Escola Politècnica Superior
de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FIN DE CARRERA

Título del TFC: Algoritmos de interpolación de vistas.

Titulación: Ingeniería Técnica de Telecomunicación, especialidad en Sistemas de telecomunicaciones**.**

Autor: Sergio Sánchez Berruga

Director: Francesc Tarrés Ruiz

Fecha: 02-09-2005

Título del TFC: Algoritmos de interpolación de vistas.

Autor: Sergio Sánchez Berruga

Director: Francesc Tarrés Ruiz

Fecha: 02-09-2005

Resumen

La visión estreoscópica es una de las areas más importantes dentro del tratamiento digital de imagen. Es posible extraer la información 3D de una escena mediante varias capturas sincronizadas de la misma desde diferentes puntos de vista. Gracias a esto es posible desarrollar aplicaciones 3D de teleimmersión.

El objetivo de este proyecto es la creación de vistas virtuales (rendering) a partir de la información obtenida por un sistema de captación multicamara. Dos métodos diferentes de rendering han sido analizados: El primero llamado reproyección directa, consiste en reproyectar todos los píxeles de la imagen original directamente sobre la nueva vista. El segundo método usa una etapa previa en la que las imágenes originales son rectificadas antes de ejecutar el rendering. Esta rectificación permite reducir la complejidad de la búsqueda de correspondencias entre vistas, pero introduce una cierta pérdida de información.

Este trabajo esta organizado 4 capítulos: En el Primero son revisados los conceptos teóricos fundamentales sobre los sistemas estereoscópicos. En el siguiente capítulo, son estudiados con detalle los dos métodos propuestos para generar las vistas virtuales. El capítulo 3 describe las herramientas y aplicaciones desarrolladas mientras que en el capítulo 4 son presentadas las conclusiones.

Title: View interpolation algorithms

Author: Sergio Sánchez Berruga

Director: Francesc Tarrés Ruiz

Date: 02-09-2005

Abstract

Stereo-vision is one of the most relevant areas in digital image processing. It is possible to extract 3D information from multiple simultaneous captures of a scene in order to develop 3D teleimmersive applications.

The objective of this project is the creation of a virtual view (rendering) of a scene from information obtained from a multicamera system. Two different approaches have been analyzed to perform the rendering: The first method, called**direct reproyection,** consists in reproyecting all the pixels of the original images directly**.** The second method makes use of a preprocessing block in which the original images are rectified before reprojecting. This rectification permits a reduction in the complexity when searching for correspondences between views, but on the other hand there is a certain loss of information.

This thesis is organized as follows: First the fundamental concepts of theory about the stereoscopic systems are reviewed. In the next chapter, the two proposed approaches for creating the virtual views are studied in detail. Chapter 3 illustrates the tools and the applications developed, whereas in chapter 4 and the conclusions are presented.

ÍNDICE

INTRODUCCIÓN

El análisis de escenas a partir de múltiples cámaras es una de las áreas del tratamiento digital de imagen en las que actualmente se está desarrollando una gran actividad. Podemos diferenciar dos de las aplicaciones principales:

1. *Captura de información 3D de la escena*. Consiste en extraer la información de profundidad de una escena 3D a partir del análisis de dos o más imágenes de la misma escena captadas desde diferentes puntos de vista mediante técnicas estereoscópicas

2. *Síntesis de imágenes intermedias*, para aplicaciones en sistemas inmersivos. Gracias a estos sistemas es posible mover el punto de vista del espectador remoto de forma continua y natural, así como, sintetizar un par de imágenes desde distintos puntos de vista para que el receptor pueda ver la escena 3D. Esta aplicación necesita a su vez de la primera para generar las imágenes intermedias.

Las aplicaciones que ofrecen estas técnicas en el mundo del tratamiento digital de imagen son muchas y variadas. Nuestro proyecto esta directamente relacionado con dos de ellas, las cuales se están actualmente aplicando o investigando en entornos reales:

1. **Telepresencia**: Sistemas de video-cámaras estéreo permiten operar en entornos peligrosos u hostiles con la máxima precisión.

2. **Realidad Virtual**: Como se ha mencionado anteriormente, la técnica denominada Realidad Virtual básicamente es una interacción usuario-ordenador en la que se generan las imágenes estereoscópicas en tiempo real, introduciendo al espectador en un escenario 3D artificial.

Actualmente existe una nueva tecnología emergente que engloba las dos aplicaciones anteriores, *la Teleinmersión,* basada en la transmisión, entre puntos distantes, de escenas sintetizadas tridimensionales y a tamaño real, representadas y renderizadas (es decir, dotadas de texturas y volúmenes) en tiempo real, empleando técnicas avanzadas de visión y gráficos digitales. Para ello se requiere la combinación eficaz de sistemas avanzados de telecomunicaciones con la utilización de tecnologías multimedia que permiten reconocer la presencia y el movimiento de formas tridimensionales (elementos, personas, entornos 3D) para ser proyectados en “entornos de inmersión” (salas con el equipamiento necesario) geográficamente distribuidos, en los que los usuarios pueden interactuar sensorialmente. Los usuarios pueden manipular datos, compartir simulaciones de objetos y experiencias como si estuvieran en la misma habitación, participar juntos en una simulación, etc. Para ello serían necesarios, además de

múltiple equipamiento para capturar la imagen y codificarla, unas gafas polarizadas y un dispositivo en la cabeza para variar la vista de acuerdo con el movimiento de la cabeza del usuario.

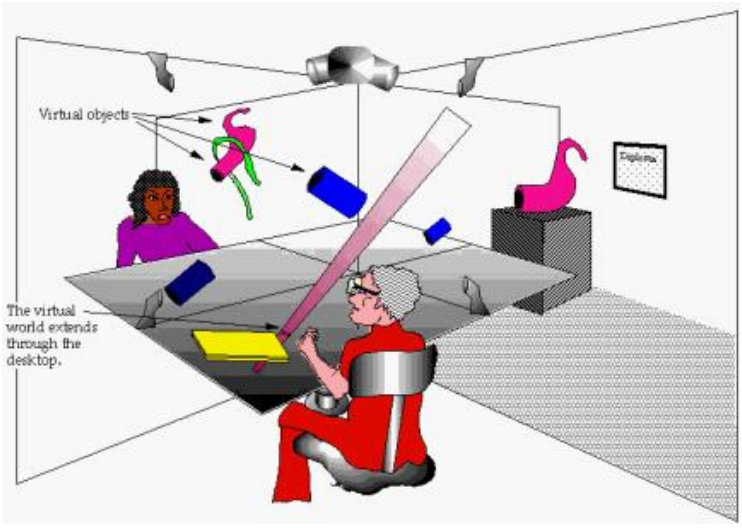


Fig. 0.1 Escenario Teleinmersivo

Esas réplicas de espacios tridimensionales con enormes cantidades de contenido visual están pensadas para ofrecer unos entornos más cercanos a la realidad, y por tanto prometen ser un revolucionario avance en las teleconferencias, para entornos de trabajo colaborativo en remoto (telepresencia tridimensional a tamaño real), así como en la industria del ocio, telemedicina, telemonitorización, información, etc. Son por tanto una evolución de los espacios de “realidad virtual” fusionados con la videoconferencia, diferenciándose de esta primera en la mayor cercanía a la realidad de los objetos y personas representados.

En nuestro proyecto vamos a realizar un estudio de una de las etapas necesarias en la teleinmersión, la interpolación de vistas. A partir de la información de una escena obtenida por un sistema de captación multicámara, es posible calcular la profundidad de la misma, así como interpolar las vistas no captadas por las cámaras.

Este tipo de aplicaciones teleinmersivas requieren sistemas con múltiples cámaras distribuidas de una forma específica. La siguiente figura muestra el ejemplo de uno de estos sistemas de captación multicámara, formado por 8 cámaras sincronizadas y distribuidas a lo largo de un arco de 30º.

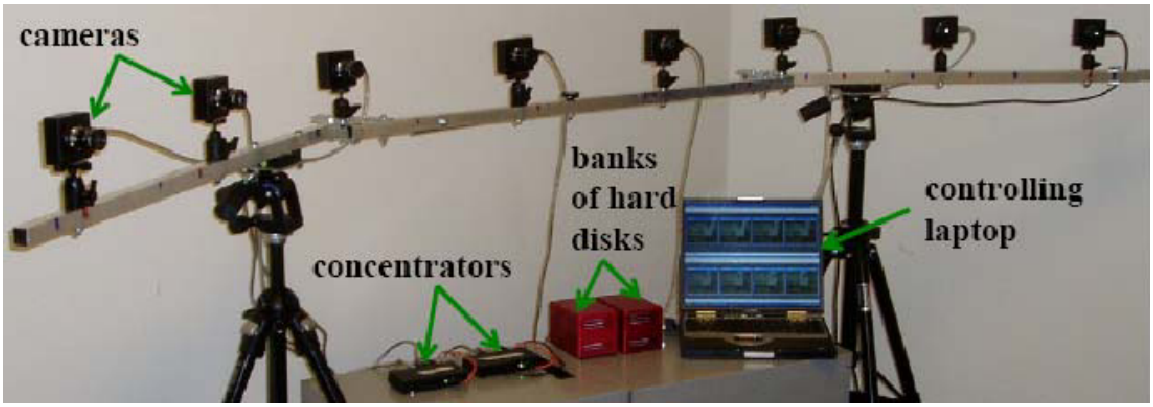
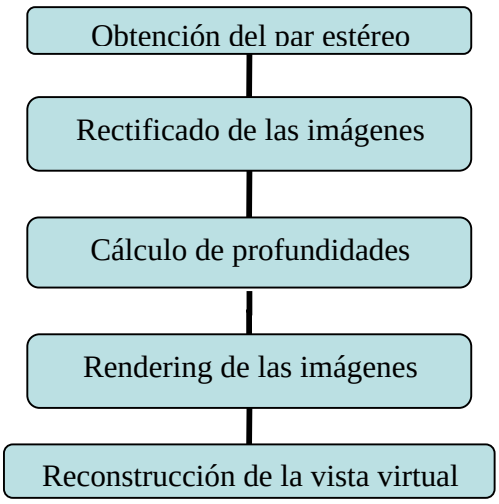


Fig. 0.2 Sistema de captación multicámara

Normalmente, el número de cámaras en estos sistemas es proporcional con la sensación de tridimensionalidad del entorno virtual. El principal problema es que un mayor número de imágenes o vistas implica mayores prestaciones económicas del sistema y más cantidad de información a codificar y transmitir; por este motivo, se suelen generar vistas intermedias entre cámaras mediante técnicas estereoscópicas

Este será el objetivo de este proyecto, generar las imágenes intermedias que existirían entre las cámaras de un sistema de captación como el anterior, con la finalidad de reproducir un video de la escena en el que sea posible variar el punto de vista del espectador, dentro del arco comprendido entre la primera y última cámara, ofreciendo una sensación de continuidad.

El siguiente esquema muestra el diagrama de bloques del proyecto:



Para entender este esquema haremos una breve explicación de cada uno de los módulos. Como se ha dicho anteriormente, queremos interpolar las imágenes intermedias que existen entre las cámaras estáticas de nuestro sistema de captación, es decir, queremos convertir un sistema de captación multicámara estático en una “cámara virtual” capaz de reproducir la escena desde cualquier punto de vista comprendido entre la primera y la última cámara de nuestro arco de visión.

Si cambiamos el punto de vista de la escena los objetos se desplazarán más o menos en función de su distancia a la cámara, esta es la idea básica de la estereoscopia. Por ello es necesario hacer un análisis mediante técnicas estereoscópicas de la escena para extraer la información 3D de la misma, es decir, extraer las profundidades de los objetos.

Para calcular las profundidades de una escena son necesarias dos o más imágenes de la misma captadas desde puntos de vista diferentes, para ello existen los llamados módulos estereoscópicos, se basan en dos o más sistemas de captación de imágenes digitales (binoculares, trinoculares, etc) sincronizados entre sí, capaces de obtener lo que se conoce como *par estéreo*.

