Musleh, B. A., Escalera, A. de la, Armingol, J. M. Detección y Localización de obstáculos mediante U-V Disparity con CUDA. *Seminario Anual de Automática, Electrónica Industrial e Instrumentación*. (Badajoz, 6-8 Julio 2011). Universidad de Extremadura, Escuela de Ingenierías Industriales, 2011. pp. 719-722.

Detección y Localización de obstáculos mediante U-V Disparity con CUDA

B. Musleh, A. de la Escalera, y J.M. Armingol

***Resumen*—Tradicionalmente la detección de obstáculos es un tema de investigación de gran interés en visión por computador, aplicada tanto a la navegación de robots, como a los sistemas avanzados de ayuda a la conducción (ADAS). Aunque otras tecnologías como por ejemplo el láser, presentan buenos resultados para detectar obstáculos en entornos urbanos, la visión por computador proporciona información 2D – ó 3D con visión estéreo – que mejora la interpretación del entorno en exteriores. En este artículo se presenta una implementación en tiempo real de la construcción del mapa denso de disparidad y del *U-V disparity*, que son utilizados para la detección y localización de obstáculos. Para minimizar los efectos sobre el tiempo de cómputo que ocasionan tanto la construcción del mapa denso de disparidad, como la del *U-V disparity*, se han implementado ambos mediante el uso de GPUs (Unidades de Procesamiento Gráfico) por su alto rendimiento con algoritmos paralelizables.**

***Palabras Clave*— Visión estéreo, ADAS, CUDA, GPU.**

I. Introducción

En visión por computador es necesario disponer de dos cámaras con ciertas condiciones ideales para que se cumplan las ecuaciones suficientemente conocidas (1) y (2). Estas ecuaciones pueden usarse para obtener la profundidad (W) para un punto $P = (U',V',W')^T$ en coordenadas del mundo respecto de un sistema de coordenadas situado en el sistema estéreo, cuya proyección sobre los planos de imagen es (u_L,v_L) para la cámara izquierda y (u_R,v_R) para la cámara derecha respecto al centro de la imagen.

$$W' = f \cdot B/(u_L - u_R) = f \cdot B/d \quad (1)$$

$$v_L/f = V'/W';\ \ v_R/f = V'/W' \Rightarrow v_L = v_R \quad (2)$$

Donde d es la disparidad, f es la distancia focal en píxeles y B es la distancia *baseline*. Las condiciones requeridas para las dos cámaras son difíciles de cumplir en la práctica, por lo que es necesaria una etapa de rectificación. En nuestro caso, se dispone de un sistema estéreo Bumblebee de Pointgrey que proporciona ambas imágenes rectificadas. A partir de éstas, se puede construir tanto el mapa de disparidad como el u-v *disparity* [1] partiendo del anterior. El mapa de disparidad representa la profundidad (W') de cada punto de la imagen, mientras que el v-*disparity* representa el histograma de los valores de disparidad para cada fila del mapa de disparidad y el u-*disparity* es igual, pero para cada columna. Los obstáculos en frente del vehículo y situados perpendicularmente a éste, aparecen como líneas horizontales en el u-*disparity* y como líneas verticales en el v-*disparity* [2] en sus correspondientes valores de disparidad. Otra característica interesante es que el suelo que se encuentra delante del vehículo aparece como una línea oblicua en el v-disparity, lo cual es muy útil porque permite calcular en todo momento tanto el ángulo de cabeceo como la altura a la que se sitúa la cámara estéreo [3]. Una aportación de este artículo consiste en poder localizar los obstáculos con mayor resolución que utilizando únicamente los valores de disparidad. Para ello se hace uso del perfil de la calzada [1], como se verá en el apartado de localización de obstáculos (sección IV).

Tanto en el caso de la construcción del mapa de disparidad como en la del u-v *disparity*, el tiempo de cómputo es elevado. Con objeto de reducirlo, se ha implementado la construcción de ambos utilizando NVIDIA CUDA [4], usado para procesar en GPUs, lo cual permite utilizar la potencia de las tarjetas gráficas en la realización de tareas de propósito general [5].

La sección II de este artículo describe la construcción del mapa de disparidad y del u-v *disparity* utilizando CUDA. En la sección III se describirá el sistema de detección de obstáculos y el método para la localización de estos se explicará en la sección IV. Finalmente, se presentarán los resultados experimentales del algoritmo en la sección V y las conclusiones del artículo en la secciónVI.

