Algoritmos de navegación para robots móviles usando visión estéreo

Agustín Olmedo,*Estudiante, UBA*
Amit Stein,*Estudiante, UBA*,
y Fernando Bugni, *Estudiante, UBA,*

Abstract—El presente trabajo desarrolla una implemetación para la navegación de un robot utilizando visión estéreo. Al mísmo utiliza conceptos sobre vision computacional: calibración estéreo, mapa de disparidad, rectificación. La implementación fue desarrollada integramente en C++ utilizando la libreria OpenCV, además de otras librerias para la movilidad del robot.

Index Terms—Computer Society, LaTeX, Visión Computacional, OpenCV, Mapas de Disparidad

1 Introducción

La navegación de robots es el área encargada de desarrollar robots capaces de desplazarse de forma autónoma. Esta tecnología se utiliza en distintos ámbitos como por ejemplo: misiones espaciales, misiones militares, asistencia para personas con discapacidades e inclusive como guias en una exposición.

En particular en el presente trabajo desarrollaremos un algoritmo de navegación de robots utilizando visión estéreo. En la actualidad, existen diversas técnicas de visión por computadora que nos brindan la información necesaria sobre el entorno. Entre ellas se encuentra la visión estéreo que utiliza dos cámaras para obtener datos de los puntos en tres dimensiones. Algunas aplicaciones de estas técnicas son: construcción de mapas de un entorno desconocido, localización, reconstrucción 3D o también reconocimiento de objetos.

El enfoque de este trabajo consiste, principalmente, en utilizar visión estéreo para calcular el mapa de disparidad asociado a la escena capturada por las dos cámaras. Con la información de disparidad podemos obtener la distancia entre el robot y los objetos que luego será utilizada por el robot para decidir el próximo movimiento a realizar.

Un paper que ha influido fuertemente el presente trabajo fue "Region of Interest in Disparity Mapping for Navigation of Stereo Vision Autonomous Guided Vehicle". El mismo posee un enfoque similar en cuanto a las técnicas de visión utilizadas. Las diferencias radican en la interpretación de la información que nos brindan los mapas de disparidad.

La organización del paper sigue el orden cronológico en que fuimos aplicando los distintos pasos, empezando con la calibración y calibracón estéreo, luego explicamos los conceptos necesarios para entender de que forma utilizamos la visión estéreo para obtener la información necesaria para realizar el algoritmo de navegación propuesto; continuando con la explicación del algoritmo de navegación. Por último, se incluye una sección de experimentos donde se explica brevemente los experimentos realizado, la arquitectura del robot que se ha usado en los experimentos y las limitaciones del algoritmo.

2 Calibración de la cámara estéreo

Cuando capturamos una escena del mundo (tres dimensiones) con una cámara, ésta se proyecta sobre una imagen en dos dimensiones.
Para conocer la posición de cada punto es necesario definir un sistema de referencias, es decir, los tres ejes principales con su origen, sus orientaciones y sus unidades.

Existen tres sistemas de referencias posibles:

- El sistema del mundo
- El sistema de la cámara
- El sistema de la imagen

La unidad del sistema de referencia de la imagen es el pixel, mientras que en los demás es el metro. El sistema de la cámara tiene el origen en el centro óptico y el de la imagen en la esquina de la misma.

Los parámetros intrínsecos son los que relacionan el sistema de referencia de la cámara con el de la imagen y los extrínsecos son los que relacionan el sistema de referencia del mundo con el de la cámara.

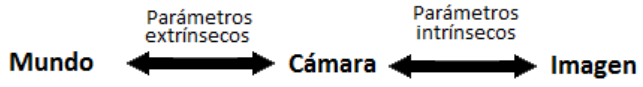


Fig. 1. Relación entre sistemas de referencias

La conversión del sistema de referencia del mundo al de la cámara implica una traslación y una rotación.

Un punto en el mundo, visto desde la cámara, se proyecta en el plano de la imagen. Geometricamente podemos calcular la relación entre el punto en el mundo y el punto proyectado en el plano de la imagen(Ver

figura 2). Además se traslada el origen del plano de la imagen llevandolo al centro, y se realiza un cambio de unidad dividiendo por el número de pixels por unidad de distancia.

