https://doi.org/10.17979/spudc.9788497498081.1003

Sistema de visión estereoscópica para la detección de obstáculos en tiempo real con sistemas empotrados de bajo coste

Miguel Gilabert Gisbert
migigis@epsa.upv.es

Jaime Masiá Vañó
jmasia@mcm.upv.es

Pau Micó Tormos
pabmitor@upv.es

Resumen

En este documento se pretende mostrar el análisis para el desarrollo de una aplicación de visión estereoscópica destinada a proporcionar autonomía en sistemas robotizados, utilizando sistemas empotrados de bajo coste. Se procederá a analizar información en tiempo real, obtenida mediante dos cámaras de forma simultánea, reconstruyendo la escena 3D y representando la profundidad de los objetos detectados.

Palabras Clave: visión artificial, visión estereoscópica, robótica, sistemas empotrados, escena tridimensional, sistema de visión.

1 INTRODUCCIÓN

La visión artificial, concretamente la visión estereoscópica, tiene un gran interés para crear sistemas robotizados que puedan interactuar con el entorno y realizar algunas tareas de forma automática. El tamaño del dispositivo utilizado es un factor muy importante, ya que puede ser necesario el acceso a lugares complicados. De ahí el interés en utilizar sistemas empotrados.

2 ANTECEDENTES

2.1 Visión artificial

La potencia demostrada por los sistemas de visión inspiró a los investigadores a desarrollar sistemas basados en la visión humana, también denominada visión binocular [1], que será el tema central del presente trabajo.

La visión artificial tiene un gran interés para crear servicios que puedan satisfacer las necesidades del ser humano, como en los casos de las personas discapacitadas. También es importante para crear sistemas robotizados que puedan interactuar con el entorno y realizar algunas tareas de forma automática, en este caso cabe destacar proyectos de exploración como "Opportunity" [2], desarrollado por la NASA.

2.2 Visión estereoscópica

En el campo de la visión por computador podemos encontrar una técnica denominada visión estereoscópica, que permite obtener una escena tridimensional utilizando dos imágenes bidimensionales. El proceso para representar los elementos de la escena tridimensional se denomina "Matching" o correspondencia. Con dicho proceso se obtiene un mapa de disparidad, que sirve para obtener información sobre la distancia a la que se encuentran los objetos proyectados.

El objetivo principal del trabajo realizado se centra en la obtención de información en tiempo real de una escena tridimensional, obtenida mediante dos imágenes bidimensionales.

2.2.1 Disparidad

Una de las imágenes bidimensionales se utiliza como imagen de referencia, cada píxel de dicha imagen se encuentra desplazado horizontalmente en la otra imagen. Dicho desplazamiento se conoce como disparidad. Los objetos lejanos presentan niveles de disparidad pequeños, al contrario de lo que ocurre con los objetos cercanos.

Si se rectifican las imágenes verticalmente, la disparidad afectará únicamente a una dimensión, lo que se refleja en la ecuación 1.

$$(X, Y) \rightarrow (X', Y) = \text{Epipolaridad horizontal} \quad (1)$$

La disparidad se mide en píxeles y si los valores no son enteros se dice que la disparidad tiene una precisión sub pixel. Para reconstruir la escena tridimensional se calcula el mapa de disparidad [3] y se pueden obtener diferentes tipos de mapa:

- Mapa denso (dense DM): El mapa se calcula para todos los píxeles.
- Mapa disperso (sparse DM): Se realiza el cálculo para algunos píxeles.

Si se utiliza un sistema de adquisición especial (láser, luz estructurada, etc.) se puede obtener el mapa de disparidad real, denominado "Ground Truth" [4]. Mediante el mapa de disparidad se puede obtener la profundidad de los objetos detectados proporcionando una escena 3D, que está formada por una nube de puntos.

El mapa de disparidad se consigue mediante los denominados métodos de correspondencia, que se pueden clasificar de la siguiente forma [5]:

- Métodos basados en píxeles.
- Métodos basados en área.
- Métodos basados en características.