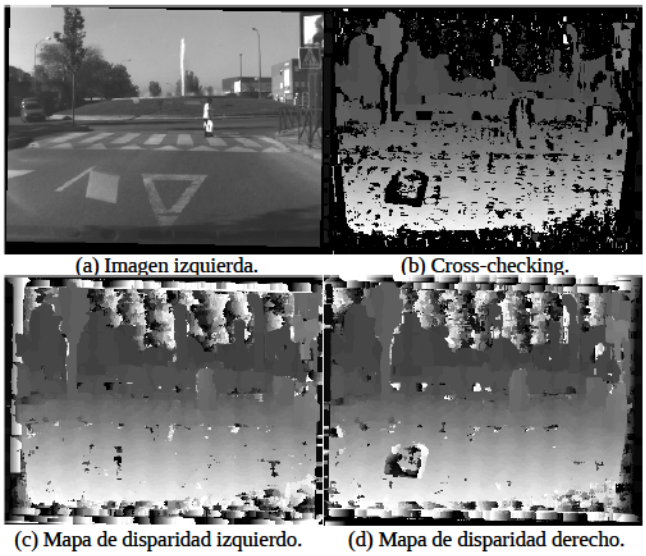
II. Construcción del Mapa de Disparidad y del U-D Disparity usando CUDA

Como se ha explicado anteriormente, el mapa denso de disparidad representa la profundidad (W') de cada punto de la imagen, y como muestra (1) para determinar la disparidad ($d = u_L - u_R$) es necesario determinar la correspondencia entre puntos de la imagen izquierda y derecha respectivamente, lo que se denomina problema de correspondencia estéreo. En el caso de que ambas imágenes estén correctamente rectificadas, este problema es un problema 1D. El algoritmo sigue la taxonomía presentada por Scharstein y Szeliski [6] para resolver el problema de correspondencia estéreo, donde se propone que este problema se realiza mediante 4 etapas: la primera es el cálculo de la función de coste [7] y donde se ha utilizado el método Diferencias Cuadradas (SD) por ser más rápido y sencillo de implementar en CUDA. La segunda etapa es el cálculo del coste de agregación, que se realiza mediante ventanas cuadradas de agregación. Aunque existen

Manuscrito recibido el 14 de Enero, 2010. Este trabajo ha sido subvencionado por la CICYT a través del proyecto FEDORA (TRA2010-20225-C03-01) y el proyecto VIDAS-DRIVER (TRA2010-21371-C03-C02) y por la CAM a través del proyecto SEGVAUTO-II.

Los autores son mienbros del Laboratorio de Sistemas Inteligentes del Departamento de Sistemas y Automática en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid. Leganés, Madrid. (e-mails: bmusleh, escalera, armingol)@ing.uc3m.es

implementaciones más precisas, el tiempo de cómputo aumenta, siendo nuestro resultado suficientemente preciso utilizando ventanas cuadradas. En la tercera etapa se procede al cálculo de la disparidad, mediante la utilizacion del método local WTA (*Winner-Take-All*) donde, para la posterior etapa de refinamiento de la disparidad, se ha construido el mapa de disparidad para la imagen izquierda (mapa de disparidad izquierdo Fig. 1(c)) y para la imagen derecha (mapa de disparidad derecho Fig. 1(d)) evitando cálculos redundantes. La última etapa es el refinamiento de la disparidad, implementándose un *cross-checking* para reducir los posibles errores en el mapa de disparidad (Fig. 1(b)).



(a) Imagen izquierda. (b) Cross-checking.

(c) Mapa de disparidad izquierdo. (d) Mapa de disparidad derecho.

Fig. 1. Ejemplo de la construcción del mapa denso de disparidad en entornos urbanos.

Nuestra implementación se basa en el trabajo presentado por Stam en [8], al cual se ha añadido: el preprocesamiento usando la LoG (Laplaciana de la Gausiana), la obtención de ambos mapas de disparidad (izquierdo y derecho) evitando realizar cálculos redundantes, el *cross-checking* y la generación del u-v disparity. En la implementación se han utilizado imágenes de tamaño 640x480 en niveles de gris, las cuales pueden ser procesadas utilizando dos GPUs diferentes. La primera de ellas es una NVIDIA Geforce 9300 GS, típica de ordenadores portátiles y una segunda más potente, la NVIDIA Quadro FX 380 LP.

