# Aproximación del Mapa de Disparidad Estéreo Mediante Técnicas de Aprendizaje Automático

Martin Peris Martorell
Tesis Máster IARFID UPV

Septiembre 2011

**Resumen**

El objetivo de este trabajo es abordar el problema del cálculo del mapa de disparidad de un par de imágenes estéreo. En lugar de considerar los métodos clásicos para calcular el mapa de disparidad a partir de dos imágenes, se va a tomar una aproximación basada en técnicas de aprendizaje automático. El cálculo de la disparidad se considerará un mero problema de clasificación y se entrenará un clasificador cuyo objetivo alcanzar un rendimiento igual o superior al de los métodos clásicos. Para solucionar el problema de la falta de imágenes reales con los verdaderos mapas de disparidad conocidos, se va a presentar una técnica original de generación sintética de imágenes con disparidad conocida que puedan usarse para entrenar el clasificador y aplicarse a imágenes reales. Así mismo, se introducirán las características utilizadas para entrenar el clasificador, el tipo de clasificador a utilizar, se compararán los resultados obtenidos con los de los métodos clásicos y se propondrán nuevas líneas de investigación en base a los resultados de este trabajo.

# Índice

# 1. Introducción

El ser humano y otros seres de la naturaleza perciben el mundo en 3 Dimensiones gracias a su sistema de *visión binocular*, también conocido como *visión estéreo*, que les dota de un sentido de percepción de la profundidad debido a un fenómeno visual conocido como *estereopsis* ([7]). Gracias a éste fenómeno, se puede calcular la posición en el mundo físico de un objeto dada la proyección de dicho objeto, que será ligeramente diferente en cada retina. Aunque este proceso, completamente natural e instantáneo en los seres vivos, es extremadamente complejo, se puede emular hasta cierto punto mediante computadoras.

Para tal fin, se utilizan *cámaras estéreo*: básicamente dos cámaras unidas cuyo desplazamiento mutuo es conocido, véase la Figura 1.

Figura 1: Ejemplo de cámara estéreo. Modelo Bumblebee2 de la empresa Point Grey (http://www.ptgrey.com)

En una computadora, la tarea de la visión binocular se lleva a cabo encontrando correspondencias entre puntos vistos por una de las cámaras y los mismos puntos vistos por la otra cámara. Con tales correspondencias y conociendo el desplazamiento mutuo entre las cámaras se puede calcular la posición 3D de los puntos. Sin otro conocimiento a priori, la correspondencia de un punto en una imagen puede aparecer en cualquier otro lugar de la otra imagen, haciendo el proceso de encontrar correspondencias extremadamente caro computacionalmente. Pero utilizando el conocimiento sobre la geometría del sistema de cámaras se puede acotar la búsqueda, reduciendo el coste computacional.

En la práctica, la visión estéreo utilizando dos cámaras suele englobar los siguientes pasos ([2]):

1. Eliminar matemáticamente la distorsión radial y tangencial producida por la lente de la cámara. El fenómeno de distorsión radial y tangencial que introduce la lente de la cámara provoca que las líneas rectas en el mundo físico aparezcan curvadas en las imágenes tomadas por las cámaras. El resultado del proceso de eliminación de la distorsión son imágenes reajustadas.

2. Ajustar los ángulos y las distancias entre las cámaras. A éste proceso se le conoce como rectificación. El resultado son imágenes rectificadas y cuyas filas están alineadas. En breve se explicará el interés de realizar este proceso.

3. Encontrar los mismos puntos de interés en las imágenes de la cámara izquierda y la cámara derecha. Este proceso se conoce como correspondencia. El resultado de este proceso son las disparidades de los puntos de interés, donde disparidad significa la diferencia de la coordenada $x$ de un punto de interés en las imágenes de la cámara izquierda y derecha: $x^l - x^r$.

4. Conociendo la disposición geométrica de las cámaras, se puede conocer la distancia a la que se encuentran los puntos de interés utilizando triangulación. Este proceso se conoce como reproyección y el resultado es la posición 3D de los puntos de interés.

Como se ha comentado en el anterior paso número 2, para calcular la posición física de los puntos de interés se utilizan las imágenes rectificadas. La rectificación consigue que las filas de las imágenes izquierda y derecha estén alineadas, de esta manera dado un punto de interés en la imagen izquierda, su punto correspondiente en la imagen derecha se encontrará en la misma fila en ambas imágenes.

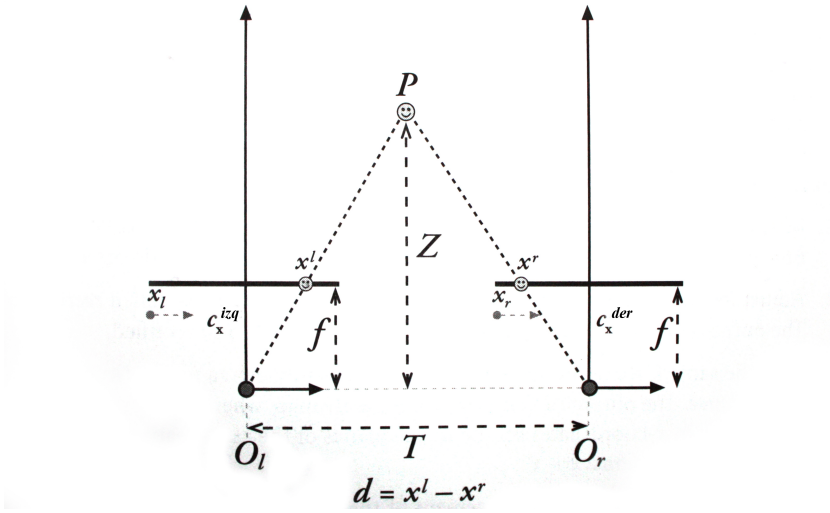


Figura 2: Con un par estéreo perfectamente reajustado, rectificado y con una correspondencia conocida, la distancia $Z$ se puede encontrar mediante triángulos semejantes. Tomado de [2]

Si se asume que se dispone de imágenes perfectamente reajustadas, rectificadas, el desplazamiento entre cámaras es conocido, como en la Figura 2: dos cámaras cuyos planos de imagen son perfectamente coplanares, sus ejes ópticos paralelos y a una distancia conocida, con longitudes focales iguales $f_r = f_l$. Si también se asume que los puntos principales $c_x^{izq}$ y $c_x^{der}$ (los puntos en donde los rayos principales intersectan con los planos de imagen) han sido calibrados para tener las mismas coordenadas en píxeles tanto en la imagen izquierda como en la derecha y que se puede encontrar un punto $P$ en el mundo físico con proyecciones $p_l$ y $p_r$ en las imágenes, cuyas coordenadas en el eje horizontal son respectivamente $x^l$ y $x^r$. Entonces, en este escenario simplificado, se puede derivar la distancia $Z$ de la siguiente manera ([2]):

$$\frac{T - (x^l - x_r)}{Z - f} = \frac{T}{Z} \rightarrow Z = \frac{fT}{x^l - x^r}$$

El valor $d = x^l - x^r$ es conocido como disparidad.

En el mismo escenario, se define la matriz de reproyección $Q$:

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

Donde $c_x$ y $c_y$ es el punto principal en la imagen izquierda, $f$ es la longitud focal de las cámaras (se supone la misma en ambas) y $T_x$ es la distancia horizontal entre los centros de las cámaras. En la fórmula anterior todos los parámetros son referentes a la imagen izquierda excepto $c'_x$ que es la coordenada x del punto principal en la imagen derecha.

Dadas las coordenadas homogéneas en la imagen izquierda de un punto de interés y su disparidad $d$ con su punto correspondiente en la imagen derecha, la matriz Q permite calcular las coordenadas 3D en el mundo físico del punto de interés de la siguiente manera:

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

Y finalmente las coordenadas en 3D del punto de interés son $(X/W, Y/W, Z/W)$.

Si en lugar de calcular la disparidad $d$ de puntos de interés aislados, se calcula la disparidad para todos los puntos de una imagen, se obtiene lo que se conoce como *mapa de disparidad denso* o simplemente *mapa de disparidad*. Ejemplo en la figura 3.

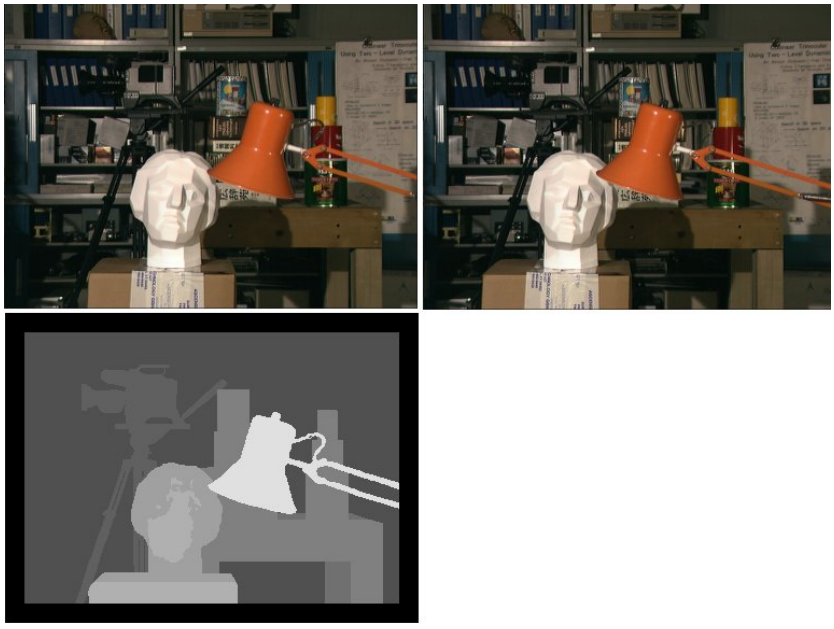

Figura 3: Par estéreo de la escena Tsukuba (imágenes en color) con su mapa de disparidad (imagen en escala de grises). El marco negro alrededor del mapa de disparidad corresponde a zonas en las que la disparidad no está definida.

Al representar como una imagen el mapa de disparidad, se puede observar que los puntos más cercanos a la cámara aparecen en un tono de gris claro (el valor de disparidad $d$ es más grande) y los puntos más alejados de la cámara aparecen en un tono de gris mas oscuro (el valor de disparidad $d$ es más pequeño). Dadas las características del escenario simplificado que se ha definido, los puntos que estén situados físicamente en el infinito (o a una distancia lo suficientemente grande) aparecerán en la misma posición en ambas imágenes, tendrán disparidad $d = 0$ y por lo tanto en el mapa de disparidad se visualizarán en un color completamente negro.