2.2.2 Funciones de coste

Las funciones de coste se utilizan para reducir el error obtenido durante el proceso de correspondencia. Las versiones más complejas admiten cambios radiométricos en la intensidad de los píxeles y la presencia de ruido. Las funciones de coste más extendidas son las siguientes [6]:

- Suma de diferencias absolutas (SAD), que presenta una velocidad de ejecución y una precisión que se pueden asumir.
- Suma de diferencias al cuadrado (SSD), se mejora la precisión sacrificando la velocidad obtenida.
- Correlación cruzada normalizada (NCC), es más compleja que las anteriores, pero aporta buenos resultados cuando los niveles de gris son altos o cuando las variaciones de intensidad se producen continuamente.

Como se ha indicado, en las búsquedas sencillas el proceso se realiza para cada par de píxeles. Este proceso se llama "The Winner Take All" (WTA) y aunque existen otros métodos que ofrecen una mayor precisión, el método indicado ofrece un buen rendimiento [7].

2.2.3 Cálculo de la profundidad

Para realizar el cálculo de la profundidad se deben realizar algunos cálculos sobre las imágenes. Los símbolos que se van a utilizar en las ecuaciones se muestran en la tabla 1.

Tabla 1: Símbolos para el cálculo de la profundidad.

Símbolo	**Significado**
b	Distancia entre los centros ópticos de las cámaras
f	Distancia focal de las cámaras
Z	Profundidad
X	Posición real del punto proyectado
X_I	Posición de la proyección del punto en la cámara izquierda
X_D	Posición de la proyección del punto en la cámara derecha

Las ecuaciones 2 y 3 corresponden al cálculo realizado sobre la imagen izquierda.

$$\frac{\left(\frac{b}{2}+\mathrm{X}\right)}{Z}=\frac{X_I}{f} \tag{2}$$

$$\mathrm{X_I}=\frac{f}{Z}\left(X+\frac{b}{2}\right) \tag{3}$$

Del mismo modo se realizan las operaciones sobre la imagen derecha. Dichas operaciones se representan en las ecuaciones 4 y 5.

$$\frac{\left(\frac{b}{2}-\mathrm{X}\right)}{Z}=\frac{X_D}{f} \tag{4}$$

$$\mathrm{X}_D=\frac{f}{Z}\left(X-\frac{b}{2}\right) \tag{5}$$

Teniendo en cuenta las ecuaciones anteriores, podemos calcular la disparidad utilizando la ecuación 6.

$$d=X_I-X_D=\frac{fb}{Z} \tag{6}$$

Para poder utilizar unidades métricas, la disparidad "d" (en píxeles) se multiplicará por la distancia inter píxel (dx), que se obtiene al dividir el tamaño del sensor (normalmente en mm) entre su resolución en píxeles. Ambas variables se obtienen sobre el eje X.

De la fórmula anterior se puede observar que la profundidad "Z" es inversamente proporcional a la disparidad "d", lo que se puede expresar mediante la ecuación 7.

$$Z=\frac{fb}{d} \tag{7}$$

2.2.4 Trabajos relacionados

Actualmente existen numerosas aplicaciones donde se utilizan sistemas de visión estereoscópica, como el procesamiento de imágenes aéreas y por satélite, sistemas de visión para personas invidentes, aplicaciones para asistencia al conductor en automoción y por supuesto, proyectos destinados a proporcionar autonomía en sistemas robotizados [8].

3 MATERIALES

3.1 Sistemas empotrados

La portabilidad es un factor muy importante, por lo que el tamaño físico del sistema implementado no debe ser excesivo. Una estructura de gran tamaño limitaría la utilidad del sistema y las posibilidades de integración. Actualmente existe un gran interés en proyectos relacionados con Internet de las cosas y Smart City [9]. Los sistemas empotrados representan una posibilidad muy interesante.

Como se ha indicado anteriormente, el desarrollo del proyecto se basa en la utilización de sistemas empotrados de bajo coste para conseguir un proyecto asequible. A continuación, se exponen los sistemas analizados para mostrar algunas posibilidades:

3.1.1 ODROID XU-4

Este dispositivo utiliza la arquitectura ARM® big.LITTLE™ y ofrece una solución de multiprocesamiento heterogéneo (HMP), ya que está equipado con 4 núcleos ARM® Cortex®A15™ de 2 GHz y 4 núcleos ARM® Cortex®A7™ de 1.4 GHz, también dispone de 2 GB de memoria RAM. Las dimensiones son de 82 x 58 x 22 mm, con un peso de 60 gramos y el precio de dicho dispositivo es de 83.96 euros.