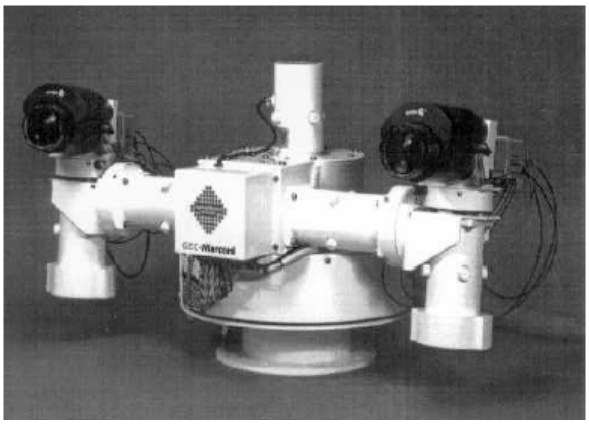


Fig. 0.3 Sistema de captación binocular

El cálculo de las profundidades de la escena a partir del par estéreo de la misma se reduce a un simple problema geométrico siempre y cuando las cámaras estén bien alineadas entre si y que los defectos de las lentes (distorsiones) se hayan corregido previamente, en la práctica esto nunca ocurre por lo que es necesario un módulo previo llamado *rectificación***.**

La rectificación se basa en la geometría epipolar descrita más adelante, se encarga de transformar mediante software las imágenes del par estéreo dotándolas de las rotaciones necesarias para conseguir alinear las líneas epipolares entre si. De esta manera conseguimos que un mismo punto de la

escena 3D visible en las dos imágenes, se encuentre en la misma posición vertical para las dos imágenes, esto significa que reducimos el problema de la búsqueda de correspondencias, tan importante en el cálculo de profundidades, a 1D.

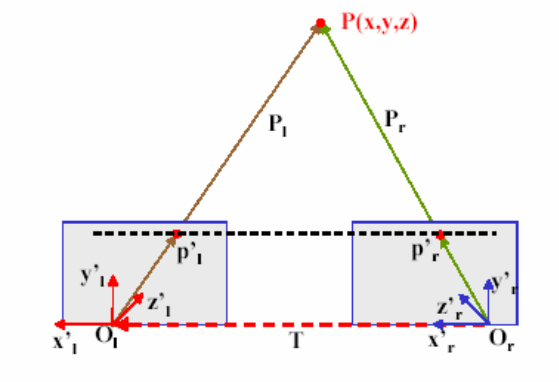


Fig. 0.4 Planos de imagen alineados

Una vez tenemos las imágenes rectificadas, se obtendrán los *mapas de disparidad* (imágenes de la escena en escala de grises, donde la información de la profundidad estará en la intensidad de la imagen), la idea de la profundidad es sencilla. Con las imágenes alineadas como muestra la figura (1.6), podemos conocer por trigonometría la profundidad de un punto de la escena a partir de su posición en ambas imágenes, por ello lo más importante de esta etapa es el cálculo de correspondencias entre píxeles, es decir, localizar para un píxel de la imagen izquierda cual es su posición en la imagen derecha, en función del desplazamiento sufrido obtendremos la profundidad del punto en la escena 3D.

Una vez tenemos calculados los mapas de profundidad, pasaremos a realizar el *rendering de las imágenes*. Esta etapa calcula, a partir de sus profundidades, el movimiento de los objetos al cambiar el punto de vista de la escena.

Habrá zonas que no serán visibles debido al movimiento no uniforme de los objetos. Lo que haremos es realizar el rendering tanto de la imagen derecha como de la izquierda del par estéreo, de esta manera las zonas no visibles por una cámara serán visibles por la otra, ya que el nuevo punto de vista siempre estará comprendido entre la cámara derecha e izquierda del par estéreo.

La última etapa será la *reconstrucción de la nueva vista*, para ello se realizará un filtrado de las dos imágenes renderizadas del que obtendremos la imagen final.

En la realización del proyecto nos hemos centrado en el estudio y valoración de los algoritmos de renderización de imágenes intermedias. Por ello, no hemos tratado otros tipos de problemas directamente relacionados con este tipo de aplicaciones como son la calibración de las cámaras y la estimación de las

imágenes de profundidad. Para poder comparar los resultados de nuestros algoritmos con otras alternativas hemos utilizado un conjunto de imágenes estándares, proporcionadas por Microsoft y que a menudo son utilizadas como referencia para la valoración de los algoritmos. Estas imágenes han sido captadas por 8 cámaras distribuidas a lo largo de un arco de 30º, están sincronizadas entre si y calibradas mediante la técnica de calibración propuesta en **[2]**, de esta calibración conocemos todos los parámetros. Las imágenes utilizadas y los parámetros de calibración se pueden encontrar en la web que aparece en **[3].**

Debido a la extensión del proyecto y la limitación temporal de la que disponemos trabajaremos con los mapas de profundidad generados por el grupo de trabajo de Microsoft cuyos resultados son adecuados a las necesidades de nuestro estudio. En el anexo quedan definidos los parámetros de calibración proporcionados por Microsoft.

Existen dos caminos a seguir para obtener las imágenes de las cámaras virtuales, el primero es trabajar con las imágenes rectificadas y el segundo trabajar con las imágenes sin rectificar. Nuestro objetivo será realizar un estudio sobre los dos métodos y obtener una conclusión, a partir de los resultados, sobre que método es más factible utilizar en posibles aplicaciones comerciales futuras.

Esta memoria esta organizada en 4 capítulos, detallados a continuación:

En el capítulo 1, **el estudio teórico**, se presentan los conceptos teóricos fundamentales sobre los sistemas estereoscópicos que constituyen la base científica de los algoritmos utilizados.

En el capítulo 2, **desarrollo de la aplicación**, se realiza un estudio de los parámetros necesarios para elaborar el proyecto, se sitúa el problema, y se explica detalladamente el desarrollo de los algoritmos utilizados.

En el capítulo 3, **descripción de las aplicaciones,** se da a conocer al lector los entornos visuales de las aplicaciones desarrolladas en el capítulo 2, así como una guía base sobre su funcionamiento.

En el capítulo 4, **conclusiones y líneas futuras**, aparecen las conclusiones extraídas de los resultados y de la elaboración de las aplicaciones, junto con una propuesta sobre líneas futuras de investigación.

CAPÍTULO 1. ESTUDIO TEÓRICO SOBRE LA ESTEREOSCOPÍA

1.1. El sistema visual humano

La percepción visual o visión permite a los seres vivos la comprensión o el conocimiento del medio ambiente donde se encuentran. Visión es el proceso por el que el cerebro es capaz de descubrir, a partir de imágenes, que es lo esta presente en el mundo y donde esta.

El sistema visual humano esta formado normalmente por dos receptores, los ojos, que deben funcionar totalmente coordinados para que los resultados perceptivos sean óptimos. Ambos ojos deben ser dirigidos exactamente al mismo punto del espacio. El sistema que comanda esta función se denomina *sistema de vergencias* y su accionar produce movimientos disyuntivos de los ojos acercándolos o alejándolos entre si de acuerdo a la distancia que se encuentre el objetivo.

Al dirigirse ambos ojos hacia el mismo punto del espacio el objeto observado producirá una imagen semejante en cada ojo aunque no exactamente igual ya que cada ojo lo percibe desde un lugar ligeramente diferente debido a la separación que hay entre ellos. Estas imágenes son utilizadas por el sistema visual para elaborar su más compleja función la percepción de *estereopsis*, es decir, la capacidad de reconocer profundidades espaciales con relativa precisión

1.2. Modelo de Cámara y Geometría 3D

El sistema de visión humano puede modelarse mediante un par de cámaras que representan cada uno de los ojos. Cada cámara está definida por un conjunto de parámetros que identifican su posición en el espacio tridimensional y sus posibles distorsiones.

En nuestro caso, para valorar los algoritmos de interpolación de vistas hemos utilizado imágenes de cámaras que habían sido calibradas previamente. No obstante, los parámetros de la cámara constituyen un aspecto fundamental en el cálculo de proyecciones y retroproyecciones de puntos del espacio en el proceso de renderización. Por ello, en estas secciones se detallan e interpretan los principales parámetros de calibración, así como la proyección de un punto 3D sobre la imagen. Toda esta teoría se puede encontrar en **[5]**.

Parámetros intrínsecos, son los parámetros propios de la cámara:

1. *Distancia Focal*: Es la distancia entre el plano de imagen y la lente de la cámara.

2. *Punto central*: Son las coordinadas en píxeles del punto central de la imagen, es decir, el punto que define el eje óptico en el plano de imagen.

3. La *distorsión radial* de la lente

4. Tamaño efectivo del píxel en milímetros.

Parámetros extrínsecos: Definen la relación espacial entre la cámara y el mundo real.

1. *Parámetros de rotación*: relacionan la orientación de la cámara respecto al sistema de coordenadas del mundo real.

2. *Parámetros de Translación*: relacionan la posición de la cámara respecto al sistema de coordenadas del mundo real.

Para explicar la proyección de un punto 3D en una imagen 2D trabajaremos con el modelo de "Pin-Hole", dicho modelo es una aproximación de la realidad pero para nuestro modelo de visión estereoscópica ya nos es suficiente. La geometría de este modelo se define en la figura (2.1), el plano h es el plano de imagen**,** sobre el cual se proyectan los puntos del mundo real y la línea vertical es el eje óptico. La lente esta situada perpendicular al eje óptico en el punto focal **C** (también llamado centro óptico). La distancia focal ***f*** es un parámetro de la lente y el punto $\mathbf{U_0}$ es el ya mencionado anteriormente punto central (parámetro intrínseco).

La proyección se realiza a través de un rayo óptico que es reflejado por un punto de la escena **X** y originado por una fuente de luz. Este rayo óptico pasa a través del centro óptico y queda proyectado sobre el plano de imagen, de aquí el nombre de "Pin-Hole", ya que todos los rayos atraviesan un mismo punto, el centro óptico.

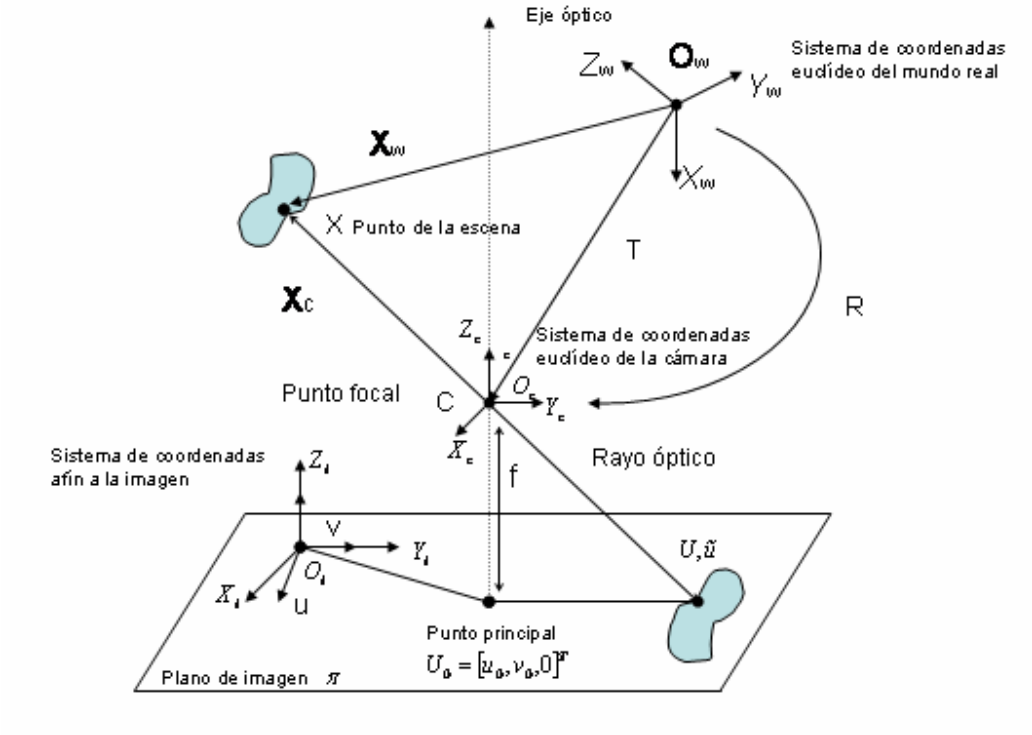


Fig. 1.1 Esquema de proyección de un punto 3D

En la imagen también se puede ver como quedan definidos los parámetros **R** de rotación y translación **T** respecto al sistema de coordenadas del mundo real.