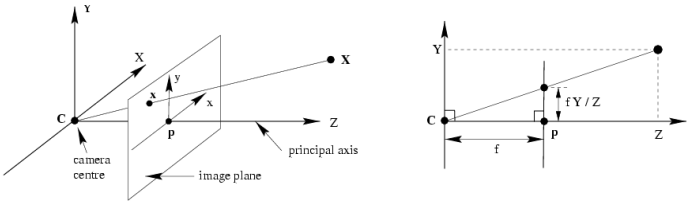


Fig. 2. *Parámetros intrínsecos: El punto en el mundo* $(X, Y, Z)^T$ *visto desde la cámara se proyecta en el plano de la imagen como* $(fX/Z, fY/Z)^T$

Con lo cual los parámetros intrínsecos son: la distancia focal, el punto principal y se agregan el skew (pixel oblicuo), y la distorción de los lentes, etc.

La calibración de una cámara es el procedimiento por el cual obtenemos los parámetros intrínsecos y extrínsecos de la cámara a partir de imágenes tomadas por la misma.

Una vez que obtenemos los parámetros intrínsecos y extrínsecos, podemos decir que la cámara esta calibrada, pudiendo relacionar cualquier punto del mundo real con los puntos proyectados en el plano de la imagen.

Para el desarrollo del presente trabajo utilizamos una camara estéreo, con lo cual no nos alcanza solo con calibrar cada una de las cámaras, sino que además debemos calibrar las dos cámaras de forma conjunta. A esto se lo denomina calibración estéreo.

La calibración estéreo nos provee los parámetros intrínsecos de cada cámara (que son los obtenidos en la calibración de cada una) y los parámetros extrínsecos, que nos definen la posición relativa de las cámaras.

Para realizar dicha calibración utilizamos un toolbox de calibración de cámaras para matlab [1]. Esta herramienta utiliza un patrón plano. El mismo debe contener cuadrados blancos y negros intercalados de un cierto tamaño conocido. El proceso del toolbox consiste en capturar imágenes de este patrón desde distintos angulos y luego marcar las cuatro esquinas del patrón en cada una de ellas.

3 Imagen estéreo

Al proyectarse los objetos, de un espacio tridimensional a una imagen bidimensional se pierde la información de la distancia a la cámara o profundidad (eje Z) de cada punto. Una forma de tratar de recuperar esta información es mediante el uso de dos cámaras, en lo que se conoce como visión estéreo.

A continuación explicaremos la configuración geométrica que se aplica en el campo de visión estéreo. La misma se conoce como geometría epipolar.

1. http://www.vision.caltech.edu/bouguetj/calib_doc/

3.1 Geometría epipolar

La geometría epipolar posee dos sistemas de referencias, uno por cada cámara, y surgen dos puntos importantes llamados epipolos. Un epipolo es la imagen del centro óptico de una de las cámaras con respecto a la otra (ver figura 5).

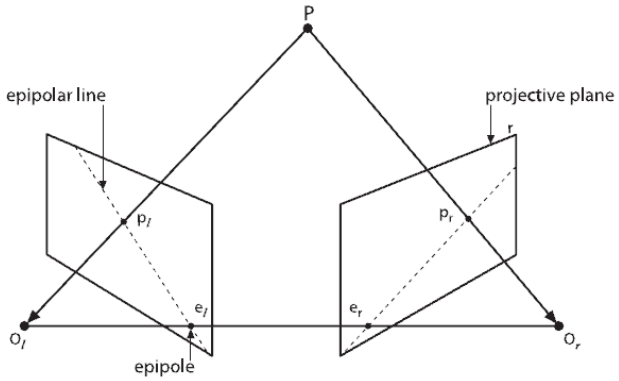


Fig. 3. Geometría epipolar

La línea que intersecta los dos centros ópticos O_l y O_r se llama línea base. Además los epipolos e_l y e_r surgen de la intersección de la línea base con el plano de la imagen de cada cámara.

Los puntos O_l, O_r y P definen un plano llamado plano epipolar. Las proyecciones p_l y p_r así como los epilpolos e_l y e_r pertenecen al plano epipolar.

Las líneas epipolares son las intersecciones del plano epipolar con los planos de las imagen de cada cámara. Para cualquier plano epipolar, las líneas epipolares incluyen los epipolos.

La imagen de las ubicaciones posible de un punto visto en una de las imágenes es la línea que va desde el correspondiente punto y el punto epipolar en la otra imagen. Es decir, viendo la figura 6 el punto X en el mundo va a estar proyectado en la imagen, de la cámara derecha, en la línea epipolar l'. Esta es la utilidad de los epipolos.