3.1.2 Raspberry

Se trata de un ordenador de placa reducida (SingleBoard Chip) de bajo consumo y flexible. El modelo Raspberry Pi 2 dispone de un procesador de cuatro núcleos ARM Cortex-A7 de 900 MHz y 1 GB de memoria RAM.

Este dispositivo tiene unas dimensiones de 85 x 56 x 17 mm con un peso de 45 gramos y su precio es de 41.95 euros.

3.1.3 myRIO

Se trata de un sistema empotrado destinado al ámbito educativo, relacionado con la ingeniería. Posee un procesador ARM® Cortex™A9 Dual Core en tiempo real y E/S personalizadas de FPGA Xilinx.

Este dispositivo dispone de un sistema operativo Linux embebido, las dimensiones son de 136.6 x 86.6 x 25 mm con un peso de 193 gramos. Este dispositivo tiene un precio de 470 euros.

3.2 Cámaras

El trabajo realizado se ha orientado al uso de componentes de bajo coste. Lo realmente interesante es la compatibilidad que puedan ofrecer con los sistemas operativos existentes y las prestaciones que ofrezcan. Los factores más importantes son los FPS (frames por segundo), las resoluciones disponibles y el formato de imagen. El tipo de conexión es un factor de gran importancia, por la velocidad de transmisión, pero normalmente suelen ofrecer conexión USB. En la tabla 2 se muestran características sobre algunos de los dispositivos utilizados.

Tabla 2: Comparativa de las cámaras utilizadas.

Característica	**Logitech C270**	**Logitech C905**
Conexión	USB 2.0	USB 2.0
FPS	30	30
Resolución	320x240 640x480 1280x960	320x240 640x480 960x720 1600x1200
Enfoque	Fijo	Automático
Precio	29.95 euros	129.94 euros

3.3 Tipo de memoria

Los sistemas empotrados, por su tamaño y características, suelen utilizar memoria externa para la instalación del sistema operativo, las herramientas software y como sistema de almacenamiento. Debido a esto el tipo de memoria utilizado se convierte en un factor importante. A continuación, se va a describir el soporte utilizado:

3.3.1 Tarjetas eMMC

Este tipo de memoria presentan un mayor rendimiento que las memorias de tipo SD, ofreciendo mayores prestaciones en cuanto a velocidad de transmisión. En las tablas 3 y 4 se muestran algunas comparativas.

Tabla 3: Comparativa con operaciones de lectura.

SDClass 10	**SDUHS1**	**eMMC 5.0**
8.5 MB/s	10.8 MB/s	39.3 MB/s

Tabla 4: Comparativa con operaciones de escritura.

SDClass 10	SDUHS1	eMMC 5.0
18.9 MB/s	35.9 MB/s	140 MB/s

4 METODOLOGÍA

El lenguaje de programación utilizado para la implementación ha sido C/C++, utilizando la librería OpenCV. Se trata de software libre y ofrece compatibilidad con la mayoría de sistemas empotrados indicados en este documento. En esta sección se describe el proceso realizado para el desarrollo de la implementación utilizando esta tecnología [10]. A continuación, se describe el proceso realizado:

4.1 Calibración

El proceso de calibración se realiza mediante la obtención de pares de imágenes, procedentes de la cámara izquierda y derecha. Se utiliza un patrón, que deberá ser detectado en las dos imágenes de forma simultánea para obtener las muestras. Dicho patrón se muestra a continuación, en la figura 1.

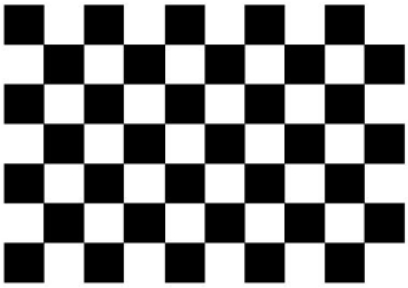

Figura 1: Patrón para calibrar el sistema de visión.

Es conveniente cambiar la inclinación del patrón utilizado para obtener diferentes muestras. De esa forma el proceso de calibración obtendrá más información para mejorar los resultados.