Para entender el dibujo definiremos tres sistemas de coordenadas:

1. ***Sistema de coordenadas euclídeo del mundo real***: Es el sistema de referencia del mundo, su origen es O_w, y es a partir del cual se expresan los puntos 3D.

2. ***Sistema de coordenadas euclídeo de la cámara****:* Su origen es el centro de la cámara $O_c = C$. La relación entre el sistema de coordenadas del mundo real y el de la cámara queda definida por una rotación **R** y una translación **T.**

3. ***Sistema de coordenadas euclídeo de la imagen:*** Sus ejes están alineados con el sistema de referencia de la cámara, con X_i, Y_i extendidos sobre el plano de imagen.

4. ***Sistema de coordenadas afín a la imagen:*** Sus coordenadas son **u**, **v**, **w,** y su origen es O_i.

El hecho de utilizar el sistema de coordenadas afín a la imagen nos sirve para trabajar con operaciones lineales y poder usar álgebra lineal para operar con todas las propiedades de las matrices.

Si tenemos una operación del estilo,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underline{\underline{A}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \underline{t} \qquad \textbf{(1.1)}$$

podemos encontrar una matriz **A**' y un vector **f** que cumplan,

$$a\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \left[\underline{\underline{A}}' \mid \underline{f}\right] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \textbf{(1.2)}$$

de esta manera obtenemos una operación lineal con la que podemos trabajar más cómodamente. Como ya veremos este cambio introduce una constante llamada constante de indeterminación.

Al extender el vector (x, y, z) lo que hacemos es transformarlo en coordenadas homogéneas, esto hace que el punto (u,v) quede expresado en el sistema afín como (u, v, w).

La proyección de un punto 3D "***X"*** sobre el plano de imagen queda definida por la *Matriz de Proyección* **P** de la siguiente manera,

$$a\begin{bmatrix} U_i \\ U_j \\ W \end{bmatrix} = \underline{\underline{P}} \cdot \begin{bmatrix} X_x \\ X_y \\ X_z \\ 1 \end{bmatrix} \qquad \textbf{(1.3)}$$

donde P, es una matriz 3x4, definida como:

$$P = \left[\underline{\underline{K}} \cdot \underline{\underline{R}} \mid -\underline{\underline{K}} \cdot \underline{\underline{R}} \cdot \underline{T}\right] \qquad \textbf{(1.4)}$$

K es la matriz intrínseca [3x3]
R es la matriz de rotación [3x3]
T es el vector de translación [3x1]

a es una constante de indeterminación, indica que hay infinitos puntos de una misma recta que pueden proyectarse sobre el mismo punto dentro del plano de imagen, esta constante aparece por el hecho de proyectar un punto 3D sobre un plano 2D.

$$K = \begin{bmatrix} a & g & u_0 \\ 0 & b & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(1.5)}$$

"**u0**"y "**v0**" son las coordenadas del punto principal, "**a**" y "**ß**" son las distancias focales respecto a los ejes "x" e "y" en píxeles y el parámetro "**?**" describe la oblicuidad entre los ejes de los dos sistemas de coordenadas.

Para entender la matriz de proyección, estudiaremos como se proyecta un punto 3D en el plano de imagen.

Un punto **X** es expresado en el sistema de referencia del mundo como un vector X_w [3x1], para expresar este mismo punto en el sistema de referencia de la cámara, X_c, debemos trasladarlo a partir del vector **T** y rotarlo según la matriz **R**.

$$X_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \underline{X}_w \cdot \left[\underline{\underline{R}} \mid -\underline{\underline{R}} \cdot \underline{T}\right] \qquad \textbf{(1.6)}$$

El punto X_c se proyecta en el plano de imagen como U_c, y puede ser calculado geométricamente, como se muestra en la siguiente figura.

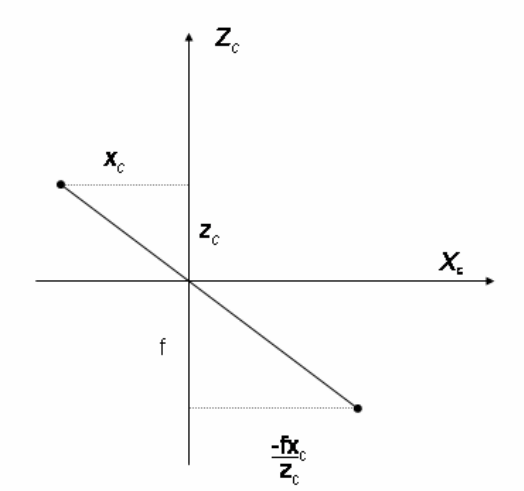


Fig. 1.2 Proyección de un punto 3D al plano de imagen

De aquí obtenemos:

$$U_c = \left[\frac{-fx_c}{z_c}, \frac{-fy_c}{z_c}, -f \right]^T \qquad \textbf{(1.7)}$$

El punto proyectado puede ser representado en el plano de imagen 2D en coordenadas afines $\hat{u} = [U, V, W]^T$.

Y este a su vez en coordenadas euclídeas $u = [u, v]^T = \left[\frac{U}{W} \quad \frac{V}{W} \right]^T$

El hecho de utilizar el sistema de coordenadas afín nos permite multiplicar el punto por una matriz [3x3] cuyas incógnitas **a**, **b**, **c** describen las distorsiones que sufre el punto debido a las características internas de la cámara. Esta matriz **K** es la definida anteriormente como matriz intrínseca y se encarga de representar el punto dentro del plano de imagen en unidades de píxel. Si multiplicamos en coordenadas afines por esta matriz, nos queda:

$$\tilde{u} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} a & b & -u_0 \\ 0 & c & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{-fx_c}{z_c} \\ \frac{-fy_c}{z_c} \\ 1 \\ \end{bmatrix} = \begin{bmatrix} -fa & -fb & -u_0 \\ 0 & -fc & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \\ \end{bmatrix} \quad \textbf{(1.8)}$$

Para aislar las variables X_c y Y_c, multiplicamos todo por Z_c,

$$z_c\tilde{u} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} -fa & -fb & -u_0 \\ 0 & -fc & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} -fa & -fb & -u_0 \\ 0 & -fc & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R(X_w - T) = \underline{\underline{K}} \cdot \underline{\underline{R}}(\underline{X}_w - \underline{T})$$

(1.9)

Si expresamos el punto de la escena en coordenadas homogéneas $\hat{X}_w = [X_w, 1]^T$ podemos escribir la proyección con una matriz 3x4 de la siguiente forma

$$\hat{u} = \begin{bmatrix} U \\ V \\ W \end{bmatrix} = [\underline{\underline{K}}\underline{\underline{R}} \mid -\underline{\underline{K}}\underline{\underline{R}}\underline{T}] \cdot \begin{bmatrix} X_w \\ 1 \end{bmatrix} = \underline{\underline{P}} \begin{bmatrix} X_w \\ 1 \end{bmatrix} = \underline{\underline{P}}\hat{X}_w \quad \textbf{(1.10)}$$

En este apartado se ha explicado el modelo geométrico de una cámara. En los siguientes se va a exponer la teoría fundamental sobre el análisis de una escena a partir de dos imágenes parecidas de la misma, la estereoscopía.

1.3. Dos cámaras, estereoscopia.

1.3.1. Geometría epipolar

Para conseguir simular el sistema visual humano, son necesarios dos sistemas de captación de imágenes como hemos visto anteriormente (par estéreo), por este motivo se trabaja con un par de cámaras separadas una pequeña distancia y con un punto de vista parecido, con la finalidad de procesar la información obtenida generando una imagen de profundidad de la escena. Dado que para nuestro trabajo ya disponemos de estas imágenes de profundidad no vamos a entrar a

explicar el procesado a partir del cual se generan estas imágenes, dicho proceso puede encontrarse en **[3]**.

La relación geométrica entre dos proyecciones de un mismo punto físico puede ser expresada a través de la *geometría epipolar*, en la figura 3 aparece un esquema de dicha geometría.

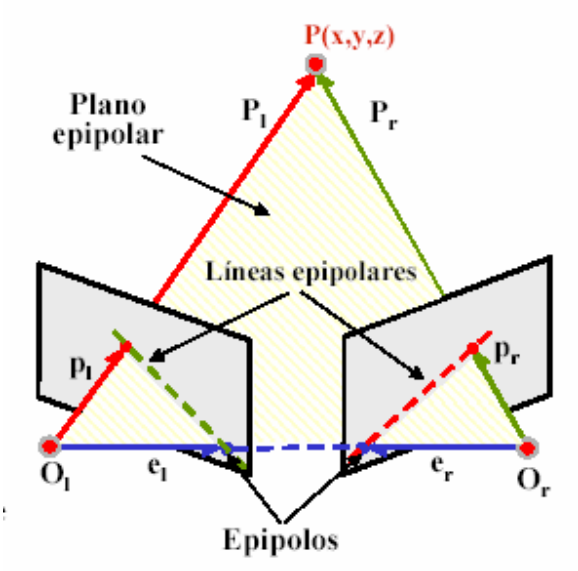


Fig. 1.3 Geometría epipolar

La línea que conecta los centros ópticos O_l y O_r es la llamada *línea base* (baseline) y el punto **P** es el punto de la escena observado por las dos cámaras. Al plano definido por el punto **P**, y los dos centros ópticos se le llama plano epipolar. Este plano intersecciona con los planos de imagen en las líneas epipolares, las cuales a su vez intersecan con la línea base en los llamados epipolos (proyección del centro óptico de una cámara en la otra).

Resumiendo, podemos decir que las líneas epipolares son las proyecciones en la imagen derecha de la línea que une el centro óptico de la cámara izquierda con el punto “p” de la escena y viceversa.

Toda esta geometría es fundamental para facilitar la búsqueda de correspondencias entre píxeles en las dos imágenes. Como se puede ver en la figura anterior, siendo p_l y p_r las proyecciones del punto “p” en las imágenes izquierda y derecha respectivamente. Ambos puntos quedan comprendidos en las líneas epipolares, por lo que la búsqueda de correspondencias queda reducida a un problema de 1D.

1.3.2. Rectificación

El proceso de rectificación esta basado en la geometría epipolar (explicada en el apartado anterior) del par estéreo. Básicamente rectificar significa transformar mediante software las imágenes obtenidas, conociendo los parámetros tanto intrínsecos como extrínsecos de las cámaras, de manera que las líneas epipolares de ambas imágenes queden alineadas horizontalmente. Con esto se consigue que los puntos p_l y p_r tengan la misma posición en el eje "y" en sus respectivas imágenes. En la siguiente figura se ve claramente el objetivo de la rectificación.

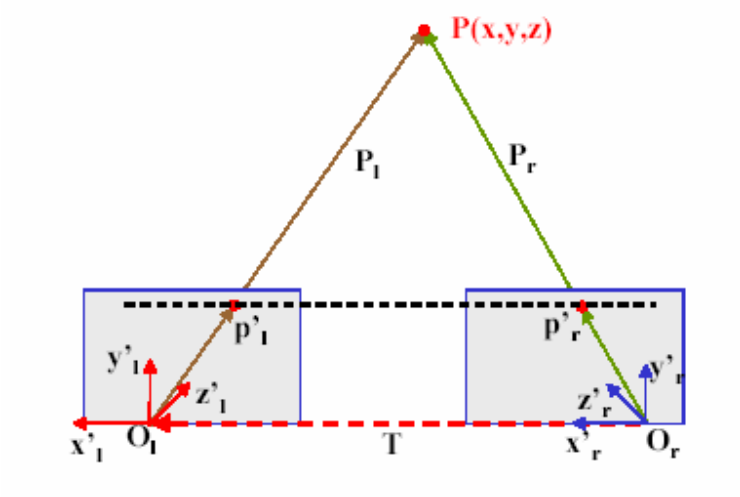


Fig. 1.4 Planos de imagen alineados

Para conseguir rectificar, es necesario alinear los ejes ópticos de manera que los planos de imagen transformados queden paralelos entre si. Hay 3 pasos a seguir:

1. Rotar la cámara derecha e izquierda, hasta que sus ejes "x" coincidan: $O_r - O_l = T$
2. Definir la matriz de rotación R_{rect} para la nueva posición.
3. Por último calcular las nuevas matrices de proyección y reproyectar los puntos 3D sobre los nuevos planos de imagen.

En la siguiente imagen se puede ver un esquema de la transformación que debemos realizar.