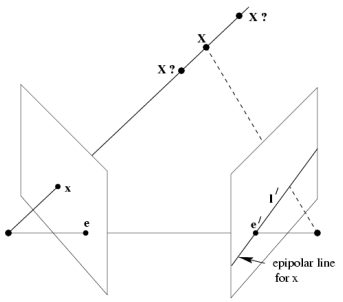


Fig. 4. Geometría epipolar

3.2 Mapas de disparidad

El algoritmo de navegación realizado en el presente trabajo utiliza información de la profundidad que existe

entre los puntos del mundo y las cámaras, precisamente al centro óptico de las mismas.

Una forma de obtener la distancia a la que se sitúa físicamente un objeto en el mundo con respecto a las dos cámaras, consiste en identificar en ambas imágenes aquellos píxeles que se corresponden con la misma entidad física en la escena 3D para luego hallar la distancia que separa estos píxeles. Esto es lo que se conoce como disparidad.

Para calcular la disparidad hemos usado la libreira Libelas[2](Library for Efficient Large-scale Stereo Matching). Básicamente, lo que hace esta librería es buscar los descriptores característicos de cada imagen con una implementación del algoritmo SURF[3](Speeded Up Robust Features) para luego buscar las correspondencias entre los puntos encontrados en el paso anterior. Con lo cual, la disparidad es d = x_l - x_r donde x_l y x_r son las correspondencias entre los puntos de cada imagen.

Luego, conociendo el valor de disparidad y la distancia entre los centros ópticos de las cámaras(línea base) podemos calcular la profundidad de los puntos usando triangulos semejantes (ver figura 3.2).

$$\frac{T-(x_l-x_r)}{Z-f}=\frac{T}{Z}\Rightarrow Z=\frac{fT}{x_l-x_r}\Rightarrow Z=\frac{fT}{d}$$

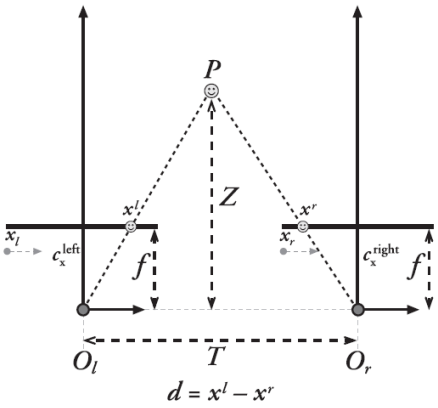


Fig. 5. figura 1:

De la fórmula anterior podemos observar que la distancia de un punto a las cámaras es inversamente proporcional al valor de disparidad (ver figura 3.3). Si el valor de disparidad es cercano a cero, pequeñas diferencias en la disparidad producen grandes diferencias en la profundidad. Y si la disparidad es grande pequeñas diferencias en la disparidad no producen diferencias en la profundidad. La relación de estas dos variables es claramente no lineal.

Con lo cual, si el valor de disparidad es muy pequeño o muy grande la disparidad no nos dice mucho sobre la profundidad a la que se encuentra el objeto de las cámaras. En consecuencia los sistemas de visión estéreo tienen buena resolución de profundidad cuando los objetos se encuentran relativamente cerca.

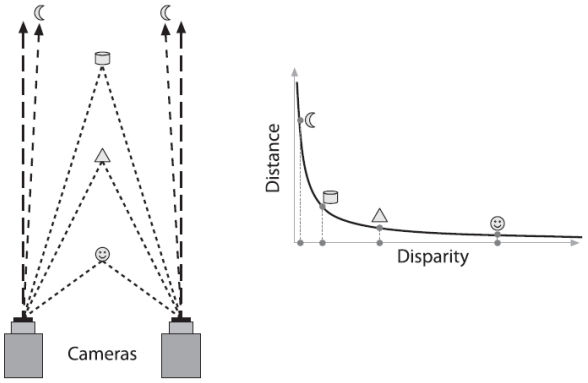
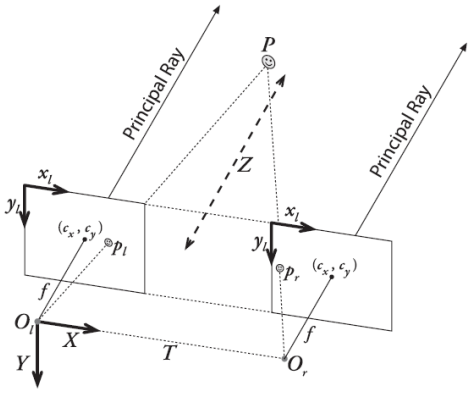


Fig. 6. Figura 1

Cabe aclarar que para conseguir el mejor resultado posible eliminaremos la distorsion radial y tangencial de las imagenes obtenidas con la cámara Minoru. Además se ajusta los ángulos y distancias entre las cámaras para que los planos de las imagenes queden coplanares[4]. A este último paso se lo conoce con el nombre de rectificacion. Estas operaciones las realizamos con la librería OpenCV.