A día de hoy, existen múltiples aproximaciones para poder calcular un mapa de disparidad denso. Aunque la complejidad que entraña calcularlo de manera precisa hace que se tenga que sacrificar calidad en pro de velocidad de cálculo y viceversa. Normalmente los algoritmos ligeros de cálculo de correspondencias suelen generar mapas de disparidad bastante pobres. Por el contrario, algoritmos más sofisticados de cálculo de correspondencias generan mejores mapas de disparidad pero su tiempo de ejecución es prohibitivo para aplicaciones de tiempo real.

Las aplicaciones que tendría el poder calcular un mapa de disparidad denso de calidad en tiempo real son prácticamente ilimitadas. Siendo uno de los mayores desafíos, su aplicación a la robótica móvil. La visión estéreo es, hoy en día, una técnica básica para la percepción visual de los robots y grandes esfuerzos en avanzar en este tema de investigación se están llevando a cabo. La mayor parte en métodos basados en pares de imágenes estéreo calibradas [13].

Para muchas aplicaciones robóticas modernas, como por ejemplo Odometría Visual [6] y Reconocimiento de Objetos 3D [9], se requieren mapas de disparidad denso. Desafortunadamente, como ya se ha comentado, la mayor parte de los métodos de cálculo de correspondencias existentes tienen severas limitaciones a la hora de utilizarse en aplicaciones de tiempo real. Es por eso que, para muchas aplicaciones se calcula la disparidad para un reducido número de puntos de interés en lugar de calcular un mapa de disparidad denso [4][14].

Como se ha visto, teniendo el mapa de disparidad denso de una escena se pueden calcular las coordenadas 3D del mundo físico de los objetos que aparecen en dicha escena. Existen otras maneras de conocer estas coordenadas, como son el uso de Láser o métodos de luz estructurada. Pero normalmente este tipo de dispositivos suele tener un coste que va desde cientos hasta incluso miles de Euros haciéndolos muy poco asequibles, sobre todo para proyectos de robótica domestica en los que el precio de los robots debería ser lo más bajo posible.

Recientemente están apareciendo en el mercado dispositivos como Kinect, Figura 4, que permiten obtener una imagen en 3D (conocido como nube de puntos o point cloud) de la escena que hay en su campo de visión utilizando para ello un proyector de luz infrarroja estructurada y una cámara infrarroja. Obteniendo unos resultados, dentro de los parámetros de funcionamiento establecidos para el dispositivo, de muy buena calidad y además a un precio que ronda los 150 Euros [18].


Figura 4: Microsoft Xbox Kinect.

Este dispositivo, aunque está concebido en un principio para la industria de los videojuegos, está causando una auténtica revolución en el mundo de la sensorización robótica y la visión por computador. Sentando rápidamente las bases para la evolución de los bloques sobre los que construir aplicaciones basadas en nubes de puntos 3D [17], como pueden ser descriptores basados en histogramas de puntos característicos (figura 5), filtrado de nubes de puntos (figura 6) o segmentación de objetos (figura 7) por citar algunos.

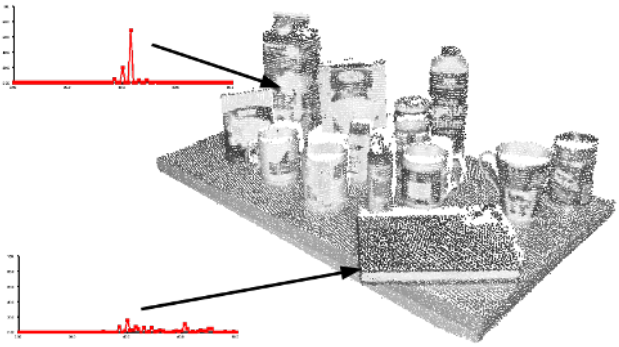

Figura 5: Ejemplo de Descriptor basado en Histogramas de Puntos Característicos

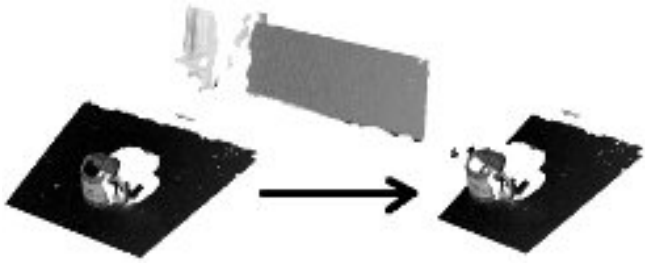

Figura 6: Ejemplo de filtrado de imágen basado en 3D

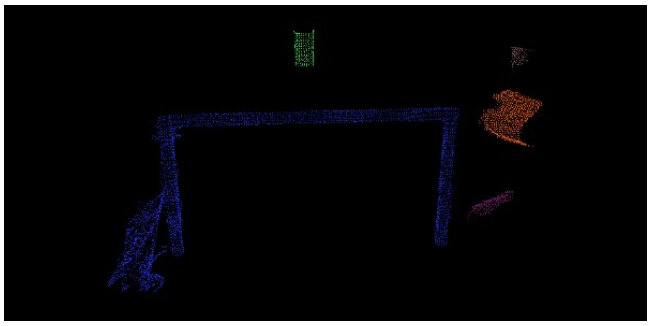

Figura 7: Ejemplo de segmentación de objetos 3D

Estos bloques básicos jugarán un papel importantísimo en el desarrollo de futuras aplicaciones basadas en 3D. Pero los dispositivos como Kinect sufren un grave problema: no funcionan en exteriores, solo funcionan en interiores. Lo cual es un gran impedimento. El motivo es que en entornos de exterior el sol emite luz infrarroja que hace que la cámara de infrarrojos del dispositivo no sea capaz de identificar los patrones de la luz estructurada que él mismo proyecta, haciendo prácticamente imposible estimar las distancias 3D.

Aquí es donde los mapas de disparidad pueden marcar la diferencia, ya que las cámaras de luz visible funcionan tanto en exteriores como en interiores. Si se consigue calcular un mapa de disparidad en tiempo real con la calidad de los dispositivos de luz estructurada y de manera barata, entonces se podrán seguir desarrollando los beneficios de disponer de imágenes en 3D o nubes de puntos.

Así pues, en el presente trabajo se va a abordar el problema del cálculo del mapa de disparidad desde una perspectiva poco estudiada hasta el momento. En lugar de considerar los métodos clásicos para calcular el mapa de disparidad a partir de un par estéreo de imágenes, se va a tomar una aproximación basada en técnicas de aprendizaje automático. Se realizará una breve revisión del estado del arte del problema en cuestión. A continuación se realizará una descripción del trabajo desarrollado en el que se introducirá una técnica para original para la generación sintética de imágenes con disparidad conocida. Se evaluarán los resultados y por último se expondrán las conclusiones y se establecerán las posibles líneas de investigación a seguir en base a los resultados de este trabajo.

# 2. Estado del Arte

En esta sección se hace una breve revisión del estado del arte en el cómputo de mapas de disparidad densos.

## 2.1. Correspondencia estéreo como problema de clasificación

Este problema ha sido objeto de múltiples investigaciones y sigue siendo un campo de investigación muy activo en el presente.

Aunque hasta el momento, muy pocos trabajos [12] se han centrado en enfocar el problema de la correspondencia estéreo como un problema de clasificación. Una de las principales razones que desmotiva a los investigadores en este campo es que, para utilizar métodos de aprendizaje automático, serían necesarias grandes cantidades de imágenes con un mapa de disparidad denso conocido para poder ser utilizados como datos de entrenamiento.

Algunos conjuntos de imágenes estéreo con mapa de disparidad conocido se encuentran disponibles [11]. Dichos conjuntos de imágenes se obtienen utilizando técnicas de luz estructurada en las que un láser posicionado en la misma ubicación que las cámaras que toman las imágenes, mide con gran precisión la distancia a todos los puntos de la imagen. En la Figura 8 y 9 se pueden apreciar algunos ejemplos de los citados conjuntos de imágenes.