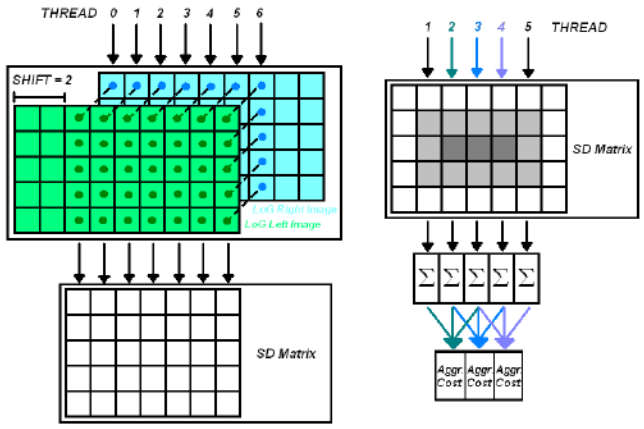
A. *Preprocesamiento mediante LoG*

La primera tarea es la construcción de la LoG sobre ambas imágenes. Las dos imágenes de entrada son almacenadas en la memoria de texturas de la tarjeta gráfica, siendo esta memoria de solo lectura y, en consecuencia, más rapida que el resto y puede ser direccionada usando coordenadas matriciales como las de la imagen. La implementación más sencilla para la obtención de la LoG emplea un hilo para cada píxel de la imagen, el cual lee los datos necesarios de la memoria de texturas y computa la convolución con el filtro, accediendo así varios hilos a las mismas zonas de memoria. Para reducir estos conflictos, la convolución se obtiene siguiendo estos tres pasos:

1) Se usa un hilo para cada píxel, pero ahora este hilo calcula primero la convolución de la fila correspondiente a su píxel con cada una de las filas del filtro y entonces almacena los resultados en la memoria compartida. En este caso el filtro es simétrico y, por tanto, se pueden evitar cálculos redundantes.
2) Para que todos los resultados parciales del paso anterior se hayan almacenado, antes de continuar se impone una sincronización.
3) Cuando todos los hilos se han sincronizado, cada uno calcula el valor final de la convolución para su píxel, sumando los resultados parciales de la convolución de las filas vecinas anteriormente almacenadas en la memoria compartida.

B. *Cálculo de la función de coste*

Una vez que ambas LoG han sido obtenidas y almacenadas en la memoria de texturas, se puede pasar a calcular la función de coste SD, almacenando el resultado en una matriz. Esta operación y las restantes hasta la etapa de refinamiento, deben repetirse para cada posible valor de disparidad, desplazando las LoG a lo largo del eje x. Como muestra la Fig. 2(a), se utiliza un hilo para cada columna de píxeles, obteniendo la SD resultante de ambas imágenes para un valor determinado de disparidad (desplazamiento).



(a) Esquema cálculo func. de coste. (b) Esquema *coste de agregación*.

Fig. 2. Esquema de la implementación del mapa de disparidad con CUDA.

C. *Cálculo del coste de agregación*

El coste de agregación se calcula fila a fila y en dos etapas: en primer lugar, se usa un hilo para sumar el SD para cada columna de la ventana de agregación y se almacena el resultado en la memoria compartida, lo que permite que estén accesibles para el resto de hilos. En segundo lugar, se utiliza un único hilo para obtener el valor final del coste de agregación para cada píxel, sumando el coste de agregación acumulado de cada una de las columnas vecinas dentro de la ventana de agregación correspondiente al píxel. La ventaja de proceder de esta manera es que el coste acumulado de las columnas puede ser reutilizado para obtener el coste de agregación para varios píxeles. Se presenta un esquema en la Fig. 2(b).

Una vez que todos los píxeles de una fila han sido procesados, la siguiente fila se calcula utilizando el concepto

presentado en [9], que consiste en sustraer del coste de agregación de la fila anterior, el coste de agregación correspondiente a la primera fila de la ventana de agregación anterior y añadir el coste de agregación de la última fila de la nueva ventana de agregación. De esta manera se ahorra una ingente cantidad de cálculos redundantes que, aparecerían si no se utilizara esta optimización.

D. Cálculo de la disparidad

Cuando el coste de agregación ha sido calculado para cada píxel con un valor concreto de disparidad (desplazamiento), el hilo verifica si este nuevo valor de coste de agregación es el mínimo y, en ese caso, almacena este valor como el nuevo valor mínimo de coste de agregación, así como el valor de disparidad asociado a éste en dos matrices situadas en la memoria global para posteriores comparaciones. A fin de evitar cálculos innecesarios, el coste de agregación con coordenadas (u,v) en el mapa de disparidad izquierdo es el mismo coste de agregación con coordenadas (u – d, v) para el mapa de disparidad derecho. El tiempo de cómputo se ha reducido en aproximadamente un 35% frente a la implementación original de Stam al utilizar esta optimización.

E. Refinamiento de la disparidad

Los pasos previos deben repetirse para cada posible valor de disparidad y cuando se han obtenido los mejores valores de disparidad para cada mapa de disparidad (izquierdo y derecho), ya se puede realizar el *cross-checking* entre ambos mapas. La implementación consiste en usar un hilo por cada píxel que compare los valores de cada mapa, si estos coinciden se conserva su valor y, en caso contrario, se fija a 0 el valor de ese píxel.

F. U-V Disparity

Como se ha comentado anteriormente, para obtener el u-*disparity* hay que realizar el histograma a cada columna del mapa de disparidad y, en el caso del v-*disparity*, a cada fila. A tal efecto se va a utilizar un hilo para cada columna o fila. Este hilo calcula el respectivo histograma y cuando ha finalizado, comprueba la aparición de posibles desbordamientos, almacenando el resultado en la memoria global.

