Para poder estimarla se usa la información sobre la rotación de las ruedas.

Por otro lado, el término "odometría visual" ,fue acuñado por Nister [8], no requiere conocer la geometría del robot, se puede usar en cualquier robot que incorpore una cámara.

Dependiendo del tipo de visión que presente el robot, la odometría visual se basará en diferentes métodos.

3.1.1 Odometría visual con visión monocular

La utilización de una sola cámara para localizarte implica una mayor complejidad que en el problema de visión estéreo. Debemos tomar una serie de imágenes separados a una cierta distancia, y emparejar aquellos puntos de interés. Además si queremos conocer la escala real de la escena debemos conocer las medidas de algunos de los objetos presentes en el entorno.

Los algoritmos que hacen uso de la visión monocular se podrían clasificar de la siguiente forma:

- Basados en características: buscan puntos característicos en la escena y realiza un seguimiento de éstos. El movimiento de la cámara se obtiene minimizando el error de reproyección de los puntos emparejados, utilizando algoritmos iterativos de optimización.

  El punto de fuerte de estos métodos se encuentra en que existen algoritmos de detección de puntos de interés bastante robustos, pero por contra dependerá mucho de los umbrales que se impongan y su precisión puede verse afectada por emparejamientos incorrectos.

- Métodos directos: en lugar de buscar de puntos característicos, utiliza la información de la intensidad de los píxeles para determinar el movimiento del sensor.

  Estos métodos funcionaran mejor en aquellas situaciones en las que el fondo no posea mucha textura (es más díficil buscar esquinas) y, además el tiempo de cálculo se verá reducido ya que no se tiene que extraer puntos y calcular sus emparejamientos.

- Métodos híbridos: son métodos que combinan ambos sistemas.

3.1.2 Odometría visual con visión estéreo

Los algoritmos en la visión estéreo son similares a los explicados en la visión monocular, con la diferencia de que ahora tenemos dos cámaras que toman imágenes de forma simultánea y que comparten una parte de su campo de visión.

La principal ventaja que tiene esta odometría es que al poseer dos cámaras podemos estimar la profundidad a la que se encuentra los objetos mediante geometría epipolar[4].

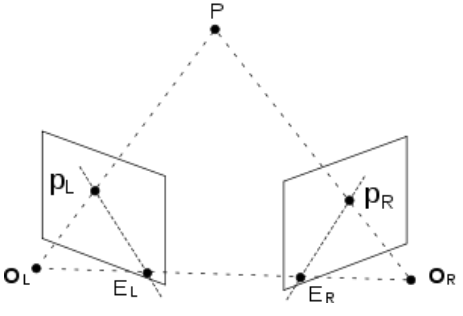

Figure 4: Modelo de cámara Pin Hole.

Esta odometría visual se ha utilizado en el campo de la robótica tanto para la navegación [referencia] como para algoritmos SLAM

3.2 SLAM

Los algoritmos conocidos como SLAM (*Simultaneous Localization and Mapping*) son aquellos capaces de generar un mapa y localizar al sensor de forma simultánea. Estos algoritmos se han utilizado en el campo de la robótica típicamente con láseres, sónares o cámaras estéreos pero con el paso del tiempo se desarrollaron algoritmos que utilizasen un solo sensor.

3.2.1 Mono-SLAM

El término Mono-SLAM hace referencia al uso de una sola cámara para RGB para la localización y creación de un mapa de forma simultánea. El primero que propuso el uso de una única cámara Andrew Davison a partir del año 2002 [1].

Éste se encuentra basado en el uso de un filtro extendido de Kalman (EKF) para estimar la posición y orientación 3D de la cámara, así como la posición de una serie de puntos 3D que conforman el mapa. Para calcular la posición inicial de la cámara es necesario dotar al filtro de información a priori con la posición 3D de al menos cuatro puntos. Una vez estimada la

posición, el algoritmo es capaz de situar la cámara 3D y de generar nuevos puntos del mapa que sirven como apoyo a la propia localización de la cámara.

Filtro kalman extendido El vector de estado del EKF se compone de la posición 3D de la cámara, su velocidad lineal, la rotación y su velocidad angular. Por otra parte, el modelo de transición actualiza el estado de la cámara aplicando un modelo de velocidad constante a la posición y orientación de la cámara teniendo en cuenta las velocidades lineal y angular de la iteración anterior.

Además el modelo de observación se compone de las proyecciones de cada uno de los puntos 3D en el plano imagen teniendo en cuenta la posición y orientación de la cámara.

3.2.2 PTAM

El algoritmo PTAM (*Parallel Tracking and Mapping*) ,escrito por Georg Klein y David Murray [7],pretende solucionar los problemas que tiene el algoritmo de MonoSLAM pero desde un punto de vista totalmente diferente. MonoSLAM estima y actualiza su localización a través de un filtro de Kalman extendido, mientras que PTAM utiliza métodos de optimización.