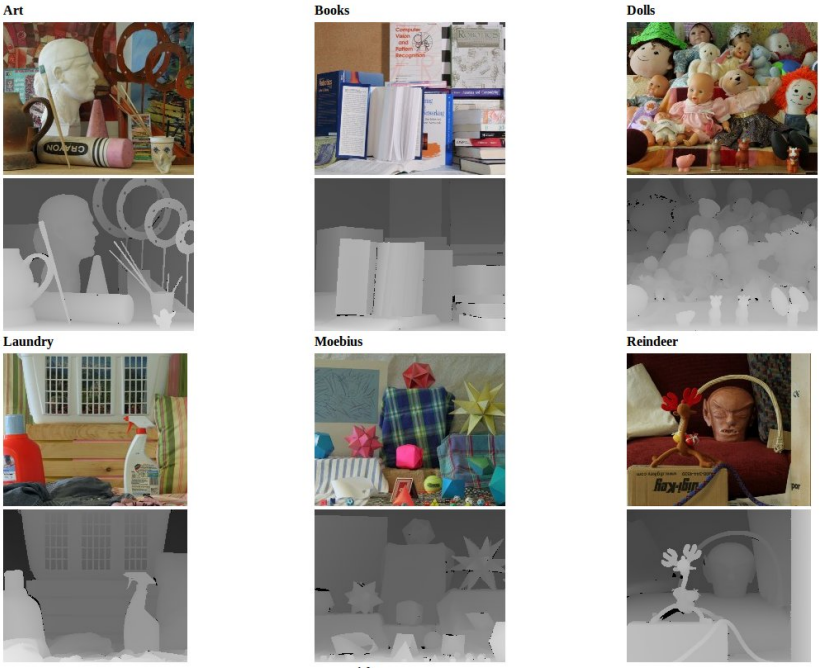


Figura 8: Ejemplos de Middelbure Stereo Dataset 2005

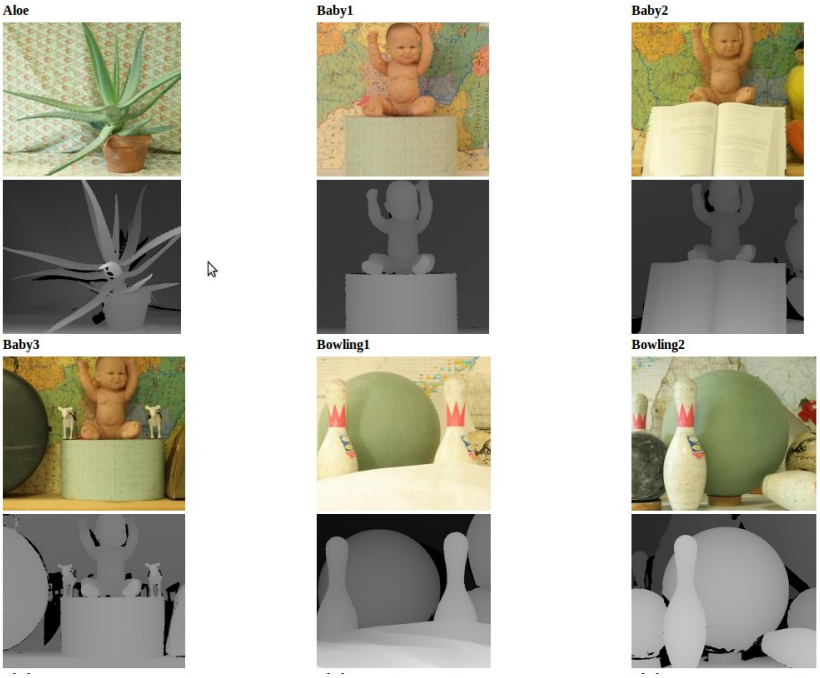


Figura 9: Ejemplos de Middelbure Stereo Dataset 2006

Dada la complejidad de calibrar el láser con la cámara para obtener resultados precisos, existen pocos dataset disponibles. Las escenas en Middlebury Stereo Dataset 2005 y 2006 [12] suman un total de 30. También existen los conjuntos Middlebury Stereo Dataset 2001 y 2003 con un número de escenas de 6 y 2 respectivamente, haciendo un total de 38 escenas entre los datasets 2001, 2003, 2005 y 2006.

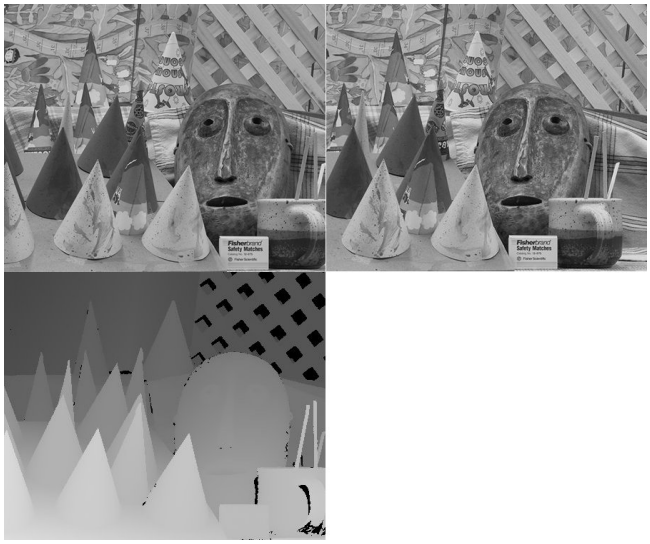

Figura 10: Ejemplos de Middelbure Stereo Dataset 2003, escena *Cones* con mapa de disparidad denso real.

En [12] se utilizan los conjuntos de imágenes en Middlebury Stereo Dataset 2005 y 2006 para entrenar un clasificador basado en SVM que obtiene una tasa de acierto razonablemente alta aunque no en tiempo real. Además, dada la limitación en los datos de entrenamiento, la generalización a nuevos datos (nuevas escenas, cambios de iluminación, ruido, etc.) se espera que no sea del todo satisfactoria.

Por este motivo, la mayor parte de trabajos se han centrado en utilizar técnicas de cálculo de correspondencias basadas en métodos tradicionales. A continuación se citan algunos de los más utilizados.

## 2.2. Métodos clásicos

En esta sección se revisan algunos de los métodos de cálculo de mapas de disparidad densos más utilizados.

### 2.2.1. Sum of Squared Differences

La técnica tradicional básica para generar un mapa de disparidad denso se denomina SSD (Sum of Squared Differences). Se trata de una técnica de correspondencia basada en la correlación entre los píxeles de la imagen izquierda y derecha dentro de un vecindario. Normalmente se escoge una ventana de un cierto tamaño $n \times m$ alrededor de un píxel en la imagen izquierda $I_{izq}$ y se desplaza una ventana del mismo tamaño a lo largo de la misma línea en la imagen derecha $I_{der}$, considerando el píxel homologo (correspondiente) aquel que tenga un menor valor de correlación. La correlación sobre cada punto de la imagen izquierda $I_{izq}(x^l, y^l)$ y cada valor de disparidad $d$ de la siguiente manera:

$$C_{SSD}(x^l, y^l, d) = \frac{1}{nm} \sum_{i=x^l-n/2}^{x^l+n/2} \sum_{j=y^l-m/2}^{y^l+m/2} (I_{izq}(i,j) - I_{der}(i-d,j))^2$$

De esta manera se obtiene el mapa de disparidad denso $D_{calculada}(u, v)$ dado un par de imágenes estéreo:

$$D_{calculada}(x^l, y^l) = \arg \min_{d \in [0,D]} C_{SSD}(x^l, y^l, d)$$

Donde $D$ es el valor de disparidad máximo esperado en la escena. La Figura 10 muestra el par de imágenes estéreo para la escena Cones de Middlebury Stereo Dataset 2003 con su mapa de disparidad real. La figura 11 muestra el mejor resultado obtenido mediante SSD. Éste resultado se obtiene utilizando una ventana de $7 \times 7$ y un valor de disparidad máxima esperada $D = 60$.

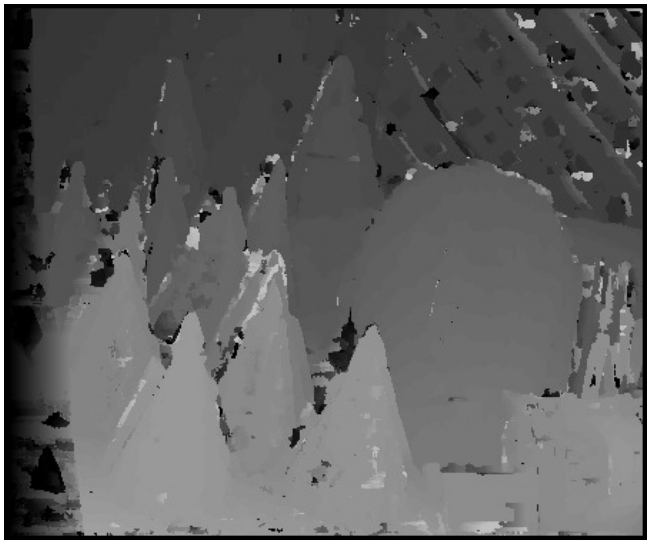

Figura 11: Mapa de disparidad para *Cones* estimado mediante SSD

### 2.2.2. Block Matching

El uso de ésta técnica está muy extendido ya que se encuentra implementada en la conocida biblioteca de funciones de visión artificial OpenCV (`http://opencv.willowgarage.com`). Se basa en el trabajo realizado en [8]. En dicho trabajo se pone de manifiesto que la correlación entre las áreas de las imágenes se ve afectada por variaciones de iluminación, perspectiva y diferencias de imagen entre las cámaras (la cámara izquierda percibe el color o la intensidad de forma ligeramente diferente a la cámara derecha). Para compensar estos efectos, en lugar de utilizar las imágenes en crudo para encontrar las correspondencias, se propone encontrar las correspondencias sobre las imágenes transformadas de alguna manera.

De los diferentes tipos de transformaciones disponibles, el autor de [8] propone utilizar la transformación Laplacian of gaussian (LOG) para normalizar las imágenes, resaltando sus bordes, y utilizar la distancia L1 (diferencia absoluta) para calcular la correlación entre dos ventanas de imagen. Para optimizar los cálculos, se fuerza a que la ventana tenga el mismo alto y el mismo ancho $n \times n$ y que el valor máximo de disparidad esperado $D$ sea múltiplo de $16$. De esta manera, la fórmula que representa la correlación es:

$$C_{BM}(x^l, y^l, d) = \frac{1}{n^2} \sum_{i=x^l-n/2}^{x^l+n/2} \sum_{j=y^l-n/2}^{y^l+n/2} |I_{izq}(i,j) - I_{der}(i-d,j)|$$

Y el mapa de disparidad se obtiene:

$$D_{calculada}(x^l, y^l) = \arg \min_{d \in [0,D]} C_{BM}(x^l, y^l, d)$$

Una vez obtenido el mapa de disparidad denso, se aplica un filtro para suavizar el resultado. La figura 12 muestra el resultado obtenido de aplicar éste método a la escena Cones de Middlebury Stereo Dataset 2003, utilizando su implementación en OpenCV y los parámetros por defecto.

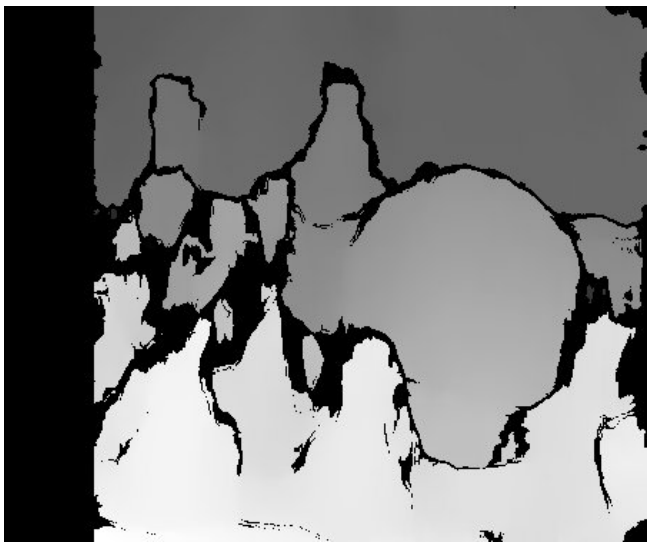

Figura 12: Mapa de disparidad para Cones estimado mediante BM

### 2.2.3. Graph Cut

En los últimos años se ha extendido mucho el uso de Graph Cut [1] en la aproximación de los mapas de disparidad densos. Se trata de una familia de métodos que, en lugar de utilizar los valores de intensidad alrededor de un pixel en una ventana finita (local) de píxeles vecinos, utilizan el global de la imagen asumiendo explícitamente cierta suavidad sobre el mapa de disparidad y utilizando varias técnicas de minimización para obtenerlo.