G. Resultados

Se han realizado diversos tests variando el valor máximo de disparidad y el tamaño de la ventana de agregación. Estas dos variables de configuración tienen una gran influencia en el tiempo de cómputo. Se presenta un resumen de los resultados usando la tarjeta gráfica GeForce 9300 GS, en la tabla 1.

Tanto la disparidad máxima como el tamaño de la ventana de agregación incrementan el tiempo de cómputo, pero resulta interesante que el aumento del valor de disparidad máxima afecta en mayor medida que el incremento del tamaño de la ventana de integración. Esto es debido a que la implementación para el cálculo del coste de agregación anteriormente descrita reduce los cálculos redundantes. Por ejemplo, para un incremento desde un tamaño de ventana de 11x11 a 21x21 implica aproximadamente 2.5 veces más píxeles a ser procesados y produce únicamente un incremento de 0.15 veces aproximadamente en el tiempo de cómputo.

Support Region / Max Disparity	11x11	15x15	21x21
30	0.122 s	0.129 s	0.140 s
40	0.154 s	0.162 s	0.179 s
50	0.185 s	0.197 s	0.217 s

Tabla. 1. Tiempos de cómputo para la construcción del mapa de disparidad y del u-v disparity para diferentes tamaños de la ventana de agregación y de la disparidad máxima usando la NVIDIA Geforce 9300 GS.

La tabla 2 presenta los tiempos de cómputo para cada una de las etapas de la construcción, tanto del mapa de disparidad como la del u-v *disparity* utilizando las dos tarjetas gráficas de las que se dispone. La configuración final se ha fijado en un valor de 30 para la disparidad máxima y un tamaño de ventana de agregación de 17x17. Usando la tarjeta gráfica más potente de las dos, es posible computar en tiempo real con un *frame rate* de 20 imágenes por segundo.

Proccesing time (ms)	CPU-GPU Copy	LoG	Disp Maps	Cross Checking	u-disp	v-disp	GPU-CPU Copy	**TOTAL TIME**
GeForce 9300 GS	0,60	6,37	107.5	1,72	0,94	1,85	0,7	119,7 (ms)
Quadro FX 380 LP	0,42	2,85	43,3	0,38	0,89	1,88	0,2	49,9 (ms)

Tabla. 2. Tiempos de cómputo para cada etapa del mapa de disparidad y del u-v disparity para dos tarjetas gráfica diferentes.

Por último, a fin de comparar el procesamiento en GPU frente al de CPU, se ha implementado la misma construcción del mapa de disparidad en la CPU. El tiempo de cómputo con un Core2Duo 2Ghz y 1GB de RAM es aproximadamente 15 veces superior que el tiempo de cómputo utilizando la GPU NVIDIA Quadro FX 380 LP.

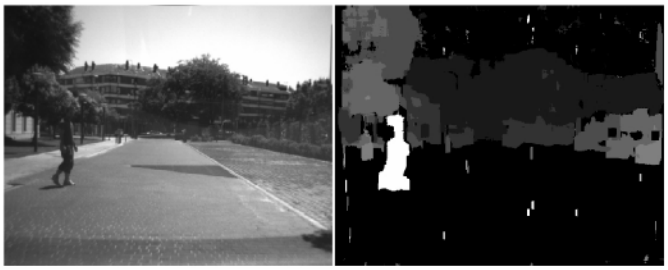
III. Detección de obstáculos

Este módulo del algoritmo es el encargado de detectar todos los obstáculos que se encuentran delante del vehículo. El resultado principal de este módulo será lo que se ha denominado el mapa de obstáculos y que corresponde a todos aquellos píxeles del mapa de disparidad que pertenecen a obstáculos, siendo nulo el valor del resto píxeles (Ver Fig. 3(b)). A partir de dicho mapa, se determinarán las posibles regiones de interés (ROI) sobre la imagen izquierda visible y que se podrían utilizar en una posterior etapa clasificatoria. La obtención del mapa de obstáculos nos permite decidir en cada momento hasta qué distancia en términos de disparidad se van a buscar los obstáculos denominándose region de estudio esta zona delante del vehículo. Además, en zonas cercanas al vehículo es plausible asumir una geometría plana de la calzada. El sistema de detección de obstáculos está dividido en los tres pasos siguientes:

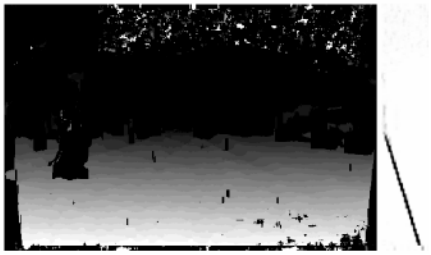
1) El primer paso consiste en una detección preliminar de obstáculos sobre el u-*disparity* basada en la umbralización, de tal manera que se detecten aquellos obstáculos que posean una altura medida en píxeles

mayor a un umbral, obteniendo un u-*disparity* umbralizado.