3.3 Reproyección de puntos en 2D

Conociendo los parámetros intrínsecos y extrínsecos de las dos cámaras, podemos obtener la posición absoluta en tres dimensiones de los puntos proyectados.

Como mencionamos anteriormente, para conseguir los mejores resultados posibles rectificamos las imágenes. Cuando openCV realiza la rectificación, a la vez calcula la matriz de reproyección Q.

$$Q=\begin{pmatrix}1&0&0&-c_x\\0&1&0&-c_y\\0&0&0&f\\0&0&-1/T_x&(c_x-c_x^{'})/T_x\end{pmatrix}$$

2. http://www.rainsoft.de/software/libelas.html
3. ftp://ftp.vision.ee.ethz.ch/publications/articles/eth_biwi_00517.pdf
4. Puntos o líneas que se encuentran en el mismo plano.

Donde T_x es la longitud de la línea base, c_x y c_y son las coordenadas del punto principal en la primer cámara, f es la distancia focal y $c_x^{'}$ es la coordenada horizontal del punto principal de la segunda cámara.

Luego podemos proyectar el punto en tres dimensiones realizando la siguiente multiplicación:

$$Q \begin{pmatrix} x \\ y \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

Las coordenadas 3D quedarían:

$$\left(\begin{array}{cccc} X/W, & Y/W, & Z/W \end{array} \right)$$

4 Navegación para robot móviles utilizando los mapas de disparidad

A continuación explicaremos brevemente los pasos realizados desde que se capturan las imágenes hasta que el robot realiza el movimiento calculado.

Como primer paso, en un mismo instante, capturamos con cada cámara la escena del mundo. Luego se calcula el mapa de disparidad, habiendo eliminado la distorsión radial y tangencial de las imágenes; así como habiendo realizado la rectificación de las mismas. Una vez que obtuvimos el mapa de disparidad se realiza una heuristica para encontrar el valor medio de disparidad. Con este valor construimos el punto a reproyectar, obteniendo su posición absoluta en el mundo real. Por último calculamos el ángulo de giro y la distancia a recorrer para ir a esta posición y le indicamos al robot que se movilice a dicho lugar.

A continuación explicaremos en detalle los pasos más importantes del algoritmo.

4.1 Procesamiento del mapa de disparidad

Los valores de disparidad del algoritmo se asignan con el mapa de disparidad que referencia a las coordenadas de la cámara izquierda.

La región de interes es el área dentro de la cual se buscan los obstaculos. Por lo tanto, el algoritmo esta diseñando para ver los obstaculos solo en esta área particular. El tamaño de la región de interes es de 640 x 40 pixels. Esta región esta determinada entre las filas 300 y 340 de pixels, siendo el tamaño de las imágenes de 640 x 480 pixels.

El procedimiento para evitar obstaculos consiste en buscar un punto, en la región de interes, al cual dirigirse que este relativamente cerca del robot y en el que ningún objeto.

se basa en el concepto de la figura 6. El cual nos indica que los valores de disparidad cercanos a cero y los que son muy grandes no nos aportan información útil sobre la distancia que hay del punto a las cámaras.

4.1.1 *Heurística de la disparidad media*

La heuristica presentada para encontrar el valor de diparidad medio,

La heuristica consiste en recorrer la región de interes calculando la mediana de los valores de disparidad de cada columna. Es decir, aplanamos el ROI a una sola fila que contiene en cada celda el valor de disparidad medio de la columna (ver figura). Luego, calculamos la mediana de esta fila consiguiendo el valor de disparidad medio del ROI.

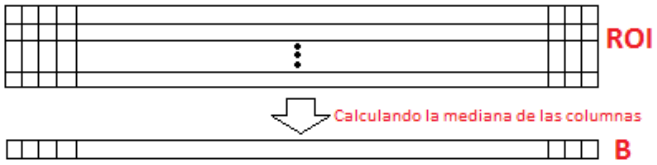


Fig. 7. Heuristica

La heuristica no nos asegura que el punto encontrado sea el mejor al cual dirigirse; pero sabemos que con dicho valor de disparidad, al proyectarlo, nos indicará la distancia correcta en la que se encuentra el punto de las cámaras.