Al finalizar la obtención de las muestras, se procesan y se obtiene una vista de las imágenes rectificadas, además de unos archivos de configuración muy importantes para realizar el proceso siguiente, denominado proceso de matching o correspondencia.

En la figura 2 se puede observar un ejemplo de los resultados obtenidos con el proceso de calibración.

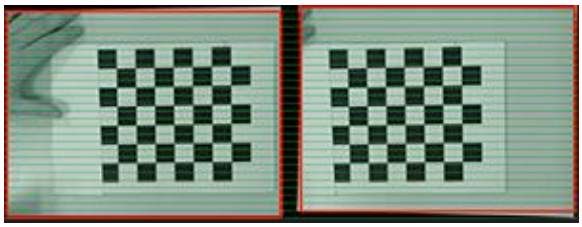

Figura 2: Imágenes rectificadas al finalizar el proceso de calibración.

En la figura 2, una imagen aparece desplazada verticalmente respecto a la otra, mediante el proceso de calibración se obtienen las imágenes rectificadas y los parámetros para los archivos de configuración, que son los siguientes:

En primer lugar, un archivo denominado "extrinsics.yml", del que se obtienen los siguientes parámetros:

- Rotación (R): Matriz de rotación de 3x3 desde el origen, que normalmente es el primer ángulo interno del tablero de ajedrez utilizado durante la calibración.
- Translación (T): La translación del primer ángulo interno del tablero de ajedrez.

En segundo lugar, un archivo con el nombre "intrinsics.yml" que proporciona la siguiente matriz, representada en la ecuación 8.

$$\begin{matrix} f_x & k & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{matrix} \quad (8)$$

Los elementos que aparecen en la matriz anterior son los siguientes:

- La distancia focal de las cámaras en píxeles, que aparece en la diagonal de la matriz obtenida (fx, fy).
- El centro de proyección (cx, cy), que aparece en la última columna de la matriz.
- Los coeficientes de distorsión (k).

Durante el proceso de calibración se pueden producir los siguientes errores:

- cRMS Error: Error producido en la rectificación de los pares de imágenes.
- Average Reprojection Error: Detectar los píxeles en las imágenes no es posible siempre, por lo que es conveniente controlar el error medio al volver a proyectar los píxeles. Debe ser lo más cercano a cero.

4.2 Matching

El proceso de matching o correspondencia se basa en localizar los puntos, que representan a los objetos con

una determinada disparidad. Al finalizar este proceso se obtendrá el mapa de disparidad, que aportará información relativa a la profundidad de los objetos detectados.

OpenCV ofrece implementaciones donde se requiere la lectura de imágenes de disco para llevar a cabo el proceso de matching, ya que se contempla como un proceso modular. En el desarrollo realizado se propone realizar la captura de las imágenes y utilizar directamente los objetos de imagen, actualmente en memoria, para calcular el mapa de disparidad. Con lo que se evita el acceso a disco, reduciendo el tiempo de ejecución. Los algoritmos utilizados para realizar el proceso de matching son "Block Matching" y "Semi Global Block Matching" [11].

Al finalizar el proceso de matching se obtiene un archivo con formato "asc", donde aparece la nube de puntos generada, almacenando las coordenadas XYZ para cada punto.

5 RESULTADOS

El sistema de visión estereoscópica utilizado está formado por un sistema empotrado ODROID XU-4, una tarjeta de memoria eMMC 5.0 donde se encuentra instalado el sistema operativo y dos cámaras Logitech C270.

Para el desarrollo de las pruebas se ha realizado la instalación de las cámaras sobre un soporte metálico, de esta forma se mantiene la posición de los dispositivos y es posible modificar la distancia a la que se desean colocar. Las imágenes se obtienen con una resolución de 320x240 píxeles.

En la figura 3 se muestran algunos detalles de la instalación realizada:

Figura 3: Instalación del sistema de visión estereoscópica.

Las pruebas se han centrado en la detección de objetos, con el objetivo de obtener información en tiempo real y poder aportar autonomía en sistemas robotizados. Debido a esto, se ha optado por utilizar el método Block Matching, que requiere un reducido tiempo de ejecución y aporta la suficiente información para detectar y localizar los obstáculos a distancia.

Se han utilizado algunos objetos y se han colocado a diferentes distancias para realizar la detección.