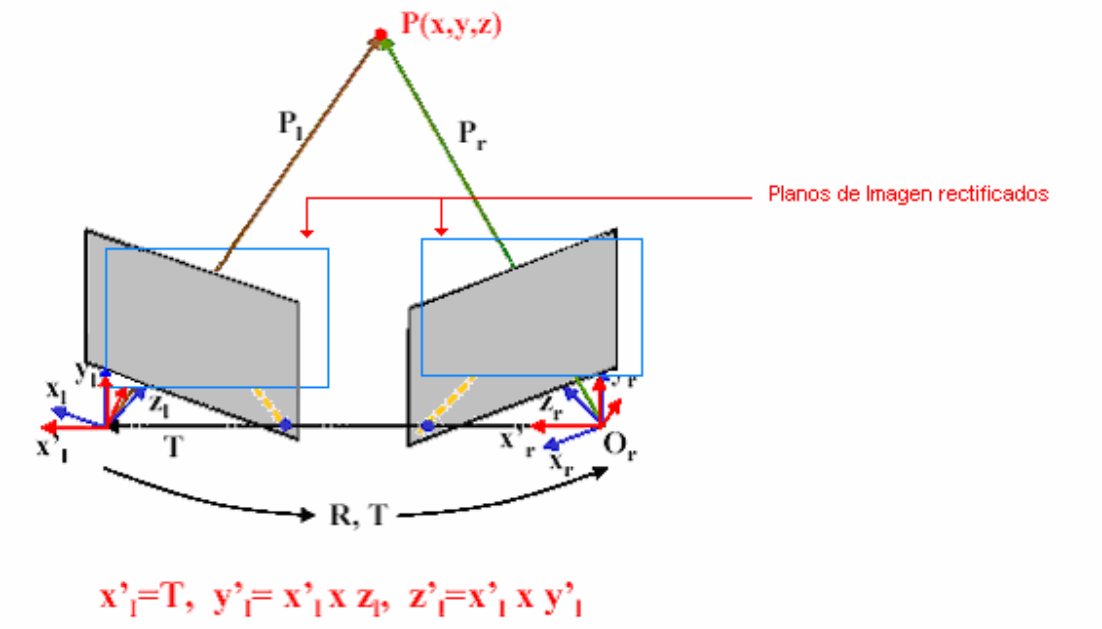


Fig. 1.5 Proceso de rectificación

El desarrollo matemático a partir del cual se calculan las llamadas matrices de rectificación esta descrito en el apartado 2.3.

1.3.3. Cálculo de la disparidad o profundidad

La teoría fundamental sobre el cálculo de profundidades es sencilla y se reduce a un problema geométrico. Para ello nos basaremos en imágenes rectificadas donde las líneas epipolares están alineadas con lo que la búsqueda de correspondencias entre píxeles es más sencilla.

En la siguiente figura se muestra dos planos de imagen, de dos cámaras separadas una distancia *2h,* con sus ejes ópticos paralelos entre si. El punto **P** (x, y, z) queda proyectado en los 2 planos de imagen como P_i y P_d respectivamente.

En la figura, el eje de coordenadas *z* representa la distancia desde las cámaras y el eje *x* representa la distancia horizontal. X=0 es el punto medio entre cámaras, y cada cámara posee un sistema de coordenadas local (x_i para la izquierda y x_d para la derecha).

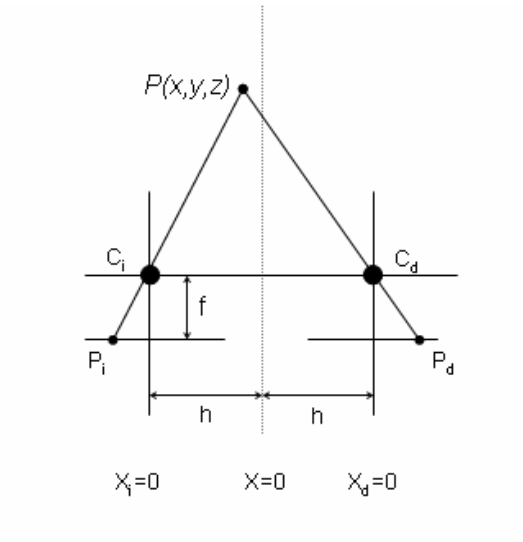


Fig. 1.6 Cálculo de la disparidad

Queda claro que hay una diferencia (disparidad) entre x_d y x_i, debido a la diferente posición de las cámaras, con esta geometría elemental podemos deducir la coordenada ***z*** del punto P partiendo de las siguientes ecuaciones:

$$\frac{P_i}{f} = -\frac{h+x}{z} \qquad \textbf{(1.11)}$$

y similar para el otro lado,

$$\frac{P_d}{f} = \frac{h+x}{z} \qquad \textbf{(1.12)}$$

Si eliminamos la *x* de estas dos ecuaciones nos queda,

$$z(P_d - P_i) = 2hf \qquad \textbf{(1.13)}$$

y de aquí obtenemos z como,

$$z = \frac{2hf}{P_d - P_i} \qquad \textbf{(1.14)}$$

CAPÍTULO 2. DESARROLLO DE LOS ALGORÍTMOS

2.1. Estudio general del problema

Como se ha dicho anteriormente trabajaremos con 8 videos de una escena proporcionados por Microsoft, capturados por 8 cámaras distribuidas en un arco de 30 grados. Cada video esta formado por 100 imágenes de color de alta resolución (1024x768) y sus correspondientes 100 imágenes de profundidad. Las cámaras con las que se tomaron estos videos fueron calibradas previamente. De esta calibración disponemos de todos los parámetros, es decir, los parámetros intrínsecos e extrínsecos de cada cámara, así como los orígenes de los tres sistemas de referencia.

La idea de este proyecto es interpolar todas las vistas que hay entre las 8 cámaras y conseguir una reproducción continúa del video en la cual nos podamos desplazar horizontalmente a través del arco formado por las 8 cámaras estáticas. Dicho de otra manera, queremos convertir estas 8 cámaras estáticas en 1 sola cámara, a la que llamaremos cámara “virtual”, con la que podremos desplazarnos de forma continua por todo el arco, para ello lo que haremos será interpolar todas las vistas que verían las cámaras intermedias partiendo de la información de las dos cámaras adyacentes derecha e izquierda.

Existen muchos métodos para tratar el problema, nosotros estudiaremos dos de ellos, con la finalidad de hacer una comparativa y valorar cual de los dos métodos sería más factible utilizar en posibles aplicaciones comerciales futuras.

El problema principal de los algoritmos de interpolación y que todavía no se ha conseguido solucionar de forma definitiva es el de la oclusión de objetos, es decir, objetos que desde una posición son vistos por la cámara pero que desde otro punto de vista son tapados por otros objetos más cercanos a la cámara. Nosotros trataremos de solucionar el problema trabajando con la información de las cámaras existentes a la derecha e izquierda de nuestro nuevo punto de vista.

En la siguiente figura se puede ver como queda distribuido el campo de visión, un objeto puede ser visible para una cámara pero no para la otra por dos motivos. Primero, las dos cámaras poseen campos de visión diferente, debido a su desplazamiento en el eje “x”. Segundo, la oclusión de objetos depende del punto de vista de las cámaras. En la figura 3.1 podemos observar como las áreas I y II son solo visibles por la cámara izquierda, las IV y V son visibles por la cámara derecha y la III es visible por ambas cámaras.

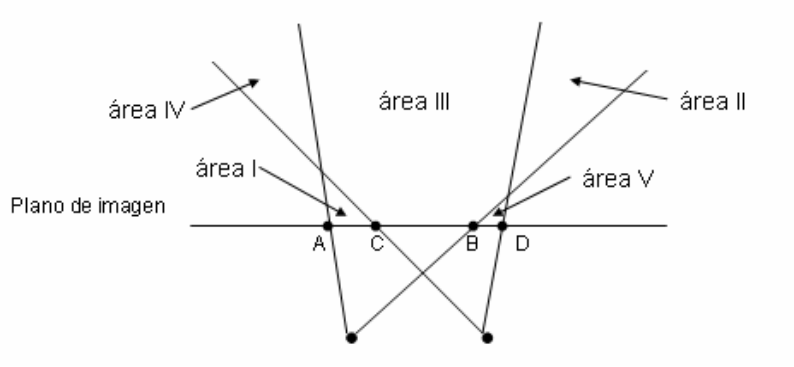


Fig. 2.1 Áreas de visión del par de cámaras

Pese a tener la información de las cámaras derecha e izquierda, habrán zonas de la nueva imagen de la que no dispondremos de información. Un objeto muy cercano a la cámara sufrirá un desplazamiento muy grande. Este desplazamiento puede hacer que aparezcan en la nueva vista objetos más lejanos no visibles para ninguna de las dos cámaras.

Tratar de solucionar este problema sería hablar de reconocimiento de objetos, y esto queda muy lejos de los objetivos de este proyecto, lo que haremos en estas zonas será rellenarlas a partir de la información de píxeles situados dentro del mismo contorno, para no observar espacios negros en la imagen.

Para entender el problema de la oclusión de objetos, debemos entender la idea principal de la visión estereoscópica, si desplazamos una cámara horizontalmente los objetos que sufren más movimiento son los objetos más cercanos a esta, esto es un simple problema geométrico, que se puede entender con la siguiente figura.

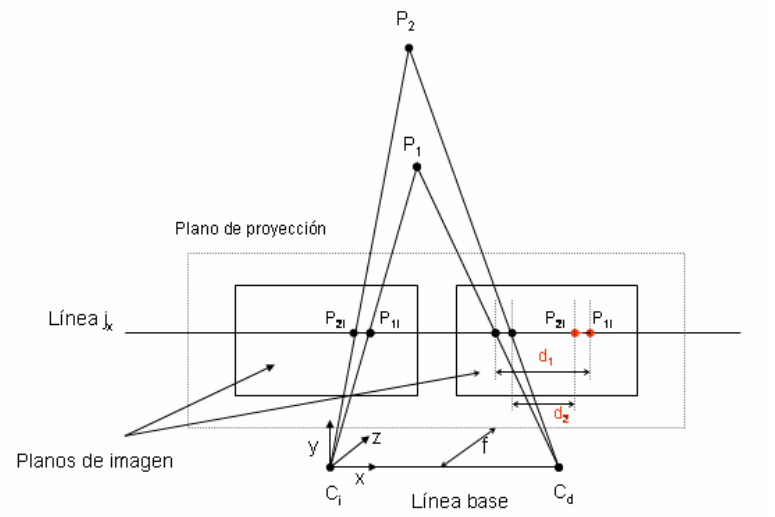


Fig. 2.2 Proyección de dos puntos de profundidades diferentes

En la figura anterior se ven las proyecciones de dos puntos con profundidades diferentes sobre dos planos de imagen separados una cierta distancia horizontal, el punto **p1** sufre un mayor desplazamiento que el punto **p2** (d1>d2), debido a la menor profundidad que posee. Por lo tanto es posible que existan puntos menos profundos que durante el rendering ocluyan a otros más lejanos y de igual manera, puntos lejanos que antes eran ocluidos es posible que una vez realizado el rendering sean visibles.

Como se ha dicho anteriormente, nuestro fin es hacer una comparativa entre dos métodos de rendering, uno será la reproyección directa y otro la reproyección a partir de imágenes rectificadas. En los siguientes puntos se explicarán detalladamente los algoritmos desarrollados para ello.

2.2. Análisis de los parámetros iniciales

Antes de empezar a trabajar con las imágenes es necesario hacer un análisis previo para determinar y entender cuales son los parámetros con los que partimos, es decir, saber que es y que significa la información ofrecida por Microsoft, la cuál será necesaria para desarrollar nuestro trabajo. Debe tenerse en cuenta que aunque desde un punto de vista analítico la formulación matricial del problema no varia es importante conocer como se han definido los ejes de rotación, en que orden y en que sentido para poder interpretar el problema de forma correcta.

Microsoft nos da una matriz [3x3] que es la matriz intrínseca, y una matriz [3x4] que le llama matriz de rotación **R'**.