Estas técnicas son atractivas por los excelentes resultados experimentales que están obteniendo [5]. En estas aproximaciones el problema de la correspondencia estéreo se formula como un problema de minimización de energía, el cual suele incluir: (i) smoothness energy que mide la suavidad entre pares de píxeles vecinos; (ii) data energy que mide la diferencia entre píxeles correspondientes en base a las disparidades asumidas. Se construye un grafo con pesos en el cual cada nodo representa un píxel de la imagen, un conjunto de etiquetas representan todos los posibles valores de disparidad y los pesos de las aristas del grafo corresponden con los valores de energía definidos. La técnica Graph Cut se utiliza entonces para aproximar la solución óptima, asignando a cada píxel su valor correspondiente de disparidad.

Normalmente se utiliza como base la imagen de disparidad obtenida utilizando la correlación introducida en el apartado 2.2.2 para la construcción inicial del grafo.

La principal limitación de este método es que su complejidad computacional lo convierte en inservible para aplicaciones de tiempo real.

La Figura 13 muestra el resultado de aplicar éste método a la escena Cones de Middlebury Stereo Dataset 2003, utilizando su implementación en OpenCV y los parámetros por defecto.

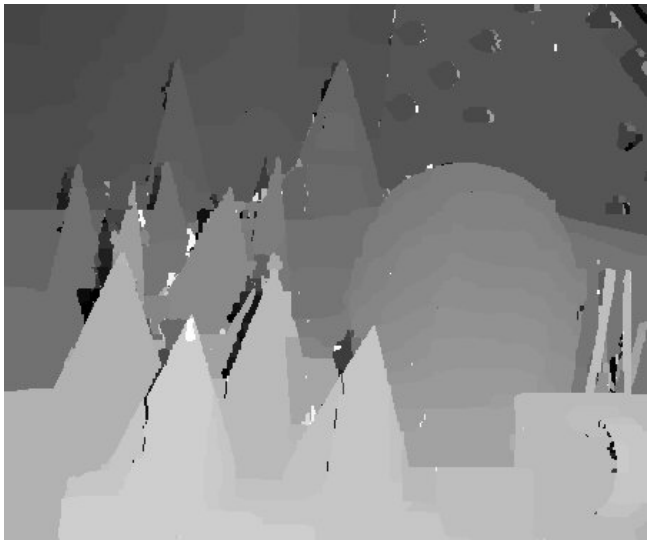

Figura 13: Mapa de disparidad para Cones estimado mediante GC

## 2.3. Medida de calidad de los mapas de disparidad densos

Un algoritmo de correspondencia estéreo perfecto debería generar exactamente el mismo mapa de disparidad que el mapa de disparidad real de la escena. Por ejemplo, en el caso de la escena Cones de Middlebury Stereo Dataset 2003, el mapa de disparidad generado debería ser exactamente el mismo que el que se puede observar en la Figura 10. De ser éste el caso, entonces para cada píxel del mapa de disparidad real $D_{real}(x, y)$ la diferencia absoluta con el mismo píxel en el mapa de disparidad calculado $D_{calculada}(x, y)$ debería ser $0$. Así pues:

$$\sum_{x,y} |D_{real}(x, y) - D_{calculada}(x, y)| = 0$$

Desafortunadamente, este no es el caso en la realidad, no todos los píxeles se calculan correctamente y por lo tanto la diferencia absoluta entre un píxel del mapa de disparidad real y su correspondiente en el mapa de disparidad calculado es mayor que $0$.

Por ese motivo, en la literatura se considera una tolerancia $t$. Por ejemplo, si se considera un valor de tolerancia $t = 2$ significa que se considera correcto todo píxel en el mapa de disparidad calculado cuyo valor de disparidad $D_{calculada}(x, y)$ caiga dentro de un valor de $D_{real}(x, y) \pm 2$. Una medida de calidad que se puede tomar es el porcentaje de píxeles correctos en el mapa de disparidad calculado, es decir, el porcentaje de píxeles $(x, y)$ tales que $|D_{real}(x, y) - D_{calculada}(x, y)| \leq t$

La Figura 14 muestra cual sería la representación gráfica del porcentaje de píxeles correctos respecto del valor de tolerancia $t$ en el caso de un algoritmo de correspondencia estéreo perfecto.

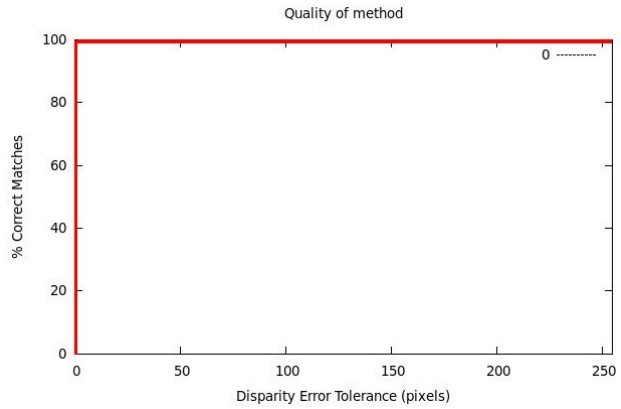


Figura 14: Para un algoritmo de correspondencia estéreo perfecto el error de disparidad en todos los píxeles es 0

Observando las figuras 11, 12 y 13 se puede llegar a la conclusión de que los mejores resultados se obtienen utilizando el método basado en Graph Cut. Esto se puede comprobar más claramente si se realiza la representación gráfica del porcentaje de píxeles correctos respecto del valor de tolerancia $t$ con el resultado obtenido de aplicar los diferentes métodos: SSD, BM y GC. Véase las figuras 15, 16 y 17.

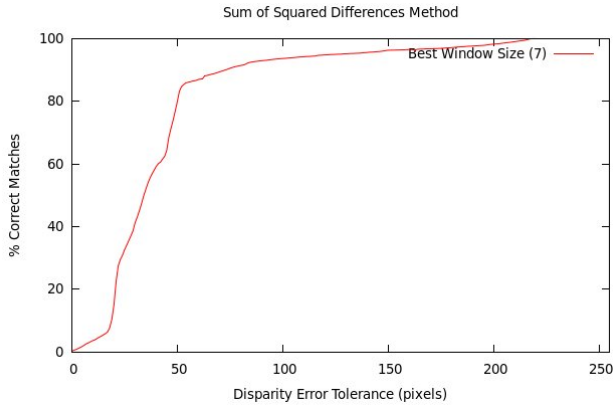


Figura 15: Error de disparidad utilizando SSD

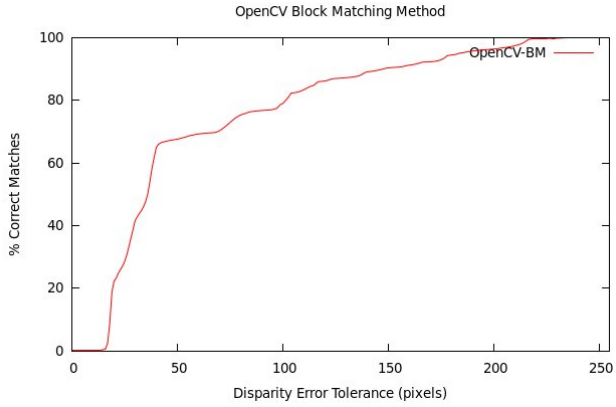


Figura 16: Error de disparidad utilizando BM (parámetros por defecto en OpenCV)

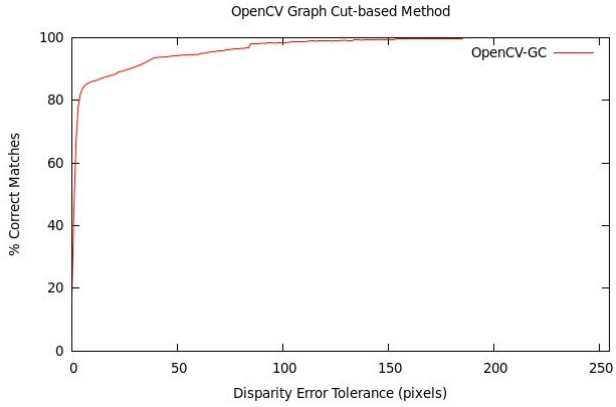


Figura 17: Error de disparidad utilizando GC (parámetros por defecto en OpenCV)

Las gráficas confirman que el mejor resultado se obtiene utilizando el algoritmo basado en Graph Cut.

En el presente trabajo se va a utilizar como medida de calidad la misma medida que se utiliza en la mayor parte de trabajos en la bibliografía.

Se utilizará el porcentaje de píxeles correctos dado un valor de tolerancia $t = 1$. Es decir, se considerará un píxel como correcto siempre que

$$|D_{real}(x, y) - D_{calculada}(x, y)| \leq 1$$

# 3. Desarrollo del Trabajo

El objetivo de este trabajo va a consistir en llevar el cálculo de correspondencias estéreo al campo del aprendizaje automático. Se considerará este problema como un mero problema de clasificación, se definirá una ventana de interés alrededor de un píxel en la imagen izquierda y derecha, se calcularán unas determinadas características y un clasificador se encargará de predecir el valor de disparidad del píxel en consideración.

Se va a asumir que los pares de imágenes estéreo están reajustados y rectificados tal y como se ha explicado en el punto introductorio de este trabajo, permitiendo así tomar ventaja de las peculiaridades de este tipo de imágenes.

Con esto se espera conseguir resultados capaces de competir con los resultados de los métodos de cálculo de correspondencias clásicos. Además, se espera poder conseguir rendimiento en tiempo real en un futuro gracias al hecho de que la predicción del valor de disparidad de cada píxel es independiente de los demás píxeles, permitiendo la paralelización del algoritmo llevando a cabo a la vez la predicción de todos los píxeles en una GPU.

Se va a construir un clasificador multi-clase en el cual cada clase va a representar un valor diferente de disparidad. Por ejemplo, si se considera que el número máximo de valores de disparidad diferentes es $60$, entonces el clasificador podrá discernir entre $60$ clases diferentes.