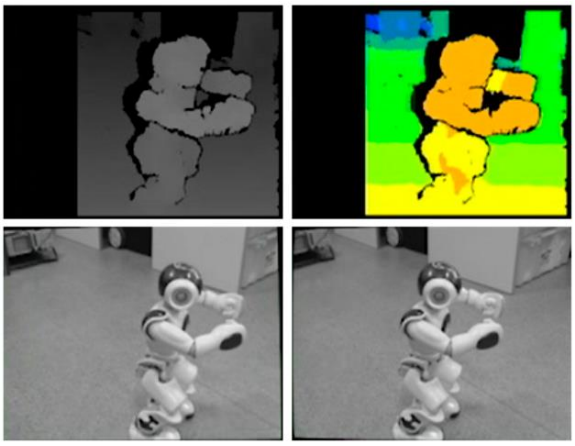

Figura 4: Representación de la profundidad mediante el proceso de matching.

En la figura 4 se muestra el mapa de disparidad en la imagen superior izquierda, el mapa de profundidad en la imagen superior derecha y las imágenes inferiores representan las imágenes bidimensionales, capturadas por las cámaras.

Los tiempos de ejecución necesarios para la obtención de los resultados se muestran en la tabla 5.

Tabla 5: Tiempos de ejecución requeridos.

Proceso	**Tiempo (ms)**
Captura de imágenes	9.00
Matching	50.86
Mapa de disparidad	20.00
Nube de puntos	18.00

Como se puede observar en la tabla 5, el tiempo requerido para localizar los objetos detectados es muy reducido, permitiendo la ejecución de la aplicación en tiempo real. Se requieren 97.86 milisegundos para realizar las operaciones, lo que se traduce en 10.22 frames por segundo.

Es interesante realizar una comparativa entre los métodos de correspondencia que han sido utilizados para las pruebas. En las figuras 5 y 6 se muestran los resultados de algunas pruebas realizadas.

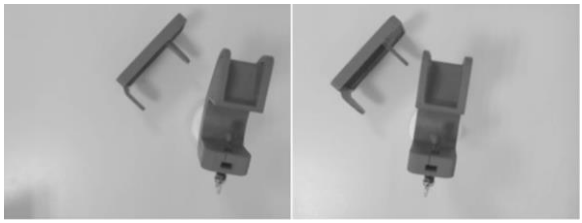

Figura 5: Imágenes bidimensionales obtenidas utilizando piezas realizadas en ABS.

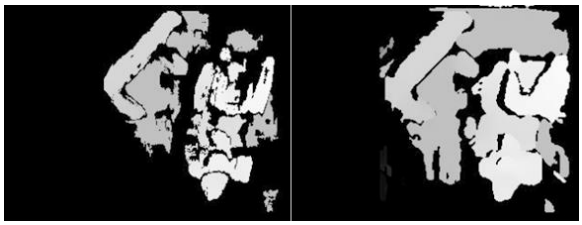

Figura 6: Mapas de disparidad. a) Block Matching b) Semi Global Block Matching.

Como se puede observar en la figura 6, el método Block Matching (figura de la izquierda) se ve afectado más fácilmente por el ruido y la definición de los conjuntos es deficiente. En cambio, el segundo método presenta una mayor inmunidad y los conjuntos se pueden definir más correctamente.

La velocidad de ejecución que presenta el método Block Matching es una ventaja, pero los problemas presentados son un aspecto importante. En la tabla 6 se muestran tiempos de ejecución requeridos por los métodos utilizados.

Tabla 6: Tiempos de ejecución requeridos.

Método	**Tiempo (ms)**
Block Matching	50.8576
Semi Global Block Matching	300.6140

Los métodos que utilizan correspondencia local también presentan debilidad ante la existencia de brillo en los objetos. Por lo tanto, es interesante reflejar esta situación, ya que puede perjudicar a la fiabilidad del resultado obtenido. En la figura 7 se presentan algunos resultados utilizando elementos que producen la situación indicada.

Figura 7: Imágenes bidimensionales obtenidas utilizando un engranaje metálico.

En la figura 7 se observa la disparidad producida al proyectar el objeto en las imágenes bidimensionales, ya que dicho objeto se encuentra a 45 cm del sistema de visión. Al utilizar una base de color homogéneo no se produce suficiente disparidad, por lo que los puntos representados en la nube de puntos corresponden al objeto utilizado.