Si sabemos que la matriz de proyección es $\underline{\underline{P}} = \underline{\underline{K}}\left[\underline{\underline{R}} \mid -\underline{\underline{R}} * \underline{T}\right]$, deducimos que la matriz de rotación que define Microsoft es,

$$\underline{\underline{R}}' = \left[\underline{\underline{R}} \mid -\underline{\underline{R}} * \underline{T}\right] \qquad \textbf{(2.1)}$$

de aquí obtenemos el vector de translación como $\underline{T} = -\left(\underline{\underline{R}}' \cdot \underline{\underline{R}}^{-1}\right)$.

También es importante conocer los orígenes de los sistemas de referencia, sabemos que para el sistema de coordenadas del mundo real se ha escogido el centro de la cámara número 4 como origen de coordenadas y que para el sistema de referencia de la imagen se ha escogido como origen el punto inferior izquierdo, es decir, la esquina inferior izquierda. Durante el proyecto se trabajará con una librería de procesado de imagen de Visual C++ llamada **IPL**, en esta se trata al punto superior izquierdo como origen de la imagen, por lo que será necesario transformar cada punto que vayamos a procesar al sistema de coordenadas de

Microsoft y una vez finalizado el procesado hay que realizar el proceso inverso, ya que sino los cálculos con los parámetros que tenemos serán erróneos.

Esto último es muy importante ya que durante el proceso de calibración de las cámaras se ha seguido este criterio por lo que todos los parámetros que tenemos están referenciados a ese punto.

Una vez tenemos una visión global de todos los parámetros, hay que averiguar como esta definida la matriz de rotación **R**, la matriz de rotación define la transformación que ha de experimentar un punto para pasar del sistema de referencia del mundo al sistema de referencia de la cámara, es decir, esta definida por tres rotaciones básicas respecto cada eje del plano **x**, **y**, **z.** En la siguiente imagen se muestra como queda definida la rotación 2D de un punto respecto al eje “z”, esta transformación 2D es suficiente para entender la naturaleza básica del proyecto, ya que la transformación 3D esta compuesta por 3 rotaciones 2D para cada eje del plano, por lo que la matriz de rotación 3D queda, $\underline{\underline{R}}_{x,y,z} = \underline{\underline{R}}_{x} \cdot \underline{\underline{R}}_{y} \cdot \underline{\underline{R}}_{z}$

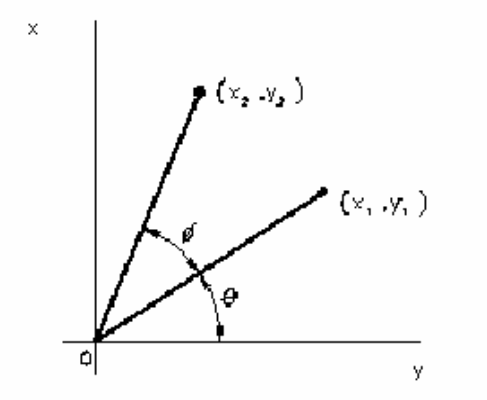


Fig. 2.3 Rotación 2D de un punto

De aquí sabemos que,

$$sen J = \frac{y_1}{L} \quad Cos J = \frac{x_1}{L} \tag{2.2}$$

con esto obtenemos que la matriz 2D R_z queda definida como:

$$R = \begin{bmatrix} \cos f & sen f \\ -sen f & \cos f \end{bmatrix} \tag{2.3}$$

El problema que aparece en la matriz de rotación es que en función de hacia donde se haya definido el giro (si horario o anti-horario) tendremos signo positivo o negativo. Si a esto le sumamos que la matriz de rotación final es la combinación de estas tres matrices, se hace imposible averiguar como han sido definidos los giros de la rotación en función de los valores que tenemos.

Para averiguarlo implementaremos una herramienta software que nos calcula, a partir de un punto marcado en la imagen izquierda, la línea epipolar de ese punto en la imagen derecha. La finalidad de esta aplicación es poder ir variando los parámetros extrínsecos hasta obtener la línea epipolar correcta, en este momento quedarán definidos correctamente todos los parámetros.

Para implementar esta aplicación recurrimos de nuevo a la geometría epipolar. Anteriormente se ha explicado que las líneas epipolares derechas son las proyecciones sobre el plano de imagen derecho de las líneas que unen el centro de la cámara izquierda con los puntos del mundo real representados en su imagen y lo mismo para las líneas epipolares izquierdas. Lo que haremos será marcar un punto en la imagen izquierda y representar sobre la imagen derecha la recta que une este punto con el centro de la cámara izquierda.

En la teoría se ha hablado de una constante de indeterminación llamada alpha, esta nos ajusta la posición de un punto de la imagen sobre su línea de proyección en el mundo 3D, es decir, nos ajusta la profundidad del punto.

Si nos fijamos en la figura (1.3), vemos como la línea que queremos proyectar es la de color rojo, eligiendo dos valores diferentes de la constante de indeterminación lo que hacemos es coger dos puntos de esta línea, que a su vez proyectaremos sobre el plano de imagen de la cámara derecha obteniendo así la línea epipolar.

En la siguiente imagen se pueden ver los resultados de la aplicación.

Fig. 2.5 Ejemplo del cálculo de las líneas epipolares

Si observamos los resultados vemos como la línea epipolar calculada pasa en la imagen derecha por el punto marcado en la imagen izquierda con una precisión inferior al píxel, por lo que podemos decir que los resultados son muy buenos.

En relación con los parámetros ofrecidos por Microsoft, es importante decir, que tenemos la fórmula que relaciona el nivel de gris de un píxel en los mapas de disparidad con la profundidad en el mundo 3D.

$$z(r,c) = 1.0/((P(r,c)/255.0)\times(1.0/MinZ - 1.0/MaxZ) + 1.0/MaxZ) \quad \textbf{(2.4)}$$

donde z(r,c) es el valor de la profundidad (en coordenadas x,y,z, el centro óptico de la quinta cámara es el origen del sistema de referencia del mundo.) P(r,c) es el valor de la intensidad del píxel, MinZ es 42.0 y MaxZ es 130.0.

Una vez explicados los conocimientos básicos sobre visión estereoscópica vamos a estudiar los métodos y algoritmos necesarios para conseguir nuestro objetivo, la interpolación de vistas.

2.3. Rectificación del par estéreo

Como se ha explicado anteriormente rectificar consiste en transformar las imágenes originales del par estéreo de manera que sus líneas epipolares queden alineadas horizontalmente. Para ello implementaremos el algoritmo descrito en **[4]** y cuya idea básica se describe en la figura (1.5).

Partiremos de dos imágenes originales obtenidas de dos cámaras adyacentes de las cuales conocemos todos sus parámetros extrínsecos e intrínsecos. El algoritmo calcula las matrices de rectificación N_i y N_d, a partir de las cuales transformaremos los puntos de la imagen original.

Primero calculamos la matriz de proyección antigua para cada cámara, es la que relaciona los puntos del plano de imagen original con los puntos de la escena 3D.

$$\begin{aligned} \underline{\underline{P}}_{i\_ant} &= \underline{\underline{K}}_i(\underline{\underline{R}}_i \mid -\underline{\underline{R}}_i \underline{T}_i) \\ \underline{\underline{P}}_{d\_ant} &= \underline{\underline{K}}_d(\underline{\underline{R}}_d \mid -\underline{\underline{R}}_d \underline{T}_d) \end{aligned} \quad \textbf{(2.5)}$$

Una vez definidas las matrices de proyección antiguas pasaremos a calcular las matrices de proyección nuevas, que relacionaran los puntos de la escena 3D con el nuevo plano de imagen rectificado, estas matrices solo diferirán una de la otra en los centros ópticos (durante la rectificación, los centros ópticos de cada cámara

permanecerán invariables). Para esto es necesario construir una matriz de rotación común a los dos nuevos planos de imagen basándonos en tres reglas fundamentales descritas en **[6]**:

1. El nuevo eje “x” ha de ser paralelo a la línea base (baseline):
$$r_1 = (T_r - T_l) / \left\| T_r - T_l \right\|$$

2. El nuevo eje “y” ha de ser ortogonal a “x” y a **k**:
$$r_2 = (r_1 \otimes k)$$

3. El nuevo eje “z” ha de ser ortogonal a “x” y a “y”:
$$r_3 = (r_1 \otimes r_2)$$

En 2, **k** es un vector unitario que fija la posición del nuevo eje “y” en el plano ortogonal a “x”. Cogemos k como el vector “z” unitario de la matriz de proyección antigua izquierda, para así hacer que el nuevo eje “y” sea ortogonal a “x” y a “z” izquierda.

De estas tres reglas obtenemos la nueva matriz de rotación como,

$$R = \begin{bmatrix} r_1^T \\ r_2^T \\ r_3^T \end{bmatrix} \qquad \textbf{(2.6)}$$

La matriz intrínseca nueva ***K*** será una media de las dos matrices intrínsecas antiguas, en la que impondremos que el parámetro de oblicuidad entre ejes sea 0. Por lo tanto las nuevas matrices de proyección nos quedaran:

$$\begin{aligned} \underline{\underline{P}}_{i\_nueva} &= \underline{\underline{K}}(\underline{\underline{R}} \mid -\underline{\underline{R}}\underline{T}_i) \\ \underline{\underline{P}}_{d\_nueva} &= \underline{\underline{K}}(\underline{\underline{R}} \mid -\underline{\underline{R}}\underline{T}_d) \end{aligned} \qquad \textbf{(2.8)}$$

Para cualquier punto 3D **w,** podemos escribir

$$\begin{cases} \hat{u}_{ant} \cong P_{ant}\hat{w} \\ \hat{u}_{nueva} \cong P_{nueva}\hat{w} \end{cases} \qquad \textbf{(2.9)}$$

A partir de la ecuación de proyección sabemos que

$$\begin{cases} w = T + l_{ant}(KR)_{ant}^{-1}\hat{u}_{ant} \\ w = T + l_{nueva}(KR)_{nueva}^{-1}\hat{u}_{nueva} \end{cases} \quad \textbf{(2.10)}$$

De aquí igualamos y obtenemos

$$\hat{u}_{nueva} = l(KR)_{ant}^{-1}(KR)_{nueva}\hat{u}_{ant} \quad \textbf{(2.11)}$$

Por lo que las matrices de rectificación obtenidas son

$$N_i = (KR)\cdot(K_i R_i)^{-1}, \text{ para la imagen izquierda.}$$
$$N_d = (KR)\cdot(K_d R_d)^{-1}, \text{ para la imagen derecha.} \quad \textbf{(2.12)}$$

Para rectificar las imágenes multiplicaremos cada punto de la imagen por su matriz de rectificación correspondiente, en general el punto rectificado no corresponderá a una posición entera de la imagen rectificada por lo que ajustarlo a una posición entera, este ajuste de posición y el hecho de cambiar el plano de imagen, provocaran que queden píxeles en los que no haya información, estos espacios vacíos serán eliminados mediante un filtrado.

Debido a que estamos cambiando la posición del plano de imagen, es posible que hayan puntos que se nos salgan de la imagen, por este motivo deberemos adaptar el tamaño de las imágenes rectificadas. Nuestro algoritmo calcula cual será la posición en la imagen rectificada de los 4 puntos extremos de cada imagen original, es decir obtiene 8 puntos rectificados, a partir de los cuales calcula el tamaño que han de tener las nuevas imágenes.

Las siguientes imágenes muestran un par estéreo original y su correspondiente rectificación, si nos fijamos en las imágenes originales, la línea roja que recorre ambas imágenes horizontalmente por la misma posición, no pasa por los mismos puntos, esto lo corregiremos gracias a la rectificación.

Fig. 2.6 Ejemplo de rectificado sin filtrado.

En las imágenes anteriores se puede apreciar los dos fenómenos que mencionábamos anteriormente, uno es el tema del tamaño de las imágenes, y otro son las líneas negras que aparecen en la imagen rectificada.

Para solucionar el tema de los píxeles vacíos hemos optado por un filtrado sencillo en el que se recorre la imagen interpolando los espacios vacíos con la información de color de los píxeles vecinos. Si nos fijamos aparecen dos tipos de líneas, unas verticales y otras horizontales, las líneas verticales se pueden filtrar durante el proceso de rectificación, en cambio las líneas horizontales las filtraremos mediante un recorrido vertical de la imagen. El hecho de recorrer la imagen verticalmente provoca que aumente considerablemente el tiempo de procesado, para procesar las imágenes utilizamos las estructuras **IplImage** de OpenCv, estas almacenan

los píxeles por filas en un array de una sola dimensión, por esto es más rápido recorrer las imágenes por filas que por columnas.