2) Para cada punto umbralizado del u-*disparity* en el paso anterior cuyas coordenadas son (u , d), se analizan todos los puntos del mapa de disparidad correspondientes a la columna situada en la coordenada u y aquellos píxeles cuyo nivel de disparidad difiera de d se sitúan a cero, manteniendo los que posean un valor igual a d. De esta manera se obtiene el mapa de obstáculos.
3) El tercer paso consiste en determinar hasta qué distancia en términos de disparidad se van a buscar obstáculos y, como se ha comentado anteriormente, es la región de estudio. Para disponer exclusivamente de los obstáculos situados dentro de esta región, se va a realizar una umbralización sobre el mapa de obstáculos de tal manera que para aquellos píxeles cuya disparidad sea mayor a la fijada como umbral se situarán a uno y el resto a cero, obteniéndose así una imagen binarizada. Por último y para determinar las regiones de interés, se realizará un análisis de blobs de la imagen binarizada, obteniendo las dimensiones y posiciones de los rectángulos que engloban a cada una de los obtáculos. Para evitar que obstáculos alejados del vehículo y a su vez cercanos entre sí se engloben en la misma ROI, se obtienen sus bordes utilizando el mapa de obstáculos y se sustraen de la imagen binarizada antes del análisis de blobs.



(a) Imagen visible izquierda. (b) Mapa de obstáculos.

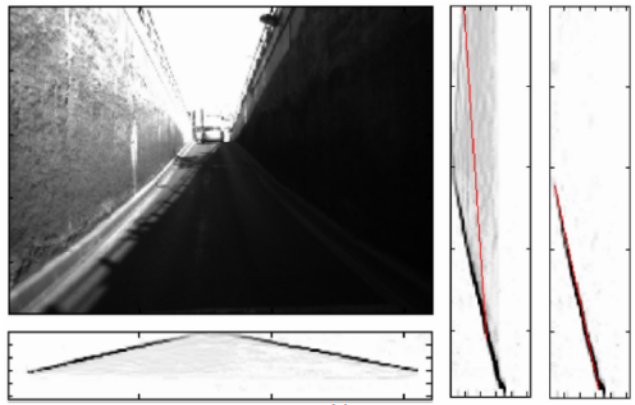


(c) Mapa libre de obstáculos y su correspondiente perfil de la calzada.

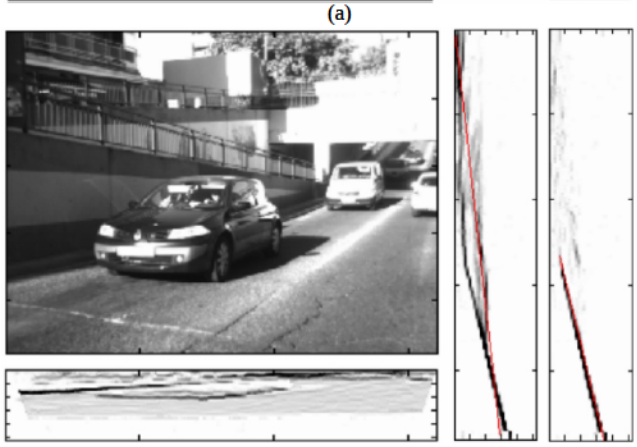
Fig. 3. Ejemplo del mapa de obstáculos en un entorno urbano.

IV. Localización de Obstáculos

Uno de los problemas de la determinación de la profundiadad usando la visión estéreo es la falta de resolución existente, que en nuestro caso es de 30 valores para una distancia entre 3.5m e infinito. Para intentar mejorar la resolución se va a determinar el perfil de la calzada delante del vehículo y, a partir de éste, se obtendrá la localización de los obstáculos con una mayor resolución que usando únicamente los valores de disparidad. Esta mejora solo es aplicable a aquellos obstáculos que se encuentren sobre la calzada, es decir, no estén elevados como es el caso de las señales de tráfico, pórticos o el techo de los túneles. Determinar si un obstáculo está elevado o no, nos podría servir para discernir si el vehículo podría pasar por debajo del obstáculo o no (túneles y pórticos).



(a)



(b)

Fig. 4. Ejemplos de la obtención del perfil de la calzada (línea roja) en entornos urbanos donde aparecen una gran cantidad de obstáculos. Imagen visible con el u-disparity en su parte inferior. En el lado derecho aparece en primer lugar el v-disparity convencional y más a la derecha, el construido a partir del mapa libre de obstáculos.