El principal problema del algoritmo de MonoSLAM es que su tiempo de ejecución aumenta exponencialmente con el número de puntos del mapa. Esto se debe a que en cada iteración, se actualiza la posición de cada objeto del mapa y la localización de la cámara.

PTAM parte de la premisa que el hilo de tracking debe ejecutarse en tiempo real, mientras que el del mapa no tiene por qué actualizarse en cada iteración.

Tracking El hilo del tracking sigue los siguientes pasos:

1. Preprocesado de las imágenes: La imagen se divide en 4 subimágenes de distinta resolución y se buscan puntos característicos con el algoritmo FAST de Rosten, Edward y Drummond, Tom [12].

2. Actualización de la posición con modelo de movimiento: A continuación se actualiza la posición con un modelo de velocidad constante[5], como se muestra a continuación:

$$v_t = \frac{x_t - x_{t-1}}{\Delta t} \tag{5}$$

3. Actualización de grano grueso: se seleccionan una número n de puntos 3D del mapan y se proyectan en la cámara. A continuación se realiza un emparejamiento comparando parches alrededor de los píxeles que se han proyectado. Una vez que se encuentren emparejados se minimiza el error de proyección a través del proceso de optimización de Gauss-Newton Kelley [6]

4. Actualización de grano fino: En esta ocasión se toman hasta mil puntos del mapa y se sigue el mismo procedimiento que en el paso anterior. Al tomar un mayor número de puntos se obtiene una estimación con mayor precisión.

Mapping Por otro lado, el hilo del *Mapping* debe inicializarse primero para su correcto funcionamiento. Ésta se realiza con el seguimiento de una serie de imágenes tomadas a un cierta distancia, emparejando puntos característicos y calculando su posición con el algoritmo de cinco puntos. Cabe decir que el mapa que se creará puede no tener una posición 3D totalmente real, pero representa la realidad a una escala diferente.

Dicho hilo sigue los siguientes pasos:

1. Añadir nuevos Keyframes: no todos los frames son utilizados para procesarlos ya que supondría una gran carga computacional y ,además, puede que contengan información redudante. Por ello, sólo se añadirán si cumplen una serie de condiciones: que el tiempo entre Keyframe y Keyframe supere un cierto valor, que su distancia sea adecuada, y que la estimación de la posición sea bueana.

2. Añadir nuevos puntos 3D al mapa: una vez añadido el *Keyframe* al mapa, se calcula una serie de puntos característicos y se buscan en anteriores Keyframes. Los puntos emparajedos se añaden al mapa, calculando su posición mediante triangulación. Dichos puntos son los que se proyectarán en el hilo de *tracking*.

3. Optimizar el mapa: Si el hilo del mapa se encuentra libre, se refina la posición 3D de los puntos guardados mediante el algoritmo de *Bundle Adjustment* con el fin de minimizar el error de proyección entre *m* posiciones en 3D de la cámara y *n* puntos 3D.

4. Mantenimiento del mapa: Finalmente, se elimina aquellas asociaciones de puntos mal realizadas y añadiendo nuevos puntos cuando sea posible.

El principal problema que posee el algoritmo de PTAM es que como el título de su paper dice sólo se puede utilizar en pequeñas zonas de trabajo. El numero de puntos que debe procesar crece con bastante rapidez provocando un mayor tiempo de cálculo.

Comparativa entre MonoSLAM y PTAM: El autor del paper intenta comparar su algoritmo con otros(EKF-SLAM), en un seguimiento de una trayectoria. Como podemos ver en la figura [5] el algoritmo PTAM ofrece una trayectoria con más precisa.

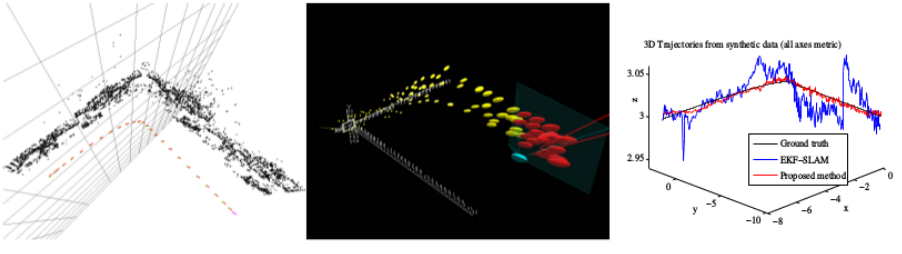

Figure 5: Comparativa entre MonoSLAM y PTAM: La imagen de la izquierda muestra el mapa creado por PTAM, mientras que la segunda muestra el mapa creado por EKF-SLAM.

3.2.3 DTAM

DTAM (*Dense Tracking and Mapping*) escrito por A. Newcombe et al. en el 2011 [10] aborda el problema del SLAM monocular usando la información de todos los píxeles en

lugar de puntos caracterísiticos. Al utilizar toda la información de la imagen se consigue un mapa denso de la escena que conllevará una gran carga computacional.