Puede ocurrir que el mapa de disparidad no nos provea información suficiente para decidir a que punto ir, ya sea porque el objeto esta demasiado cerca o porque la imagen tiene poca luz. Esto se ve reflejado en que el ROI aplanado posee más de la mitad de los valores fuera de rango. Sí esto sucede se aplica una politica de escape ya que la informacion provista no es suficiente para determinar una desición. La politica de escape consiste en girar una cierta cantidad de grados hacia la derecha buscando una escena más apropiada para encontrar un punto al cual dirigirse.

4.2 Calculo del ángulo de giro y distancia a recorrer

Como explicamos anteriormente, para reproyectar un punto necesitamos multiplicar Q por el punto a reproyectar que esta definido por:

$$\left(\begin{array}{cccc} x, & y, & d, & 1 \end{array} \right)$$

Con lo cual para reproyectar el punto al que se dirigirá el robot necesitamos definir x e y. x será la coordenada de la primer celda, del ROI aplanado, que contenga el valor de disparidad calculado por la heuristica (comenzando dede la izquierda), mientras que y será la coordenada vertical inferior del ROI.

Luego de encontrar un punto:

$$\left(\begin{array}{cccc} x, & y, & d, & 1 \end{array} \right)$$

lo reproyectamos para obtener el punto P en el mundo real (3D), al cual dirigirse.

Usando propiedades trigonometricas calculamos el ángulo de giro y la distancia.

5 EXPERIMENTOS

El entorno en que probamos el algoritmo de navegación es en ambientes interiores.

El piso no tiene ninguna diferencia de nivel, la luminosidad era bastante importante y los objetos a esquivar fueron cajas grandes con aristas bien definidas.

Si el ambiente es luminoso se consiguen mejores resultados porque la calidad de las imágenes será mejor.

5.1 Arquitectura

La arquitectura utilizada se puede dividir en dos partes: la computadora donde se corre el algoritmo descripto y la mecánica del robot. La computadora utilizada fue una Netbook Asus con sistema operativo Ubuntu Linux. La capacidad de procesamiento fue mas que suficiente para la ejecucion del mismo, entonces podemos afirmar que nuestro algoritmo no posee altas restricciones de performance. La mecanica del robot se basa en un controlador que puede controlar las ruedas. Estas son dos de cada lado unidas por horugas. Para poder comunicarse con esta placa enviamos paquetes udp desde la Netbook. Para la parte de vision utilizamos la camara stereo Minoru que se encuentra conectada por usb al frente de la Netbook.

La conexion de nuestro programa con estas dos interfaces se realiza gracias a librerias. Utilizamos la libreria Libcam que nos permite sacar buenas fotos con esta camara stereo. Para la parte de comunicacion con las ruedas utilizamos la libreria libexabot-remote.

5.2 Limitaciones

La limitacion se basa fuertemente en el entorno que utilizamos. Si no cumplimos con el entorno que se utilizo en experimentos no se va a poder asegurar los resultados obtenidos.

Otra limitacion importante es que los obstaculos no deben moverse, esto hace mas facil que nuestro algoritmo haga su trabajo.

Cabe destacar que el entorno utilizado juega un papel fundamental en el buen desempeño del algoritmo, utilizamos un entorno llano, iluminado para no perder ningún dato al tomar cada fotografía. La cámara se encuentra en la parte superior del frente del robot y su movilidad se reduce a avanzar, retroceder y rotar sobre su propio eje.

6 TRABAJOS FUTUROS

Nuestro trabajo nos provee un buen puntapie para seguir con un desarrollo mas sofisticado. Se podria mejorar en varios aspectos, por ejemplo poder ir armando un mapa visto desde el techo con cada movimiento realizado con el robot. Esto seria muy util porque nos mostraria el caminorealizado por el robot.

Otra mejora podria ser tener un buffer de las imagenes anteriores y poder utilizarlas de alguna forma para tomar la nueva decision de la direccion a seguir.

Otra mejora mas especifica podria ser utilizar un thread aparte para sacar las fotos. Este thread sacaria imagenes constantemente y las dejaria en una pila y cuando el algoritmo quera decidir un nuevo camino para continuar podria desapilar las imagenes que necesite y tomar la desicion.

7 CONCLUSIÓN

El presente trabajo es una idea simple de como poder llevar a cabo un algoritmo de navegacion en una arquitectura definida. Resulto muy interesante ver como entre sucesivos experimentos pudimos mejorar el algoritmo. Tambien nos sirve para abordar muchos mas temas interesantes sobre vision robotica.