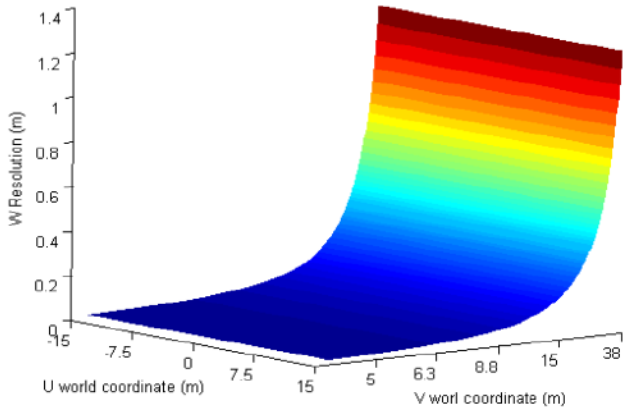
A. Determinación del perfil de la calzada

Como se ha indicado anteriormente, el perfil de la calzada aparece como una línea oblicua en el v-*disparity*. Sin embargo, como se ha comprobado en nuestros tests en entornos urbanos, cuando aparecen una gran cantidad de obstáculos en el v-disparity (líneas verticales) se dificulta la extracción del perfil de la carretera [10][11] (Fig. 4). Con objeto de mejorar la efectividad del algoritmo, se ha construido un v-disparity alternativo a partir de un mapa de disparidad que denominamos mapa libre de obstáculos (Fig. 3(c)) y que corresponde al mapa de disparidad del que se han eliminado todos aquellos píxeles pertenecientes a obstáculos. Si aplicamos sobre el nuevo v-disparity la transformada de Hough, obtendremos el perfil de la carretera con la expresión (3) y que relaciona la disparidad con la coordenada de imagen v, donde m es la pendiente de la recta y b es la ordenada en el origen, que corresponde al valor teórico del horizonte.

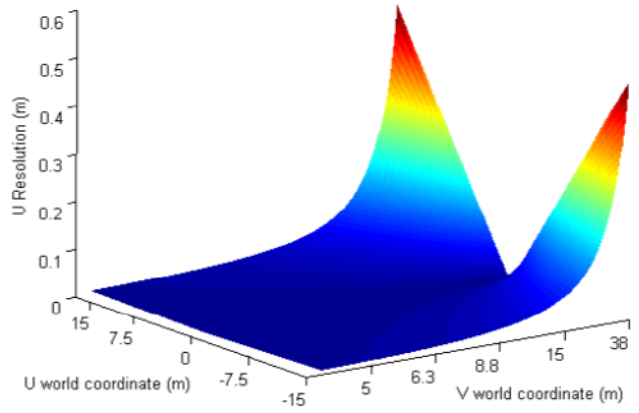
$$v = m \cdot d + b \tag{3}$$

B. Obstáculos elevados

El uso del perfil de la calzada para aumentar la resolución de la localización de obstáculos es aplicable a aquellos que estén en contacto con la calzada, por lo que es necesario implementar un criterio discriminador entre obstáculos elevados o no elevados y que es otra oportación de este trabajo. El criterio que se ha seguido consiste en determinar el punto de contacto entre el obstáculo y la calzada, que vendrá definido por unas coordenadas de imagen (u_{PC},v_{PC}), y utilizando la ecuación (3) se obtiene d_C que corresponde al valor de disparidad que debería tener idealmente el obstáculo si se encontrara sobre la calzada. Si d_C coincide con el valor de disparidad predominante en el obstáculo, se puede concluir que éste se encuentra sobre la calzada. Esta posibilidad de clasificar los obstáculos entre elevados y no elevados puede ser muy interesante para la determinación de regiones de interés en aplicaciones de reconocimiento de señales de tráfico.



(a) Gráfica del comportamiento de la resolución de la profundidad (W).



(b) Gráfica del comportamiento de la resolución de la coordenada U.

Fig. 5. Resolución del módulo de localización de obstáculos en frente del vehículo dentro del área de la región de estudio.

C. Ecuaciones de Localización

Una vez que los obstáculos que se encuentran sobre la calzada han sido detectados, se puede determinar su localización en coordenadas del mundo (U, W) respecto de un sistema de coordenadas situado en el vehículo, siendo ahora W = W'cos(θ), función de las coordenadas del punto de contacto (u_{PC},v_{PC}) en la imagen. Para ello se combinan las ecuaciones estéreo (4) y la ecuación del perfil de la calzada (3), obteniendo las ecuaciones (5) donde θ es el ángulo de cabeceo entre el sistema estéreo y la calzada.

$$W = f\frac{B}{d}\cos(\theta) \quad U = \frac{u \cdot W}{f} \tag{4}$$

$$W = \frac{f \cdot B \cdot m}{v-b}\cos(\theta) \quad U = \frac{u \cdot B \cdot m \cdot \cos(\theta)}{v-b} \tag{5}$$