Los autores del paper hacen hincapié DTAM es capaz de trabajar en tiempo real en equipos potentes debido a que el algoritmo es paralelizable y permite su ejecución en GPU.

Tracking En primer lugar,la inicialización de la pose se realiza de una manera similar que el algoritmo de PTAM [3.2.2] (actualización de grano grueso y fino) pero en lugar de minimizar el error de reproyección se minimiza el error fotométrico.

Mapping El hilo del *mapping* se ocupa de la generación del mapa mediante el uso de varias vistas. El proceso de reconstrucción se realiza con todos los fotogramas de entrada, y mediante un proceso que pretende reducir la energía de la imagen se refina la generación del mapa.

Comparativa entre PTAM y DTAM DTAM ofrece unos resultados mas satisfactorios a la hora del tracking que el algoritmo de PTAM debido al mapa denso. El autor del paper lo intenta demostrar siguiendo a un objeto (taza) y agitando la cámara. Como podemos ver en la figura [6] en los momentos en los que agita la cámara (a partir del frame 1000) el algoritmo PTAM pierde la taza.

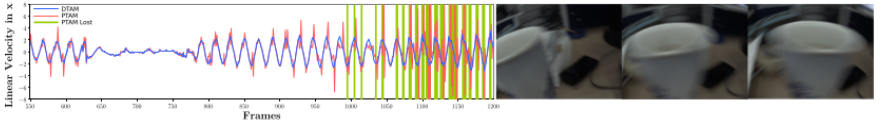

Figure 6: Comparativa entre PTAM y DTAM. La primera grafica muestra como evoluciona el seguimiento de la taza a lo largo de los frames. La segunda imagen muestra como agita la cámara distorsionando la imagen

3.2.4 ORB-SLAM

El algoritmo ORB-SLAM [9] escrito por Mur-Artal, R. y Montiel, J.M.M. y Tardos, J.D, es un algoritmo SLAM basado en la búsqueda de puntos característicos que puede funcionar con visión monocular, visión estéreo y con sensores RGBD. Su peculiaridad se encuentra en el uso del descriptor ORB [13] para encontrar los puntos de interés y un modelo de bolsa de palabras [3] para detectar cierres de bucle y, en su caso, para relocalizarse.

A diferencia de los SLAM explicados, éste posee un hilo más que se usa para detectar cierres de bucle: *looping*

Tracking Al igual que en otros algoritmos se estima su posición mediante el uso de un modelo de movimiento emparejando los puntos 3D visibles en el fotograma anterior, calculando la posición actual a partir de los emparejamientos realizados.

Pero en esta ocasión si el robot se pierde se utiliza un modelo de bolsa de palabras para obtener Keyframes candidatos que concuerden con la observación actual.

Mapping La inicialización del mapa se realiza calculando en paralelo un mapa por homografía y otro mediante una matriz fundamental. Ambos modelos obtienen una puntuación para determinar cuál será el utilizado para inicializar el mapa, de forma que en aquellas escenas que cuenten con un plano principal se utilizará la homografía y en el resto de los casos se utilizará la matriz fundamental.

Otra caracterítica de este algoritmo es que genera un grafo donde los vértices se corresponden con Keyframes y las aristas entre vértices se generan siempre que los Keyframes tengan varios puntos 3D en común. Este grafo se utiliza entre otras cosas para eliminar Keyframes redundantes

Looping Finalmente el hilo *Looping* se encarga de comprobar si se ha producido un cierre de bucle. Para ello. utiliza el grafo de Keyframes conectados y el modelo de bolsa de palabras, con el objetivo de buscar Keyframes candidatos con apariencia similar a la observación actual.

References

[1] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29:2007, 2007.

[2] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL http://doi.acm.org/10.1145/358669.358692.

[3] Dorian Gálvez-López and Juan D. Tardos. Bags of binary words for fast place recognition in image sequences. 2012.

[4] Joel A. Hesch and Stergios I. Roumeliotis. *A Direct Least-Squares (DLS) method for PnP*, pages 383–390. 2011. ISBN 9781457711015.

[5] Chung-Nan Lee Karsten Ottenberg Bert M. Haralick and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. International journal of computer vision, 1994.

[6] Carl T. Kelley. Iterative methods for optimization. 1999.

[7] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[8] leg Naroditsky David Nistér and James Bergen. Visual odometry. Computer Vision and Patter Recognition, 2004.

[9] R. Mur-Artal, J.M.M. Montiel, and J.D. Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. 2015.

[10] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2320–2327, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126513.

[11] Bernard Conio Olivier Leboulleux Radu Horaud and Bernard Lacolle. An analytic solution for the perspective 4-point problem. Computer Vision, Graphics, and Image Processing, 1989.

[12] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33832-2, 978-3-540-33832-1. doi: 10.1007/11744023_34.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-1101-5. doi: 10.1109/ICCV.2011.6126544.