Dado que el objetivo del proyecto es una comparativa entre dos métodos de rendering, el tema del tiempo no es crítico, por lo que no se ha entrado en optimización de todos los algoritmos.

En las siguientes imágenes se puede ver el resultado de filtrar las imágenes anteriores, y como la precisión conseguida por la rectificación es del orden de un píxel.

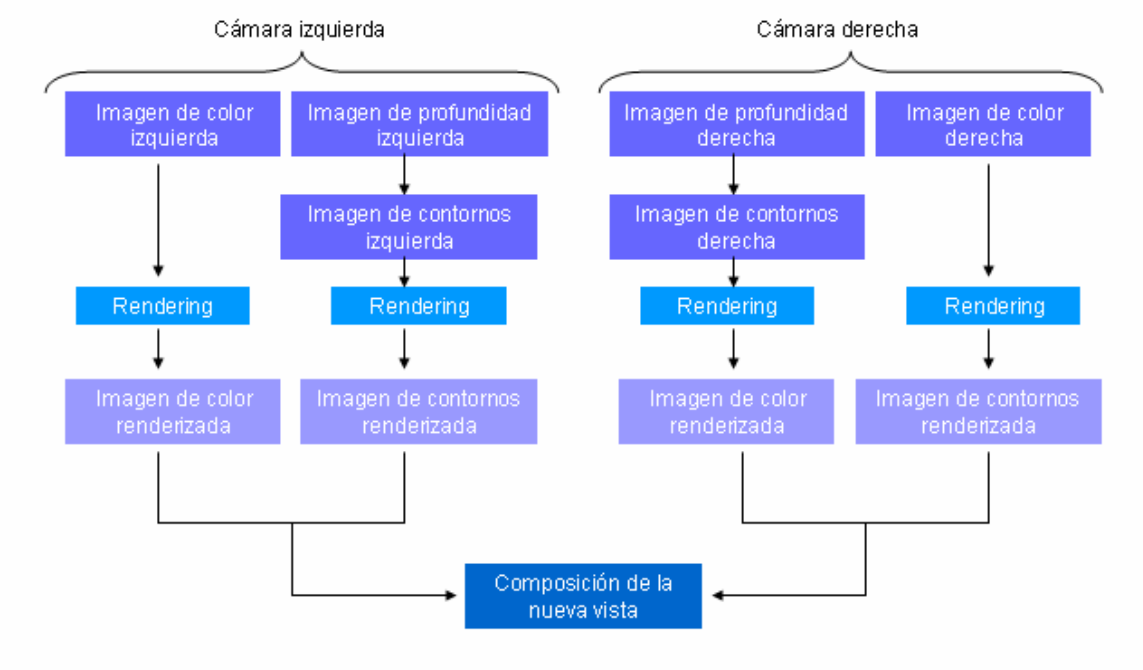
Fig. 2.7 Imágenes rectificadas y filtradas

En la figura (2.7) se pude ver como si trazamos dos líneas horizontales en la misma posición sobre ambas imágenes, las dos líneas contienen los mismos puntos, siempre y cuando estos sean visibles por las dos cámaras.

2.4. Método de la Reproyección directa

Partiendo de las imágenes de profundidad y color de las cámaras izquierda y derecha, la reproyección directa consiste en encontrar la nueva posición de cada píxel dentro del nuevo plano de imagen de la cámara "virtual" situada entre las dos cámaras.

El diagrama de bloques de este método queda definido de la siguiente manera:



En el estudio teórico se ha definido la matriz de proyección, esta relaciona los puntos del plano de imagen con los puntos 3D de la escena real a partir de los parámetros intrínsecos, extrínsecos y la posición del punto. La idea de este método es, a partir de la matriz de proyección de cada cámara y la profundidad de cada punto hacer el proceso inverso de la proyección, es decir, proyectar los puntos del plano de imagen en el mundo real.

La ecuación que relaciona el mundo 3D con el plano de imagen sabemos que tiene la siguiente forma

$$a\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underline{\underline{k}}\left[\underline{\underline{R}} \mid -\underline{\underline{R}}\underline{T}\right]\cdot\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \textbf{(2.13)}$$

Si despejamos el punto 3D nos queda

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = a\,(\underline{\underline{K}} \cdot \underline{\underline{R}})^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} + \underline{T} = a \cdot \underline{p} + \underline{T} \qquad \textbf{(2.14)}$$

Por lo tanto para encontrar el punto 3D es necesario calcular la constante de indeterminación a , como sabemos la profundidad del punto, buscaremos que a cumple la ecuación

$$z = a \cdot p(3) + T(3) \qquad \textbf{(2.15)}$$

Una vez tenemos un punto en el mundo 3D podemos crearnos la matriz de proyección de la cámara virtual y proyectar el punto sobre esta nueva vista, las nuevas matrices de rotación e intrínseca de la cámara virtual las obtendremos a partir de una ponderación de las de las cámaras derecha e izquierda en función de la posición, con esto evitaremos posibles discontinuidades a la hora de reproducir el video.

Como hemos visto anteriormente el tema de la oclusión es uno de los más importantes durante el rendering, es difícil controlar cuando un píxel es visible en la nueva imagen y cuando no. Lo que haremos será estudiar el desplazamiento de los objetos en función del movimiento y gracias a la información de profundidad recorrer la imagen de forma que los píxeles más cercanos siempre tengan prioridad respecto a los píxeles más lejanos.

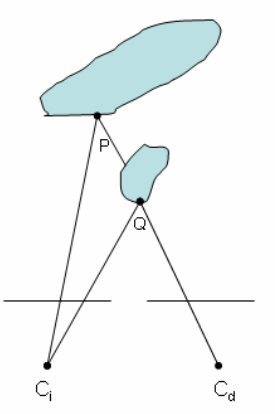


Fig. 2.8 Ejemplo de oclusión

Si estudiamos el movimiento de los objetos en función del desplazamiento vemos que al desplazar la cámara hacia la derecha los objetos en la imagen se mueven hacia la izquierda, de la misma manera si la cámara se desplaza hacia la izquierda los objetos se mueven hacia la derecha, lo que haremos será recorrer la imagen hacia el sentido del desplazamiento, es decir, si los objetos de desplazan hacia la izquierda, recorremos la imagen de derecha a izquierda y viceversa. Con esto se consigue dar prioridad a los objetos más cercanos, ya que en el caso de que hayan dos píxeles visibles en la imagen original que se proyecten en la misma posición en la nueva imagen, siempre se procesará primero el que sufre mayor desplazamiento, es decir, el más cercano a la cámara. En la siguiente figura se puede entender esta situación, tenemos las proyecciones de dos puntos para dos posiciones diferentes de la cámara.

En la proyección izquierda se ve como los dos puntos son visibles por la cámara y en la derecha se puede ver como estos dos puntos quedan proyectados en la misma posición, por lo tanto según lo explicado anteriormente, si recorremos la imagen izquierda en el sentido del movimiento, es decir, de derecha a izquierda, vemos como el primer punto a proyectar es el punto **Q** y por lo tanto le daremos preferencia frente al punto **P.**

Uno de los problemas que presenta este método es que al no tener las imágenes rectificadas se hace difícil tener el control de las posiciones de los píxeles reproyectados sobre el eje “y”. Durante el rendering los píxeles de una línea j_x no se proyectarán sobre la misma línea de la nueva imagen. Si recorremos horizontalmente la imagen, los puntos reproyectados que varíen su posición en “y” pueden ocupar zonas en las que objetos más cercanos que no han sido reproyectados todavía tienen preferencia. En la siguiente imagen puede verse este problema.

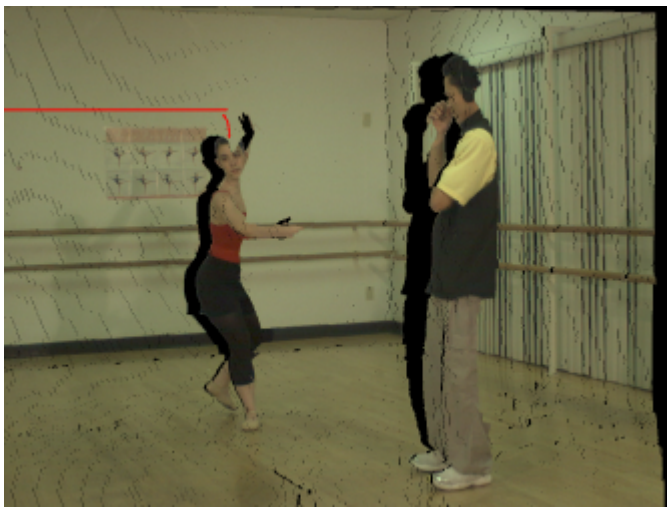


Fig. 2.9 Ejemplo de rendering horizontal

Los píxeles de la línea roja correspondientes al fondo, se reproyectan hacia abajo y ocupan espacios de la cabeza de la chica por haber sido reproyectados primero. Esto lo podemos solucionar recorriendo las imágenes verticalmente, el problema, como ya se ha comentado en la rectificación, es encontrar un compromiso adecuado entre la calidad del resultado final y el tiempo de procesado

Las siguientes dos figuras son un ejemplo de reproyección directa.

Fig. 2.10 Resultado del rendering con reproyección directa.

Las imágenes de profundidad utilizadas para el ejemplo anterior se muestran a continuación.

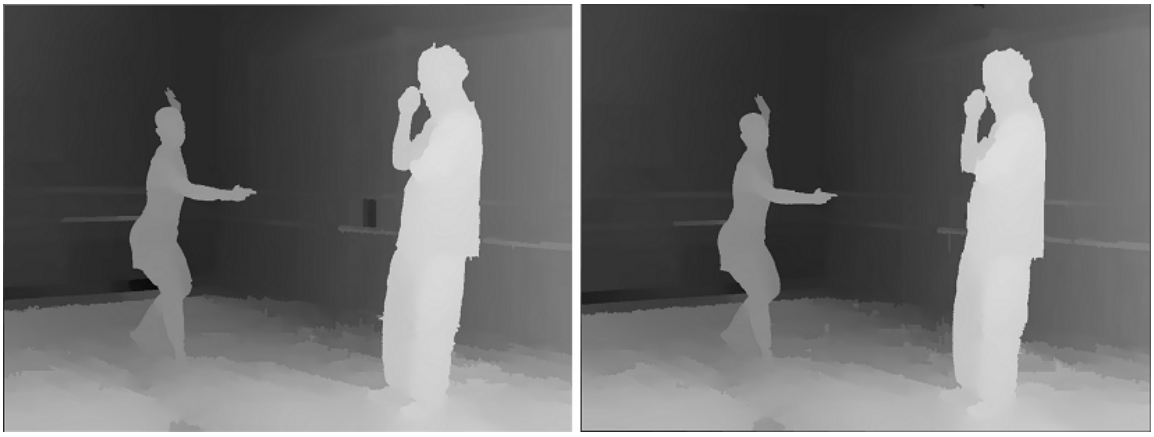


Fig. 2.11 Ejemplo de imágenes de profundidad usadas durante el rendering

En las imágenes reproyectadas aparecen un gran número de grietas vacías, esto es debido a las discontinuidades e imperfecciones de los mapas de disparidad con los que trabajamos. En el suelo es el lugar donde más ocurre este fenómeno, y si nos fijamos en las imágenes de profundidad es la zona más discontinua de la imagen.

Como hemos dicho anteriormente, con este método no resulta fácil tener un control sobre la nueva posición de los píxeles reproyectados, por lo que se hace complejo filtrar las grietas mientras se realiza la reproyección. El filtrado se deberá hacer a posteriori mientras se reconstruye la imagen final.

Una vez han sido reproyectadas las dos imágenes a la posición deseada, se creará la imagen final a partir de la información de ambas y un posterior filtrado, para ello será necesario la información de contornos de la nueva imagen. Gracias a la información de los contornos podremos rellenar zonas en las que no tenemos información. Buscaremos píxeles situados dentro del mismo contorno en el que se encuentra situada la zona oculta y asignaremos el valor de esos píxeles a los píxeles de los que no tenemos información.