Al disponer de esta nueva formulación para la determinación de la localización de los obstáculos, se aumenta a una resolución aproximadamente de 250 valores para el mismo rango de distancias. El problema que existe es que la resolución no es lineal con la profundidad (W), como se puede observar en la Fig. 5(a).

$$\frac{\partial W}{\partial v} = \frac{f \cdot B \cdot m \cdot \cos(\theta)}{(v-b)^2} \tag{6}$$

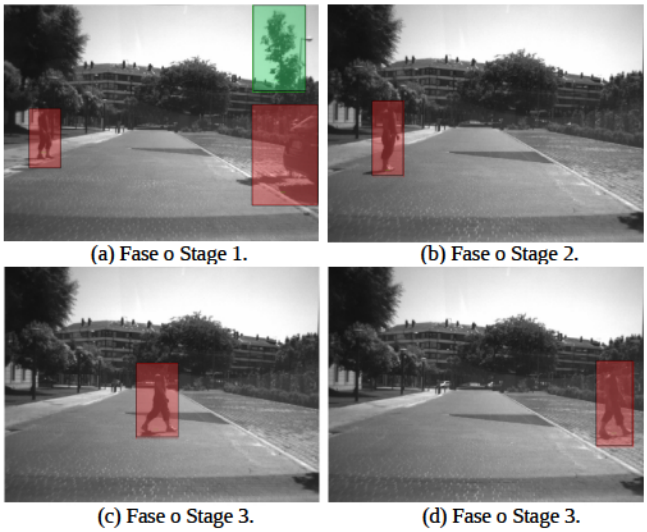
$$\frac{\partial U}{\partial v} = \frac{u \cdot B \cdot m \cdot \cos(\theta)}{(v-b)^2} \quad \frac{\partial U}{\partial u} = \frac{B \cdot m \cdot \cos(\theta)}{v-b} \tag{7}$$

La resolución a lo largo de cada una de las coordenadas (W,U) de las figuras se calcula mediante la derivada parcial, obteniéndose las ecuaciones (6) y (7). Un ejemplo de la resolución de la que se dispone lo encontramos en que en los primeros metros delante del vehículo se tiene una resolución de centímetros y, a una distancia de 30m la resolución es, aproximadamente, de 0.8 metros. Además, se ha de destacar que aparece un empobrecimiento de la resolución en la coordenada U según nos alejamos ya sea en términos de profundidad como lateralmente (ver Fig. 5(b)).

V. Resultados Experimentales

Para poder comprobar la bondad del algoritmo se han realizado varios tests en situaciones comunes de conducción en entornos urbanos, utilizando nuestra plataforma de investigación. A continuación se presenta uno de estos tests, en el que aparecen tres tipos diferentes de obstáculos dentro de la región de estudio (Fig. 6), a saber: un vehículo estacionado en uno de los laterales de la vía y junto a él, otro tipo de obstáculos, un árbol, que es detectado como un obstáculo elevado (rectángulo verde en Fig.6 (a)). El tercer y último obstáculo es un peatón, el único que se encuentra en movimiento, cruzando por delante del vehículo. Este test puede ser dividido en 3 fases o *stages*:

1) El sistema detecta al peatón aproximadamente a 19 metros acercándose a la calzada. El vehículo se aproxima al peatón hasta una distancia de unos 9 metros, sobrepasando al vehículo estacionado y al árbol situado junto a este último (Fig. 6(a)). Esta fase aparece en la gráfica de resultados como una línea con pendiente negativa debido al movimiento del vehículo (Fig. 7).
2) El peatón se detiene y espera hasta que el vehículo se detenga completamente (Fig. 6(b)).
3) El peatón cruza por delante del vehículo (Fig. 6(c) y Fig. 6(d)). En esta ocasión, la línea descrita por las localizaciones de los peatones tiene una pendiente horizontal debido a que el vehículo se encuentra parado.



(a) Fase o Stage 1. (b) Fase o Stage 2.

(c) Fase o Stage 3. (d) Fase o Stage 3.

Fig. 6. Resultado del módulo de detección de obstáculos para un test realizado en un entorno urbano.

Para cuantificar la eficacia de la detección de los obstáculos y distinguiendo entre los tipos de obstáculos, el peatón ha sido detectado como obstáculo en la secuencia con un porcentaje del 100% y englobado correctamente por la región de interés en un 77% de las ocasiones. En el caso del vehículo estacionado y del árbol situado junto a éste, se han detectado en un 96% de los casos situando la región de interés de manera correcta en todo los casos. El porcentaje de falsos positivos que se han producido han sido inferiores a 1%. Los errores de las regiones de interés prodrían mejorarse significativamente con una integración temporal.