En la figura 8 se puede observar la nube de puntos generada, en la imagen de la izquierda se representa la vista en alzado y en la imagen derecha la vista cenital del engranaje metálico. Se muestran deformaciones en la superficie del objeto representado, que se producen por el brillo del metal.

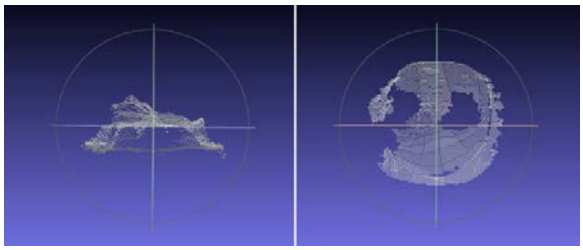

Figura 8: Representación de la nube de puntos generada.

Se ha representado la nube de puntos utilizando el software libre “Meshlab”, que se utiliza para convertir las mallas a formato 3D. Como se observa en la figura 8, sobretodo en la vista en alzado, la parte izquierda aparece más elevada, debido al brillo producido. En la parte derecha también ocurre, pero en menor medida.

6 CONCLUSIONES

Ante los resultados obtenidos, podemos decir que el sistema de visión 3D de bajo coste cumple con la premisa del tiempo real. El mapa de profundidades de una escena se obtiene en menos de 98 ms.

Esta velocidad de cálculo de los mapas deja un margen de tiempo suficiente para poder incorporar nuevas funcionalidades a la unidad de procesamiento como, por ejemplo:

- La configuración adaptativa de los parámetros, para el algoritmo de cálculo de la disparidad. Dependiendo de las características de la escena, modificará sus parámetros (tamaño del bloque, distancia entre focos, etc.) optimizando los resultados del mapa de profundidad resultante.
- La programación de algoritmos de reconocimiento, que permitan identificar ciertos patrones (redes neuronales convolucionales, etc.) [12].

Agradecimientos

Los autores del trabajo realizado desean mostrar su agradecimiento a todas las personas que se han mostrado partidarias del desarrollo del mismo y también al campus de Alcoy de la Universitat Politècnica de València por prestar sus instalaciones para la realización de algunas de las pruebas.

Referencias

[1] Myron Z., Darius B., Gregory D. (2003) "Advances in Computational Stereo", *IEE Transactions on pattern analysis and machine intelligence*, Vol.25, No.8

[2] Arvidson R., (2014) "Roving on Mars with Opportunity and Curiosity: Terramechanics and Terrain Properties", *Earth and Space*, pp. 165-173.

[3] K. Muhlmann, D. Maier, R. Hesser, R. Manner, (2001) "Calculating dense disparity maps from color stereo images, an efficient implementation", *Stereo and Multi-Baseline Vision,* pp. 30–36.

[4] D. Scharstein and R. Szeliski., (2003) "Highaccuracy stereo depth maps using structured light", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* volume 1, pp. 195-202.

[5] Bodkin B. H., (2012) "Real-time mobile stereo vision", Master's Thesis, University of Tennessee.

[6] Dall'Asta, E., Roncella, R., (2014) "A comparison of semiglobal and local dense matching algorithms for surface reconstruction", *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.,* pp. 187–194.

[7] D. Scharstein and R. Szeliski., (2002) "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms", *International Journal of Computer Vision*, Microsoft Research Technical Report MSRTR200181.

[8] Hirschmüller H., (2011) "SemiGlobal Matching – Motivation, Developments and Applications", *Invited Paper at the Photogrammetric Week*, *Stuttgart, Germany,* pp. 173-184.

[9] Wnag P., Ali A., Kelly W., (2015) "Data security and threat modeling for smart city infrastructure", *Cyber Security of Smart Cities*, pp. 1-6.

[10] Sebastian D., Moos H., Sander L., Martijn V., (2010) "Stereo Vision using the OpenCV library", pp. 9-12.

[11] Fengjun H., Yanwei Z., (2013) "Comparative Research of Matching Algorithms for Stereo Vision", *Journal of Computational Information Systems*, pp. 2-8.

[12] Jure Z., Yann L., (2015) "Computing the Stereo Matching Cost With a Convolutional Neural Network", *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1592-1599.