Esta imagen de contornos será obtenida a partir de las reproyecciones izquierda y derecha de los contornos de los mapas de disparidad originales. Las imágenes de contornos de los mapas de profundidad serán calculadas mediante un operador de Canny. El operador de Canny **[11]** es un procedimiento muy conocido para determinar los contornos de la imagen y pueden encontrarse librerías de tratamiento de imagen que lo implementan de forma eficiente. Las pruebas realizadas nos indican que este operador proporciona, en general, buenos resultados para nuestra aplicación. No obstante, el principal inconveniente de esta técnica es que los contornos que representa son muy finos y a la hora de reproyectar no dan buenos resultados debido a la inexactitud de los mapas de disparidad, es decir, puede haber zonas de los contornos que no sean reproyectadas al no coincidir con el objeto en los mapas de disparidad.

Para solucionar esto lo que haremos es una dilatación de las imágenes obtenidas por la función canny, de esta manera ensancharemos los contornos y nos aseguraremos de que se reproyecte todo el objeto. En las siguientes figuras se puede ver un ejemplo de esto.

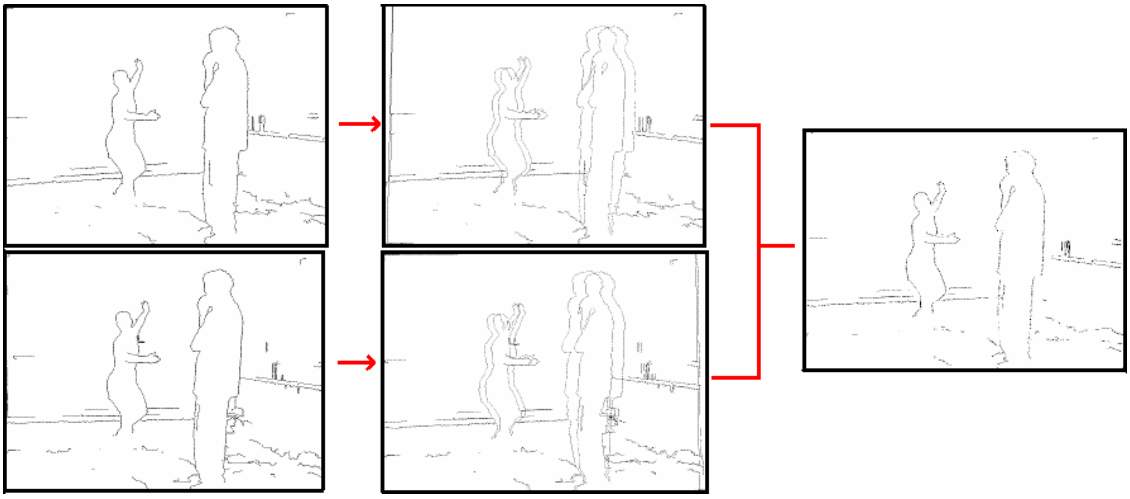


Fig. 2.12 Resultado de la reproyección de contornos sin dilatación

En la primera columna de la figura tenemos los contornos de los mapas de disparidad izquierdo y derecho respectivamente. En la segunda la reproyección de cada uno y en la última la fusión de las dos retroproyecciones.

En la imagen de contornos retroproyectada, se puede ver como las siluetas de los objetos no están completas. El problema de esto es que no podremos diferenciar correctamente los objetos a la hora de rellenar zonas de las que no disponemos de información. Lo que hará nuestro algoritmo es buscar, dentro del contorno en el donde esta la zona sin información, un píxel vecino que pertenezca al mismo objeto y del que si tengamos la información de textura, dotando a la zona oculta de la textura existente a su alrededor.

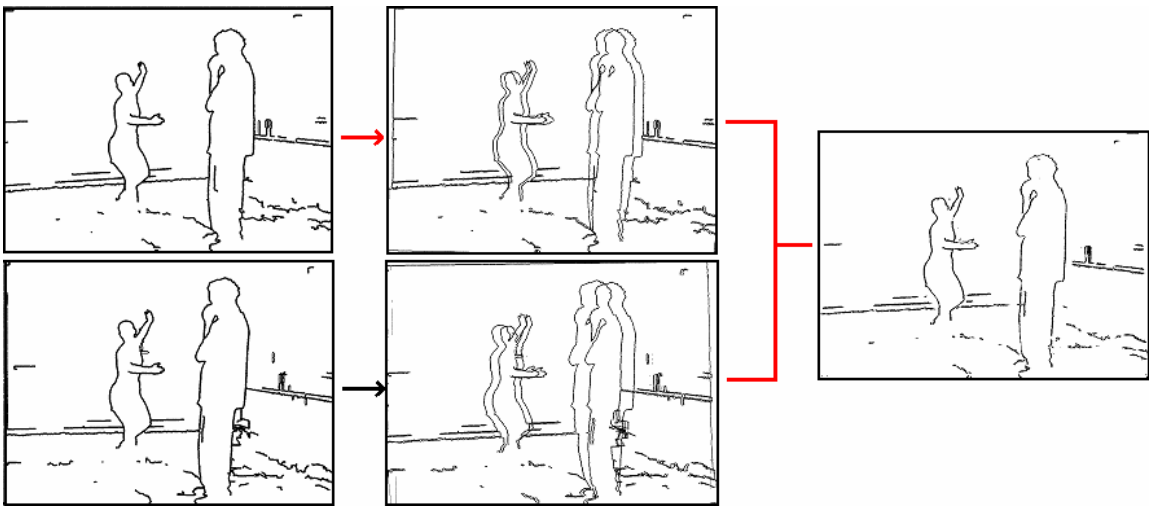


Fig. 2.13 Resultado de la reproyección de contornos con dilatación

Con la dilatación de las imágenes de contornos originales, conseguimos que la imagen de contornos reproyectada sea de buena calidad, como se muestra en la figura (2.13) por lo que es posible diferenciar texturas entre objetos distintos.

Los resultados finales obtenidos con este método se pueden ver en la siguiente imagen.

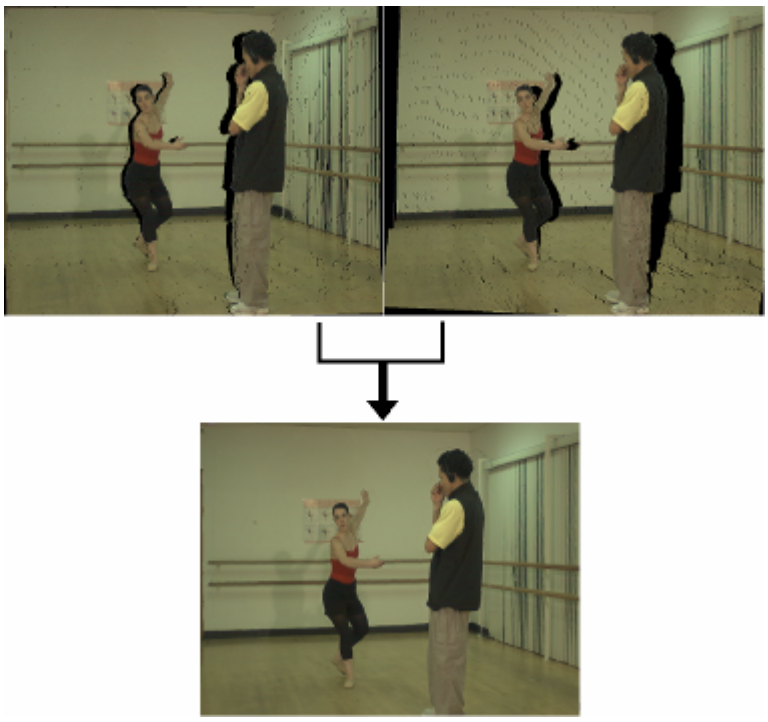


Fig. 2.14 Ejemplo de composición de la nueva vista

Existen algunos problemas al obtener la imagen final, básicamente son tres:

1. En la nueva imagen se puede llegar a ver la silueta de los objetos en sus antiguas posiciones, esto es debido a la inexactitud de los mapas de disparidad.

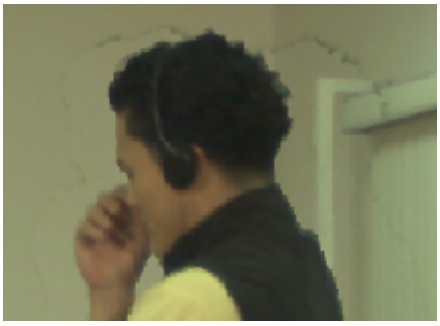


Fig. 2.15 Zoom de la nueva imagen

2. Hay zonas para las que hay información en las dos imágenes, pero una de ellas es errónea, nuestro algoritmo no tiene capacidad de decisión ante estos casos.

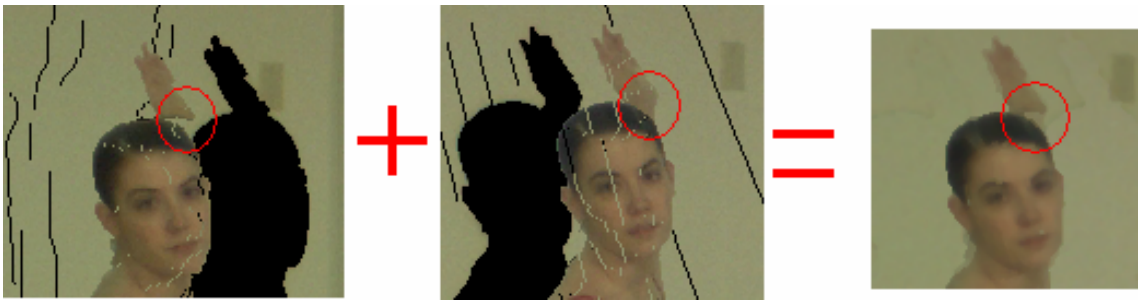


Fig. 2.16 Ejemplo de error en la composición de la nueva vista

En la anterior figura se muestra como para la imagen interpolada a partir de la cámara de la izquierda, la información de la mano es errónea, nuestro algoritmo hace una media de la información de ambas imágenes. Evidentemente, el problema podría solucionarse utilizando para esta región sólo la información que se obtiene a partir de la cámara de la derecha. No obstante, determinar cuál de las dos cámaras está proporcionando una información correcta no es trivial y a menudo requiere utilizar algoritmos de identificación de objetos que ralentizan excesivamente el proceso de interpolación. En nuestro caso, hemos utilizado un promedio entre los resultados obtenidos con cada una de las dos cámaras.

3. Por último, existe el problema de las zonas sin información. Estas zonas no son visibles por ninguna de las dos cámaras, lo que hace nuestro algoritmo es buscar los contornos de la nueva imagen, a partir de reproyectar las imágenes de contornos derecha e izquierda. Con esto, busca el píxel vecino más cercano dentro del contorno y lo asigna al píxel vacío. El problema de esto radica en la misma idea del punto 2. Para las imágenes de contornos retroproyectadas existen zonas que serán visibles por una única cámara. La incapacidad de nuestro algoritmo de decidir de que imagen coger la información, hará que falten algunos contornos, por lo que habrá problemas durante el relleno de los espacios vacíos.

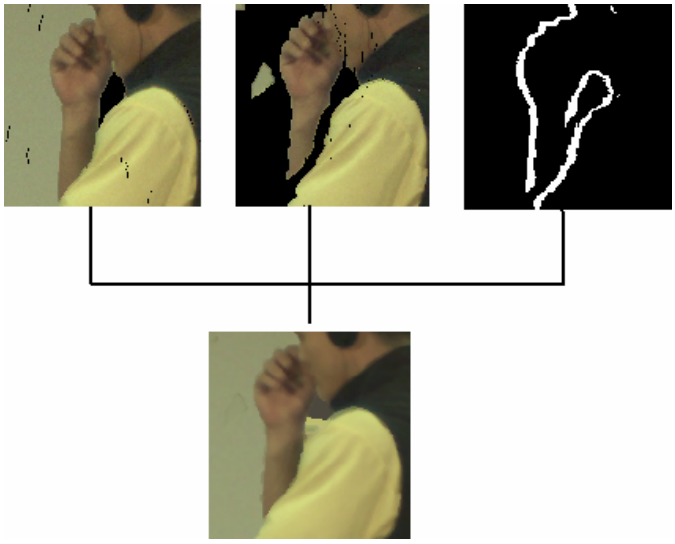


Fig. 2.17 Ejemplo del rellenado de las zonas no visibles por ambas cámaras

2.5. Método de reproyección con rectificado

El objetivo de realizar el rendering una vez rectificadas las imágenes es tener un mejor control de las nuevas posiciones que adoptan los píxeles en la imagen de la cámara virtual. Con las imágenes rectificadas sabemos que un píxel va a mantener su posición “y” después de ser reproyectado, por lo que se puede evitar el tener que recorrer la imagen verticalmente e incluso será más sencillo procesar y filtrar las imágenes para obtener mejores resultados. Gracias a la información de contornos de la imagen rectificada original, podremos distinguir durante el rendering si las zonas sin información son debidas a la existencia de un objeto o a posibles discontinuidades en los mapas de disparidad. Las zonas vaciás que aparezcan durante el rendering provocadas por errores en los mapas de disparidad podrán ser filtradas fácilmente, ya que se posee un control absoluto de las nuevas posiciones reproyectas de los píxeles originales.