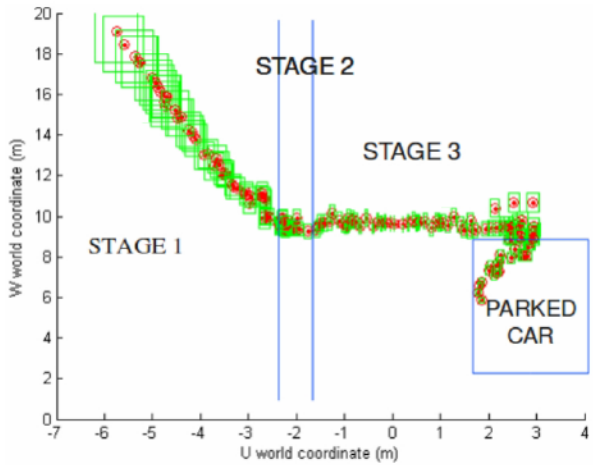


Fig. 7. Resultados de localización para un test en entornos urbanos.

También se han realizado tests donde aparecen una gran cantidad de obstáculos con diferentes características. Un ejemplo de ello es la Fig. 8, donde aparecen varios peatones cruzando delante del vehículo, un obstáculo de grandes dimensiones como es la pared de ladrillos situada a la derecha de la imagen y por último, un árbol clasificado como un obstáculo elevado.

VI. Conclusiones

En conclusión, se ha implementado un sistema de detección de obstáculos para exteriores en tiempo real con una frecuencia de muestreo de unas 15 imágenes por segundo con imágenes VGA. Este sistema puede ser usado en navegación de robots o en el desarrollo de sistemas ADAS, ya que dispone de una gran portabilidad al no necesitar de ningún tipo de calibración extrínseca previa a su funcionamiento.

Otro punto a destacar es que, al usar el perfil de la calzada para la localización de los obstáculos, se hace con una resolución mayor que únicamente utilizando los valores de disparidad, pudiendo además clasificar entre obstáculos elevados y no elevados.

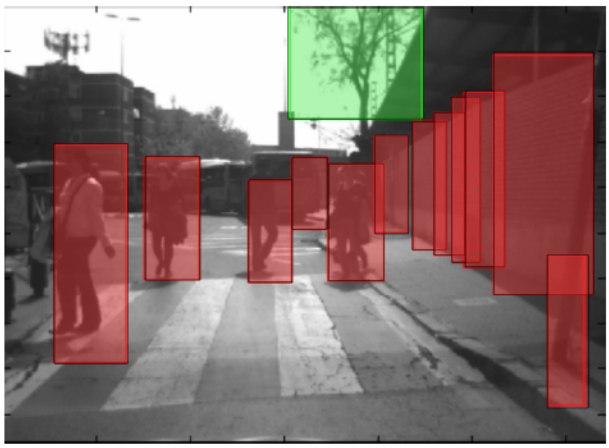


Fig. 8. Imagen resultante de un test donde aparecen varios obstaculos.

References

[1] N. Soquet, M. Perrollaz, R. Labayrade and D. Auber, "Free space estimation for autonomous navigation", Proceedings of the 5th International Conference on Computer vision Systems, 2007.

[2] A. Broggi, C. Caraffi, R. I. Fedriga and P. Grisleri, *"Obstacle detection with stereo vision for off-road vehicle navigation", Proceedings of the IEEE Conference on Computer vision and Pattern Recognition, 2005.*

[3] R. Labayrade, D. Aubert and J. P. Tarel, "Real time obstacles detection in stereovision on non flat road geometry through V-disparity Representation", Intelligent Vehicle symposium, 2002.

[4] *NVIDIA CUDA, Programming guide*, 2.3.1 version, NVIDIA Co.

[5] J. S. Kim, M. Hwangbo and T. Kanade, "Parallel algorithms to a parallel hardware: Designing vision algorithms for a GPU", Workshop on Embedded Computer vision, 2009.

[6] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms", International Journal of Computer vision, vol 47, no 1, pp. 7-42, 2002.

[7] M. Z. Brown, D. Burschka and G. D. Hager, "Advances in Computational Stereo", IEEE Transactions on Pattern analysis and machine intelligence, vol 25, no 8, 2003.

[8] J. Stam, Stereo Imaging with CUDA. Draft, 2008.

[9] O. Faugeras, T. Vieville , E. Theron, J. Vuillemin, B. Hotz, Z. Zhang and L. Moll, "Real time correlation based stereo algorithm, implementation and applications", 1993.

[10] C. H. Lee, Y. C. Lim, S. Kong and J. H. Lee, "Obstacle localization with a binarizared v-disparity map using local maxima frequency values in stereo vision", International Conference on Signals, Circuits and System, 2008.

[11] B. Musleh, A. de la Escalera and J.M. Armingol, "U-V Disparity Analysis in Urban Environments", 13th International Conference on Computer Aided Systems Theory, 2011.