Dada la limitación en los datos de entrenamiento disponibles se va a proponer un método original de generación de imágenes sintéticas con disparidad conocida. Los patrones generados por dicho método se utilizarán para el entrenamiento del clasificador, el modelo aprendido se aplicará a los conjuntos de imágenes reales y los resultados serán comparados con los mapas de disparidad reales.

En este trabajo se va a utilizar la escena Teddy del Middlebury Stereo Dataset 2003 como referencia para los experimentos a realizar. Véase la Figura 18.

Figura 18: Ejemplos de Middelbury Stereo Dataset 2003, escena *Teddy* con mapa de disparidad denso real.

## 3.1. Patrones y características para el aprendizaje

Bajo el supuesto de que los pares de imágenes estéreo están reajustados y rectificados, para cualquier punto de la imagen izquierda $I_{izq}(x^l, y^l)$ su punto correspondiente en la imagen derecha $I_{der}(x^r, y^r)$ aparecerá en la misma línea en ambas imágenes $y^l = y^r$. Además, puesto que el plano de imagen de las cámaras es paralelo, se asegura que para un píxel en la imagen izquierda su correspondiente píxel en la imagen derecha aparecerá entre la misma columna y la misma columna menos el valor de disparidad máximo, es decir:

$$x^l - D \leq x^r \leq x^l$$

De esta manera, la búsqueda de correspondencias para un píxel en la imagen izqueirda en lugar de realizarse en toda la imagen derecha, se reduce a una porción (de tamaño $D$ píxeles) de una sola línea.

Normalmente, se consigue cumplir las premias de imágenes reajustadas y rectificadas mediante un proceso de calibración de las cámaras. Dicho proceso puede ser imperfecto y contener errores que hagan que el píxel correspondiente en la imagen izquierda no aparezca exactamente en la misma linea en la imagen derecha, sino algunas líneas más arriba o más abajo. Por este motivo se va a diseñar la ventana de atención para que incluya algunos píxeles de más y de esta manera poder tolerar calibraciones imperfectas. Se considera que un tamaño de ventana de $5 \times 5$ alrededor de un cierto píxel en la imagen izquierda debería ser suficiente para tolerar pequeños errores en la calibración de las cámaras. Para la imagen derecha se define una ventana rectangular con $alto = 5$ y $ancho = D + alto$. La concatenación de las ventanas de interés de la imagen izquierda y derecha serán los patrones que se utilicen para el aprendizaje/clasificación la etiqueta de clase de cada patrón será el valor de disparidad del punto de interés. La Figura 19 pretende clarificar un poco más este concepto.

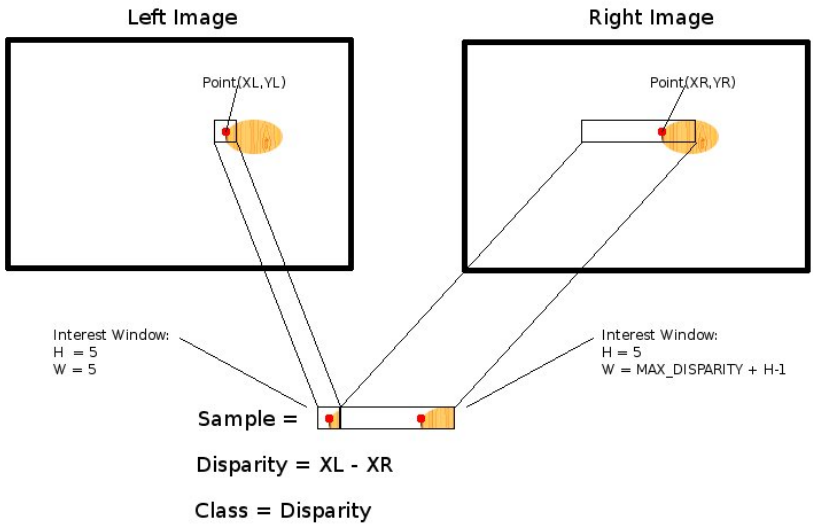


Figura 19: Esquema aclaratorio de cómo se obtienen los patrones para aprendizaje/clasificación

Una vez introducido el tipo de patrones que se utilizarán en el proceso de aprendizaje y clasificación se van a introducir las características concretas que se van a extraer de dichos patrones. En concreto estas características o *features* están inspiradas en las características utilizadas en el algoritmo de detección de objetos de Viola-Jones y que son de alguna manera similares a las *Haar features* [16]. El valor de cada característica se calculará como la simple diferencia absoluta entre columnas de la ventana de atención de la imagen izquierda y columnas de la ventana de atención de la imagen derecha. Véase la Figura 20

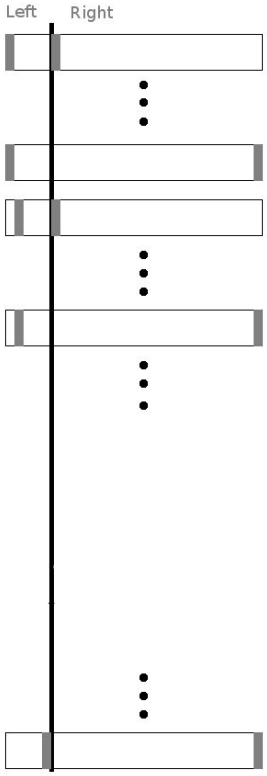


Figura 20: Se utilizarán como características los valores de la diferencia absoluta entre las zonas sombreadas en gris

Es de esperar que las columnas que contengan porciones de imagen muy similares resulten en un valor de diferencia absoluta bajo, de la misma manera columnas que contengan porciones de imagen diferentes resulten en un valor de diferencia absoluta alto.

El número total de características dependerá del $alto$ del patrón y el número máximo de posibles valores de disparidad $D$. El total de características a calcular es:

$$n = alto(D + alto)$$

Como se ha comentado con anterioridad, durante este trabajo se va a tomar como referencia la escena Teddy de Middlebury Stereo Dataset 2003, cuyo valor de

disparidad máximo es $63$. Así pues, en lo que resta de trabajo se va a considerar que el número de clases diferentes a clasificar es 63 y que el número total de características a calcular para cada patrón es:

$$n = 5 * (63 + 5) = 340$$

A continuación se muestra el pseudo-código para calcular el valor de las características.

```
CalcularCaracteristicas( Patron, alto, D, Vector_C )
{
  ancho = 2*alto + D;

  for (col_izq = 0; col_izq < alto; col_izq++)
  {
    for (col_der = alto; col_der < ancho; col_der++)
    {
        Vector_C[col_izq*ancho + col_der] =
          abs(Patron(col_izq)-Patron(col_der));
    }
  }

}
```

Basándose en estas características, el algoritmo de aprendizaje deberá calcular un modelo de predicción e intentar predecir el valor de disparidad para nuevos patrones.

## 3.2. Generación de patrones sintéticos aleatorios

Como se ha comentado con anterioridad, uno de los impedimentos más grandes a la hora de aplicar métodos de aprendizaje automático a este tipo de problemas es la falta de datos con la etiqueta de clase real. Para atajar éste problema se propone utilizar patrones generados sintéticamente de manera aleatoria utilizando los conocimientos disponibles sobre visión estéreo.

Se sabe que bajo las premisas asumidas en este trabajo, para un punto en la imagen izquierda su correspondiente punto en la imagen derecha aparecerá en la misma linea y además acotado entre unas columnas conocidas. Refierase al apartado anterior para mas detalles. Con este conocimiento se pueden generar patrones sintéticos aleatoriamente de la siguiente manera:

1. Crear una imagen de dimensiones $(D + 2 * alto) \times alto$. En lo que resta de trabajo estas dimensiones van a ser, dado que $alto = 5$ y $D = 63$: $73 \times 5$.
2. Rellenar la parte del patrón correspondiente a la imagen izquierda (cuadrado de tamaño $alto \times alto$) con valores de color aleatorios.
3. Copiar la parte del patrón correspondiente a la imagen izquierda a la parte del patrón correspondiente a la imagen derecha con un $desplazamiento = ancho - alto - disparidad$. Donde $disparidad$ es el valor de disparidad que se desea que el patrón tenga.
4. Llenar el resto del patrón con más valores de color aleatorios.

Para esclarecer el método anterior, en las siguientes figuras se plantea un ejemplo en el que se va a construir un patrón sintético aleatorio que tenga un valor de $disparidad = 20$:



Figura 21: Paso 1: Crear una imagen de $73 \times 5$ píxeles.

Figura 22: Paso 2: Rellenar parte izquierda con valores de color aleatorios.

Figura 23: Paso 3: Copiar la parte izquierda en la parte derecha con un $desplazamiento = 73 - 5 - 20$, agregando algo de ruido opcionalmente. La $disparidad$ de este patrón sera 20.

Figura 24: Paso 4: Rellenar el resto de píxeles con valores aleatorios.

Si se aplica el cálculo de características explicado en el punto anterior se observa que, tal y como se espera, el valor de la característica sera cercano a $0$ en las columnas parecidas entre ellas y sera mayor para columnas con mayores diferencias.

Este método de generación de patrones sintéticos aleatorios es muy sencillo de implementar y permite generar tantos patrones aleatorios como sean necesarios. De esta manera se va a proceder a la generación de miles de patrones con valores de disparidad asignados entre 0 y 63. Estos patrones generados se utilizaran como datos para el entrenamiento y el test del método de aprendizaje automático.

Se adjuntan, para satisfacer la curiosidad del lector, algunos ejemplos de patrones aleatorios generados mediante la técnica expuesta con diferentes valores de disparidad.

Figura 25: Patrón sintético aleatorio con $disparidad = 0$.

Figura 26: Patrón sintético aleatorio con $disparidad = 31$.

Figura 27: Patrón sintético aleatorio con $disparidad = 62$.

## 3.3. Método de aprendizaje y clasificación

La tarea de clasificación que se plantea en este trabajo requiere de un clasificador multi-clase capaz de obtener buenos resultados de manera rápida y eficiente. Para esta tarea, se considera que el clasificador basado en la teoría de *random forest*[3] o *bosque aleatorio* es el más adecuado ya que ha demostrado ser unos de los mejores clasificadores en los últimos años en variedad de aplicaciones[15], aunque no se haya estudiado en profundidad en este Máster.

En particular los *random forest* pueden aprender más de una clase a la vez simplemente contando los votos en cada una de las hojas de todos los arboles, seleccionando como clase ganadora aquella que reciba el número máximo de votos. También se pueden utilizar en problemas de regresión calculando la media de las hojas a través de los diferentes arboles. *Random forest* se basa en arboles de decisión alterados aleatoriamente, heredando muchas de las características beneficiosas de estos métodos. *Random forest* también tiene el potencial de ser implementado paralelamente, incluso en sistemas de memoria no compartida, haciéndolo aun más atractivo para su uso en problemas de tiempo real. El subsistema básico sobre el que se implementan los *random forest* los arboles de decisión. Estos arboles de decisión se construyen completamente hasta que se adaptan de manera perfecta a los datos de entrenamiento. Cada árbol es un clasificador de alta variabilidad que aprende casi perfectamente los datos de entrenamiento. Para contrarrestar este efecto se calcula la media entre muchos arboles (de ahí el nombre *forest* o *bosque*).

Por supuesto, el calcular la media entre muchos arboles no tiene ningún efecto si los arboles son muy similares entre sí. Por eso, *random forest* obliga a que cada árbol sea diferente mediante la selección aleatoria de un grupo diferente de características sobre las que es posible aprender en cada nodo. En el problema tratado en este trabajo se pueden utilizar hasta 340 características diferentes. A cada nodo de un árbol se le permite escoger de un subconjunto aleatorio de dichas características mientras determina la mejor manera de dividir los datos de entrenamiento y cada nodo siguiente del árbol obtiene un nuevo subconjunto de características determinado aleatoriamente sobre los que aprender. El tamaño de estos subconjuntos se suele escoger como la raíz cuadrada del número de características. De esta manera, en el problema en cuestión teniendo 340 posibles características cada nodo

escogerá aleatoriamente cerca de 18 y escogerá la mejor manera de clasificar los datos de entre esas 18 características. Para conseguir un rendimiento más robusto, *random forest* divide internamente los datos de entrenamiento de manera aleatoria en dos subconjuntos: entrenamiento y validación. Los datos en el subconjunto de validación se utilizan para estimar la validez de la clasificación. El tamaño del subconjunto de datos de validación se suele establecer en un tercio del total de datos.

Al igual que los métodos basados en arboles, *random forest* hereda muchas de las propiedades beneficiosas de estos métodos: sustitución para valores ausentes, capacidad para manejar tanto datos categóricos como numéricos, no necesidad de normalizar los valores y métodos sencillos para encontrar cuales son los valores más importantes para la predicción. Además, *random forest* utiliza los datos de validación para estimar la calidad de los resultados frente a datos no vistos durante el entrenamiento. Si los datos de entrenamiento tienen una distribución similar a los datos de test, la predicción de la calidad obtenida mediante los datos de validación puede llegar a ser muy acertada.

Finalmente los *random forest* se pueden utilizar para determinar, para dos puntos de datos cualquiera, su *proximidad*. Es decir, cómo de parecidos son. El algoritmo hace esto mediante (1) *dejando caer* los datos en los arboles, (2) contando cuantas veces los datos acaban en la misma hoja y (3) dividiendo el número de veces que los datos *han caído* en la misma hoja por el número total de arboles. Un valor de proximidad de 1 significa que los datos son exactamente iguales y 0 significa que son completamente opuestos. Esta medida de proximidad puede utilizarse para identificar muy diferentes al resto (outliers) y también para agrupar los datos en diferentes conjuntos (agrupando puntos próximos entre sí).

Otro de los motivos que hace atractiva la utilización de este método es que existe una implementación en OpenCV [2] que permite lanzar los experimentos de manera sencilla y eficaz.

## 3.4. Experimentación con patrones sintéticos aleatorios

En este apartado se van a exponer los resultados obtenidos en los experimentos utilizando como muestras de entrenamiento los patrones sintéticos aleatorios presentados con anterioridad.

En todos los experimentos, como ya se ha comentado, se va a utilizar la escena Teddy de Middlebury Stereo Dataset 2003 vista en la Figura 18. Con motivo de facilitar la comparación visual de resultados la Figura 28 muestra el mapa de disparidad real para la escena Teddy.

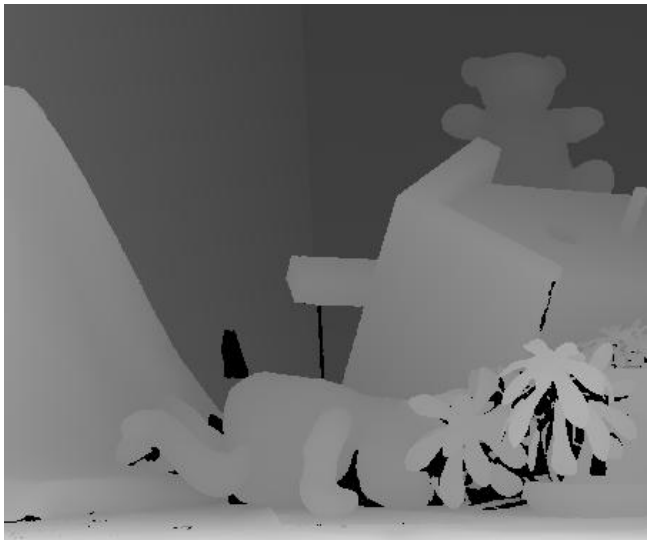

Figura 28: Mapa de disparidad real de la escena Teddy a efectos de comparación visual

Para el primer experimento, se van a utilizar pocas muestras de entrenamiento por cada clase para permitir que el proceso de entrenamiento se lleve a cabo rápidamente. Concretamente las condiciones del experimento son las siguientes:

- Entrenamiento: 160 muestras sintéticas por clase generadas aleatoriamente. Total: 10.080 muestras.
- Validación: 40 muestras sintéticas por clase generadas aleatoriamente. Total: 2.520 muestras.
- Test: Muestras extraídas de la escena Teddy ($450 \times 375$ píxeles). Total: 168.750 muestras

La Figura 29 muestra el mapa de disparidad denso generado por el clasificador recién entrenado.

Figura 29: Mapa de disparidad generado por el clasificador entrenado utilizando 160 muestras sintéticas generadas aleatoriamente.

Concretamente los resultados obtenidos en el experimento son los siguientes:

- Prueba sobre conjunto de validación: 100 % Píxeles Correctos.
- Prueba sobre conjunto de test (Teddy): 51.8 % Píxeles Correctos.

El resultado de la primera prueba es bastante esperanzador, ya que el número de muestras de entrenamiento sintéticas que se han utilizado ha sido muy bajo y el resultado obtenido sobre la escena real ha sido moderadamente bueno.

El hecho de que se alcance el 100 % de aciertos sobre el conjunto de validación indica que, posiblemente se esté produciendo un efecto de *overfitting* sobre los datos de entrenamiento. Es decir, el clasificador aprende perfectamente los datos de entrenamiento pero no generaliza bien para datos nuevos no vistos en la fase de entrenamiento.

Para intentar remediar esta situación se han probado las siguientes medidas:

- Utilizar más muestras sintéticas generadas aleatoriamente para intentar mejorar la variabilidad de los arboles. Se han hecho pruebas con 1000 y 10000 muestras de entrenamiento.
- Agregar ruido más agresivo en las muestras sintéticas generadas aleatoriamente.
- Aumentar el número de características calculadas.

Ninguna de las medidas expuestas ha conseguido una mejora significativa en el rendimiento del clasificador. Por lo cual se ha llegado a la conclusión de que las muestras sintéticas generadas aleatoriamente siguiendo el método expuesto en el apartado **3.2** no representa de manera óptima a las muestras provenientes de imágenes reales.

Analizando los valores que se obtienen al realizar el cálculo de las características sobre las muestras sintéticas generadas aleatoriamente y los valores que se obtienen

al realizar el cálculo de las características sobre las muestras procedentes de imágenes reales, se halla la explicación.

La Figura 30 muestra de manera gráfica el valor de las características calculado sobre muestras sintéticas generadas aleatoriamente. El eje horizontal representa el valor de cada una de las características (más oscuro significa que en ese punto de disparidad las imágenes izquierda y derecha son más parecidas, más claro significa que en ese punto de disparidad las imágenes izquierda y derecha son menos parecidas). Sobre el eje vertical se representan todas y cada una de las muestras utilizadas en el entrenamiento.

Figura 30: Valores obtenidos al realizar el calculo de las características sobre muestras sintéticas generadas aleatoriamente

Como se puede observar, existe una línea recta perfectamente observable que define el valor de disparidad de cada muestra.

En cambio, si se muestra de manera gráfica el valor de las características calculado sobre muestras obtenidas a partir de las imágenes reales (Teddy), Figura 31 :

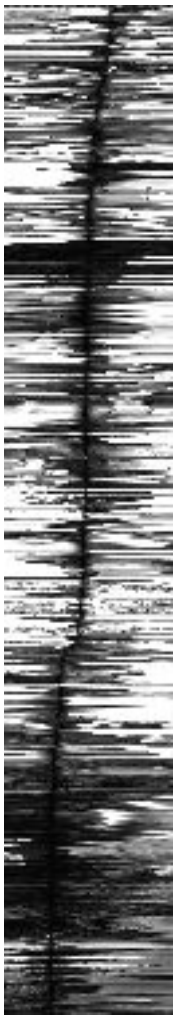

Figura 31: Valores obtenidos al realizar el calculo de las características sobre muestras obtenidas a partir de imágenes reales

Como se puede observar, la línea que representa la disparidad de las muestras se encuentra mucho más difuminada, en algunos casos es difícil de distinguir. Las muestras de entrenamiento sintéticas generadas aleatoriamente utilizadas en el primer experimento no son adecuadas para representar el tipo de muestras que se encuentran en imágenes reales.

Esto obliga a replantear la manera en la que los patrones sintéticos generados aleatoriamente son construidos.

## 3.5. Revisión del método de generación de patrones sintéticos aleatorios

Tal y como se ha visto en el apartado anterior, los patrones sintéticos generados aleatoriamente utilizando el método expuesto en el apartado **3.2** no son eficientes a la hora de entrenar un clasificador que se vaya a aplicar a imagenes reales. Cualquier intento de mejorar el comportamiento del clasificador utilizando este tipo de patrones como fuente de datos para entrenamiento, como por ejemplo agregar ruido o calcular más características ha resultado ser infructuoso.

Por ese motivo se plantea un cambio de estrategia. El problema es que los patrones sintéticos generados aleatoriamente no representan unos valores de características similares a los de una imagen real, como se desprende de las figuras 30 y 31. Para solucionar esto se propone utilizar como base imágenes reales en lugar de generar valores de color aleatorios.

Las imágenes de base se han tomado del repositorio del MIT-CSAIL [10] que contiene miles de imágenes de escenas variopintas, de interiores y exteriores, diferentes objetos y estructuras de escena, etc...

La Figura 35 muestra algunos ejemplos de estas imágenes.

Así el nuevo proceso para generar patrones sintéticos se definiría de la manera siguiente:

1. Seleccionar aleatoriamente una imagen de base y generar un patrón vacío. Véase Figura 32
2. Seleccionar aleatoriamente un trozo de la imagen de base de tamaño $(D - alto) \times alto$ y copiarlo en la porción del patrón correspondiente a la imagen derecha. Véase Figura 33
3. Seleccionar una ventana de tamaño $alto \times alto$ con $desplazamiento = ancho - alto - disparidad$ y copiarla en la porción del patrón correspondiente a la imagen izquierda. Véase Figura 34



Figura 32: Patrón vacio

Figura 33: Trozo de imagen seleccionado aleatoriamente incrustado en la parte derecha del patron.

Figura 34: Copiar la ventana desplazada segun la disparidad correspondiente a la parte izqueirda

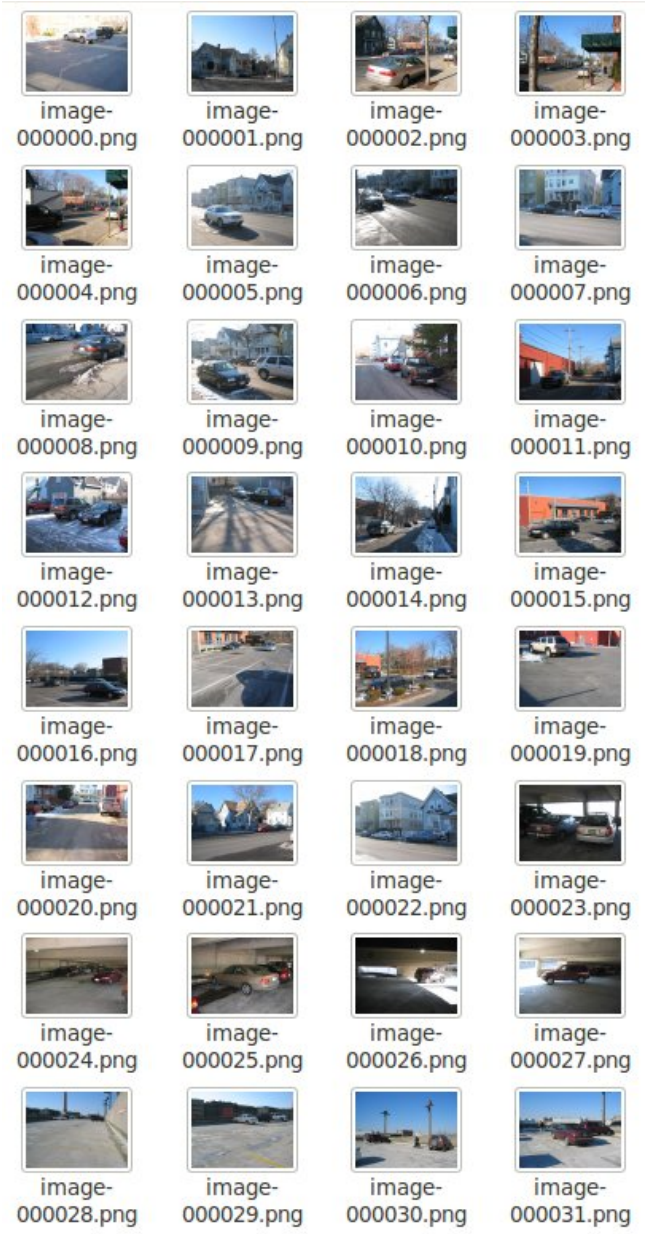


Figura 35: Algunos ejemplos del conjunto de imágenes del dataset MIT-CSAIL

A continuación algunos ejemplos de patrones generados mediante esta técnica:

Figura 36: Patrón sintético aleatorio basado en imagen real con $disparidad = 0$.

Figura 37: Patrón sintético aleatorio basado en imagen real con $disparidad = 31$.

Figura 38: Patrón sintético aleatorio basado en imagen real con $disparidad = 62$.

La Figura 39 muestra la representación gráfica de las características calculadas sobre muestras sintéticas generadas aleatoriamente con base a imágenes reales.

Como se puede observar la distribución de las características se corresponde mucho mejor con la de las imágenes reales que utilizando patrones completamente aleatorios como en el primer experimento. Gracias a esto, en principio, los resultados obtenidos al utilizar un clasificador entrenado con este tipo de patrones sobre la escena Teddy (imágenes reales) deberían ser mejores que los del primer experimento.

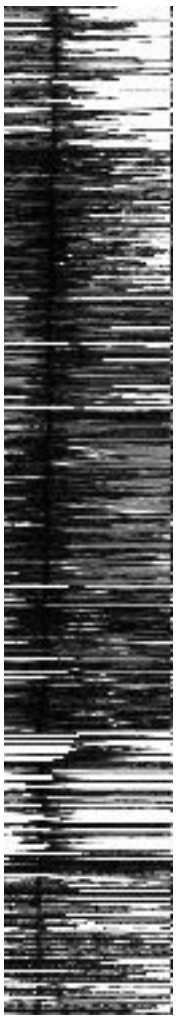

Figura 39: Valores obtenidos al realizar el calculo de las características sobre muestras sintéticas generadas aleatoriamente con base a imágenes reales

## 3.6. Experimentación con patrones sintéticos aleatorios modificados

Una vez introducidos los cambios en el método de generación de patrones sintéticos aleatorios para basarlos en imágenes reales se procede a realizar otro experimento de control. Nuevamente con pocas muestras de entrenamiento para agilizar el aprendizaje y comprobar a grandes rasgos la efectividad del método.

Concretamente las condiciones del experimento son las siguientes:

- Entrenamiento: 160 muestras sintéticas por clase generadas aleatoriamente en base a imágenes reales. Total: 10.080 muestras.
- Validación: 40 muestras sintéticas por clase generadas aleatoriamente en base a imágenes reales. Total: 2.520 muestras.
- Test: Muestras extraídas de la escena Teddy ($450 \times 375$ píxeles). Total: 168.750 muestras

La Figura 41 muestra el mapa de disparidad denso generado por el clasificador recién entrenado.

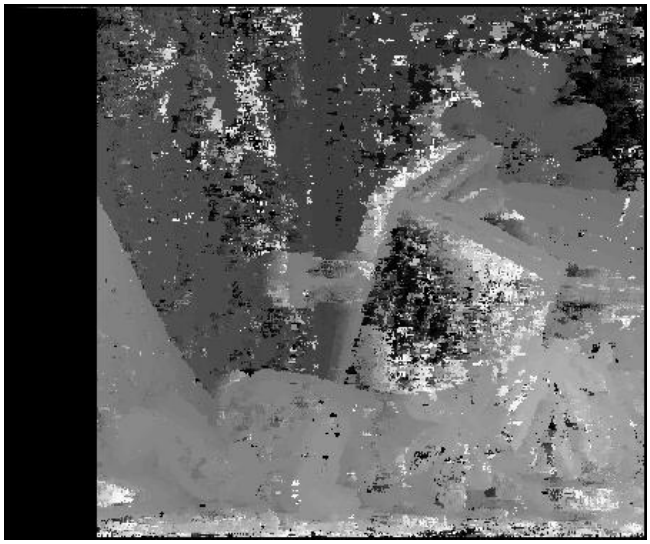

Figura 40: Mapa de disparidad generado por el clasificador entrenado utilizando 160 muestras sintéticas generadas aleatoriamente en base a imágenes reales.

Concretamente los resultados obtenidos en el experimento son los siguientes:

- Prueba sobre conjunto de validación: 72.3 % Píxeles Correctos.
- Prueba sobre conjunto de test (Teddy): 59.9 % Píxeles Correctos.

Tal y como se esperaba los resultados son sensiblemente mejores que utilizando los patrones sintéticos generados de manera puramente aleatoria. El 72.3 % de píxeles correctos sobre el conjunto de validación hace pensar que los problemas de sobre entrenamiento han disminuido. El tiempo de entrenamiento ha sido de apenas unos minutos y el test sobre las imágenes de prueba se realiza en pocos segundos.

A la luz de estos resultados se está en disposición de intentar mejorar la tasa de píxeles correctos enriqueciendo el conjunto de muestras de entrenamiento agregando mas muestras sintéticas generadas aleatoriamente en base a imágenes reales.

Para el siguiente experimento se va a considerar más muestras de entrenamiento. Los parámetros del experimento son los siguientes:

- Entrenamiento: 800 muestras sintéticas por clase generadas aleatoriamente en base a imágenes reales. Total: 50.400 muestras.
- Validación: 200 muestras sintéticas por clase generadas aleatoriamente en base a imágenes reales. Total: 12.600 muestras.
- Test: Muestras extraídas de la escena Teddy ($450 \times 375$ píxeles). Total: 168.750 muestras

La Figura 41 muestra el mapa de disparidad denso generado por el clasificador recién entrenado.

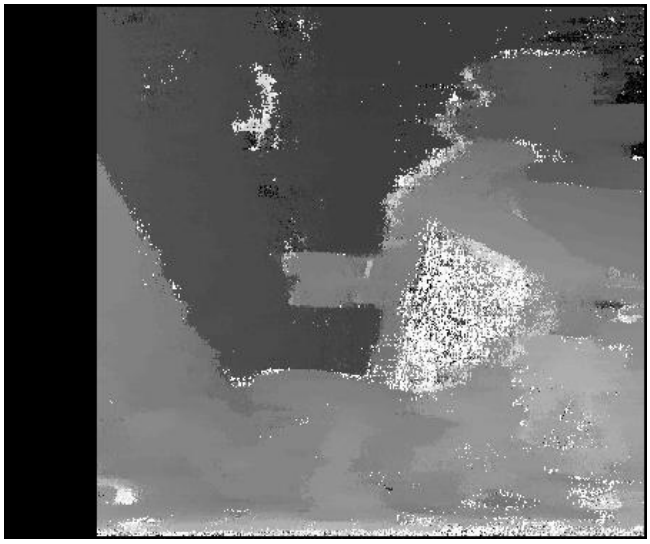

Figura 41: Mapa de disparidad generado por el clasificador entrenado utilizando 800 muestras sintéticas generadas aleatoriamente en base a imágenes reales.

Concretamente los resultados obtenidos en el experimento son los siguientes:

- Prueba sobre conjunto de validación: 84.1 % Píxeles Correctos.
- Prueba sobre conjunto de test (Teddy): 67.7 % Píxeles Correctos.

Si se construye la gráfica que relaciona el porcentaje de píxeles correctos con el valor de tolerancia $t$ del error de disparidad, Figura 42, se observa que el rendimiento de este método ya se encuentra por encima del mejor resultado obtenido mediante SSD.

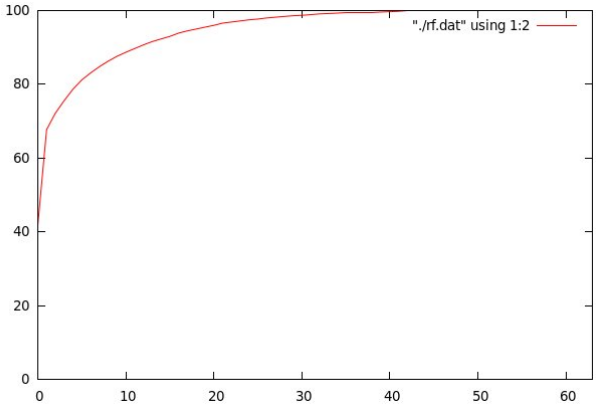


Figura 42: Error de disparidad utilizando el método propuesto

En este experimento el tiempo de entrenamiento ha aumentado considerablemente llevando cerca de una hora. También se observa que el algoritmo de correspondencia tiene problemas en zonas con alta texturización y zonas de oclusión, en un futuro trabajo se propondrá atacar estos problemas.

Para el último experimento en este trabajo se va a lanzar el entrenamiento con una cantidad considerable de muestras por cada clase para intentar conseguir resultados mejores que en experimentos anteriores.

En concreto se va a utilizar:

- Entrenamiento: 8000 muestras sintéticas por clase generadas aleatoriamente en base a imágenes reales. Total: 504.000 muestras.
- Validación: 2000 muestras sintéticas por clase generadas aleatoriamente en base a imágenes reales. Total: 126.000 muestras.
- Test: Muestras extraídas de la escena Teddy ($450 \times 375$ píxeles). Total: 168.750 muestras

La Figura 43 muestra el mapa de disparidad denso generado por el clasificador recién entrenado.

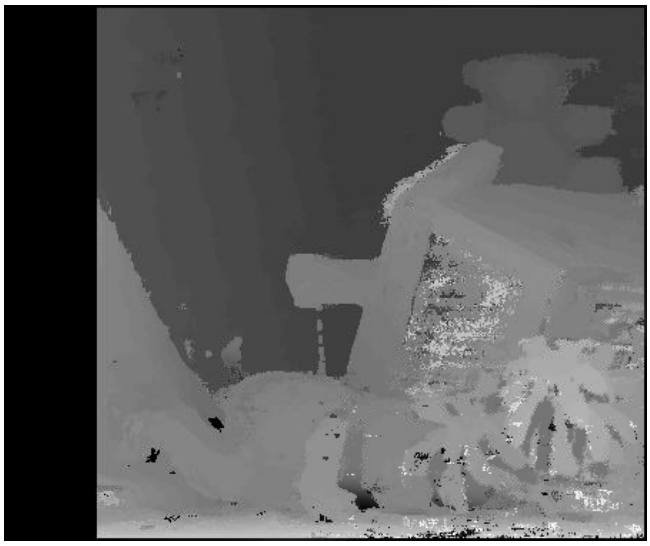

Figura 43: Mapa de disparidad generado por el clasificador entrenado utilizando 8000 muestras sintéticas generadas aleatoriamente en base a imágenes reales.

Concretamente los resultados obtenidos en el experimento son los siguientes:

- Prueba sobre conjunto de validación: 91.2 % Píxeles Correctos.
- Prueba sobre conjunto de test (Teddy): 90.3 % Píxeles Correctos.

Si se construye la gráfica que relaciona el porcentaje de píxeles correctos con el valor de tolerancia $t$ del error de disparidad, Figura 44, se observa que el rendimiento de este método ya se encuentra a la par del resultado obtenido mediante la técnica Grap Cut. Se siguen observando ciertos problemas en zonas de alta texturización y zonas ocluidas, lo cual refuerza la intención de seguir investigando el motivo de éstos fenómenos y hallar una solución que convierta el método presentado en este trabajo en mucho más preciso. El motivo más probable de este suceso sea que en zonas de alta texturización se encuentre más de un punto correspondiente entre la imagen izquierda y la imagen derecha, haciendo que el algoritmo de clasificación falle.

Para poder solucionar este problema, en un futuro se deberían estudiar nuevas características a extraer o incluso replantearse la manera de obtener los patrones para introducir más contexto de la imagen izquierda que pueda ayudar a discernir la clase de disparidad correcta.

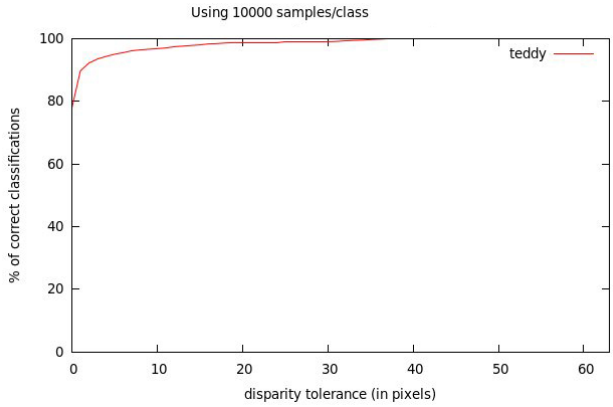


Figura 44: Error de disparidad utilizando el método propuesto

Se ha desistido de realizar experimentos con más datos de entrenamiento puesto que el tiempo que lleva el proceso de entrenamiento los hace impracticables con los recursos que se dispone (un ordenador de sobremesa estándar). La ventaja de random forest es que permite que se pueda paralelizar también el proceso de entrenamiento en un cluster de ordenadores [15]. Lo cual queda propuesto para un futuro trabajo.

# 4. Conclusiones y Futuras Líneas de Investigación

En este trabajo se han argumentado las ventajas que supondría contar con un método de cálculo del mapa de disparidad denso de un par estéreo, siendo una de las ventajas más importantes el hecho de que podría reforzar la base para nuevos métodos de visión artificial en 3 dimensiones.

Se ha considerado el problema del cálculo del mapa de disparidad de un par de imágenes estéreo desde una perspectiva poco estudiada hasta el momento. Se ha propuesto resolverlo desde el punto de vista del aprendizaje automático. Estudiando la manera de poder llevar el cálculo de correspondencias estéreo al terreno de los conocimientos adquiridos en el Máster. Esto se consigue mediante el cálculo de características basadas en diferencias absolutas entre columnas de imagen, que son posteriormente utilizadas por el algoritmo de aprendizaje *random forest*.

Se ha argumentado el uso de la técnica de aprendizaje automático basada en *random forest*, siendo ésta una técnica aplicada de manera muy satisfactoria en multitud de problemas de aprendizaje y clasificación.

Se ha analizado la problemática de disponer de un conjunto limitado de muestras de entrenamiento con valores reales de disparidad conocidos y se ha propuesto una técnica para generar fácilmente grandes cantidades de datos sintéticos de entrenamiento válidos para extrapolar los resultados al mundo real.

Se han desarrollado experimentos que confirman la validez de los métodos propuestos en este trabajo y se abre la puerta a futuras investigaciones en varios aspectos:

- Paralelización del algoritmo. Se puede explotar la posibilidad de paralelización de la ejecución del clasificador basado en *random forest*. Además, dado el hecho de que cada píxel del mapa de disparidad se puede calcular de manera totalmente independiente del resto, se espera obtener una aceleración considerable con respecto a la implementación secuencial. Una implementación inteligente utilizando una GPU no debería tener problema en alcanzar rendimiento de tiempo real sin tener que sacrificar la precisión en las predicciones de la disparidad.

- Mejora de los métodos de generación de muestras sintéticas validas para su uso en entornos reales de manera que dichas muestras sintéticas representen de una manera mas fiel la distribución de las muestras que se puedan obtener en imágenes reales.

- Mejora de las tasas de clasificación optimizando el método de clasificación utilizado en este trabajo o explorando otros métodos de clasificación diferentes.

- Mejora de las tasas de clasificación introduciendo clases artificiales que se encarguen de explicar píxeles ocluidos o zonas con baja o alta textura.

# Referencias


[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 1st edition, October 2008.

[3] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[4] Guibas L. Heath K. Multi-person tracking from sparse 3D trajectories in a camera sensor network. *Second ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[5] Li Hong and G. Chen. Segment-based stereo matching using graph cuts. volume 1, pages I–74–I–81 Vol.1, July 2004.

[6] Andrew Howard. Real-Time Stereo Visual Odometry for Autonomous Ground Vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[7] D. H. Hubel and M. S. Livingstone. Segregation of form, color, and stereopsis in primate area 18. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 7(11):3378–3415, November 1987.

[8] Kurt Konolige. Small Vision Systems: Hardware and implementation. In *Proceedings of the International Symposium on Robotics Research*, 1997.

[9] Akash Kushal and Jean Ponce. Modeling 3D Objects from Stereo Views and Recognizing Them in Photographs. pages 563–574. 2006.

[10] MIT-CSAIL. Mit computer science and artificial intelligence laboratory. `http://www.csail.mit.edu/`, September 2011.

[11] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. volume 1, pages I–195–I–202, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[12] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[13] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 47:7–42, 2001.

[14] David Schleicher, Luis M. Bergasa, Manuel Ocaña, Rafael Barea, and Elena López. Real-time hierarchical stereo Visual SLAM in large-scale environments. *Robotics and Autonomous Systems*, April 2010.

[15] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. June 2011.

[16] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[17] Willowgarage. Point cloud library. `http://www.pointclouds.org`, September 2011.

[18] Microsoft Xbox. Kinect. `http://www.xbox.com/kinect`, September 2011.