Vamos a probar dos métodos de proyección con las imágenes rectificadas, uno será la reproyección geométrica y el otro será una adaptación de la reproyección directa para imágenes rectificadas. Como se verá más adelante estos métodos basados en la rectificación nos originan problemas significativos debido a que el propio proceso de rectificación requiere una interpolación previa que distorsiona la calidad de los datos originales.

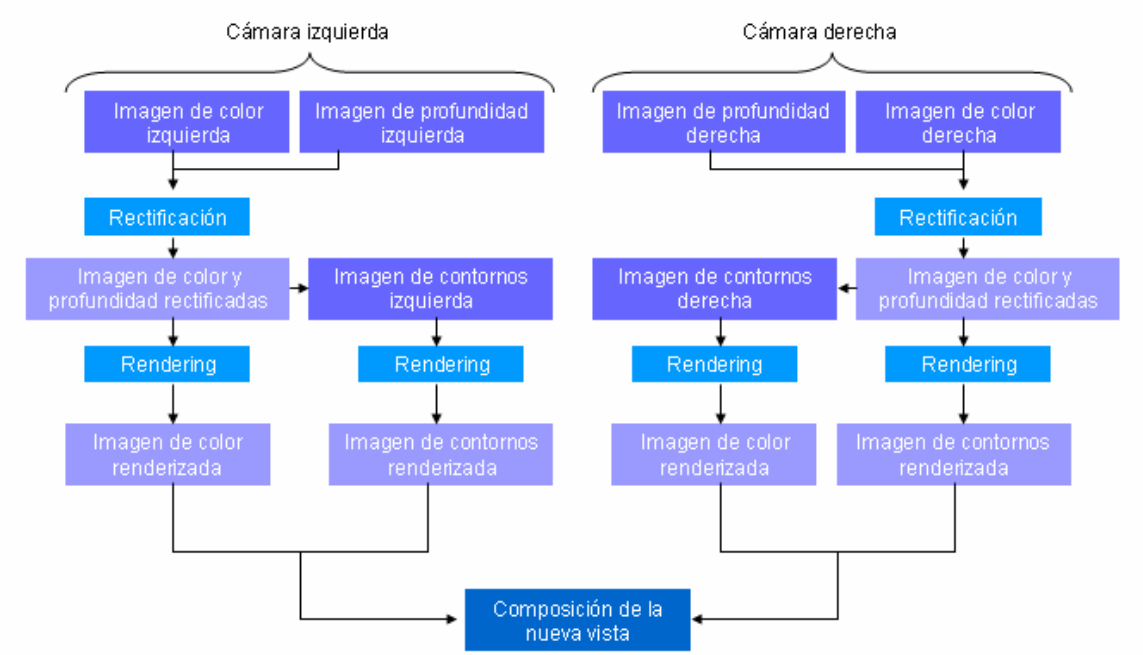
En efecto, como hemos visto anteriormente, durante la rectificación aparecen unas líneas negras en las que no hay información, estas líneas se eliminan a través de varios filtrados en los cuales la imagen pierde calidad. En las imágenes de color esta pérdida de información no es tan importante ya que visualmente no se

aprecia, en cambio, en las imágenes de profundidad, en las que la información esta en los niveles de gris, esta pérdida de información afecta de manera significativa a la hora de realizar los cálculos de las proyecciones.

Lo ideal a la hora de implementar un algoritmo que utilice este método, es realizar el calculo de los mapas de disparidad una vez tenemos las imágenes rectificadas, así no existiría esta pérdida de información.

Debido a lo expuesto anteriormente no se llegará a componer la imagen final con estos métodos ya que los resultados del rendering no son óptimos para ello.

El diagrama de bloques de este método es similar al de la reproyección directa.



2.5.1. Reproyección geométrica

La reproyección geométrica parte de la base de que tenemos el par estéreo rectificado, por este motivo es posible calcular la nueva posición del píxel en función de la profundidad y de la distancia entre cámaras sin necesidad de utilizar las matrices de proyección. La figura (2.18) muestra la distribución geométrica de este método, definido en **[7]**.

Si tenemos un píxel en la posición (i_i, j) en la imagen izquierda y sus coordenadas 3D respecto al sistema de referencia de la cámara en el plano de imagen son

$P_i(x_i, y_j, f)$, es posible encontrar su posición correspondiente (i_d, j) en la imagen derecha dentro de la misma línea "j", y de igual manera si tenemos el píxel en la imagen derecha y queremos encontrar su posición en la imagen izquierda.

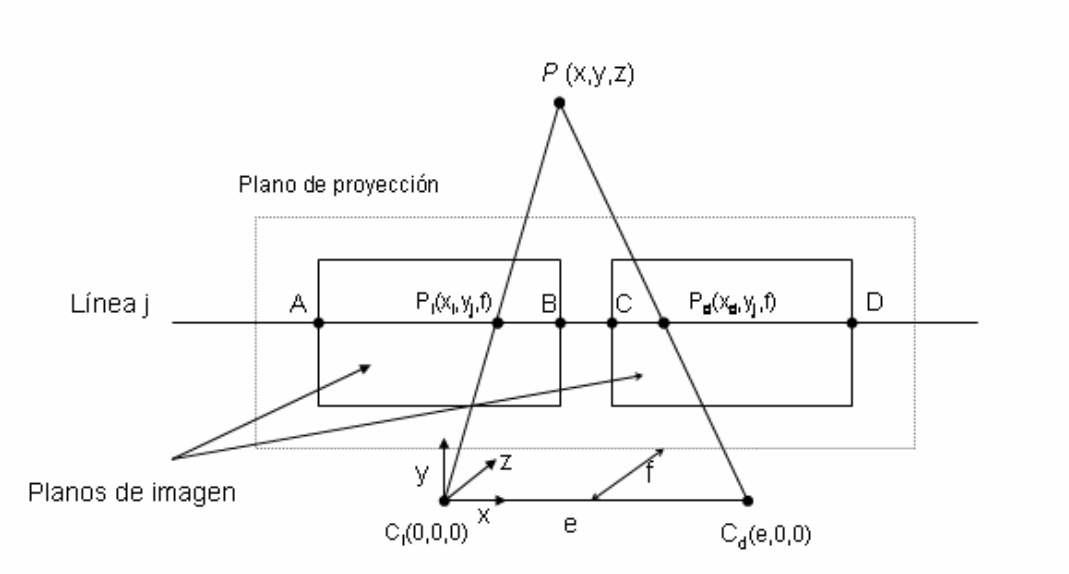


Fig. 2.18 Esquema de la reproyección geométrica

Por trigonometría obtenemos que,

$$(x_d - e)/(x - e) = f / z, \text{ y de aquí} \qquad \textbf{(2.16)}$$

$$x_d = e + f(x - e)/z \qquad \textbf{(2.17)}$$

Luego sabemos que,

$$i_d = (x_d - c)/w = (e + f(x - e)/z - c)/w, \qquad \textbf{(2.18)}$$

donde **c** es la coordenada "x" del píxel más a la izquierda de la imagen derecha y **w** es la anchura física del píxel.
Para definir las variables utilizadas en este método (A, B, C, D, w, f) se recurrió a la información proporcionada por Microsoft en [**3**], donde hablan de las características de los sistemas de captación utilizados.

Los resultados obtenidos, como ya se había avanzado anteriormente, no son satisfactorios debido a la pérdida de información durante el proceso de la rectificación. Las imágenes reproyectadas con este método tienen un error de posición de 2 hasta 8 píxeles dependiendo de la profundidad del objeto. Las siguientes imágenes muestran los resultados.

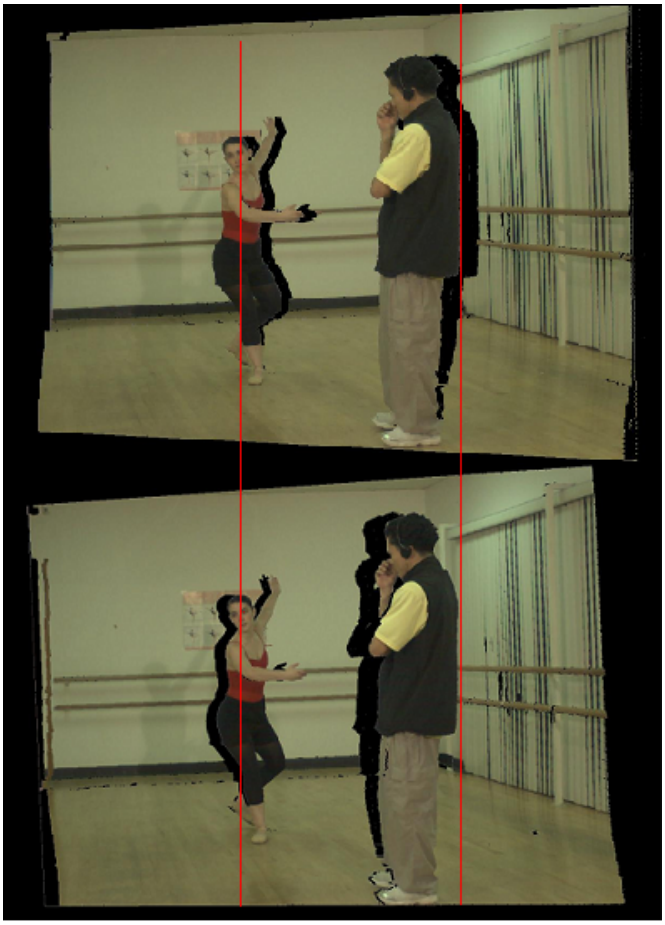


Fig. 2.19 Resultado de la reproyección geométrica

En las imágenes es difícil apreciar las diferencias que se explicaban anteriormente, ya que la resolución es menor, pero si nos fijamos, se ve claramente como las líneas verticales dibujadas no coinciden en las dos imágenes, por lo que a la hora de componer la nueva vista, los resultados no serían buenos.

2.5.2. Reproyección directa con imágenes rectificadas

La idea de este método es sencilla, partiremos del par estéreo rectificado, es decir, los planos de imagen están alineados y comparten la misma orientación. Lo que haremos será reproyectar ambas imágenes, de la misma manera que lo hacíamos en la reproyección directa, sobre el eje “x” del plano rectificado. Para ello mantendremos las matrices de rotación e intrínsecas iguales para las 3 imágenes, variando el vector de translación, con el que controlaremos la posición en “x” donde reproyectamos las imágenes.
En la reproyección directa con las imágenes originales aparecía como problema el hecho de no poder controlar las nuevas posiciones de los píxeles reproyectados en ambas dimensiones, ahora este problema se nos reduce a controlar los píxeles en 1 dimensión por lo que podemos conseguir mejores y más eficientes resultados. Una vez tenemos ambas imágenes procesadas, realizaremos la composición de la nueva imagen de la misma manera que se ha hecho en los anteriores métodos. Esta nueva imagen está situada en el plano rectificado, por lo que para darle la rotación y la posición que debe ocupar dentro del arco de cámaras lo que haremos será reproyectar esta imagen sobre el plano correcto. Como en el método de la reproyección directa, se hará una ponderación entre las matrices de las cámaras derecha e izquierda en función de la posición de la nueva vista, de la que se obtendrán las matrices de rotación e intrínseca.

Con este método tampoco se ha llegado a realizar la reconstrucción de la nueva vista, debido a los problemas mencionados anteriormente, las siguientes imágenes muestran el resultado de reproyectar las imágenes rectificadas.

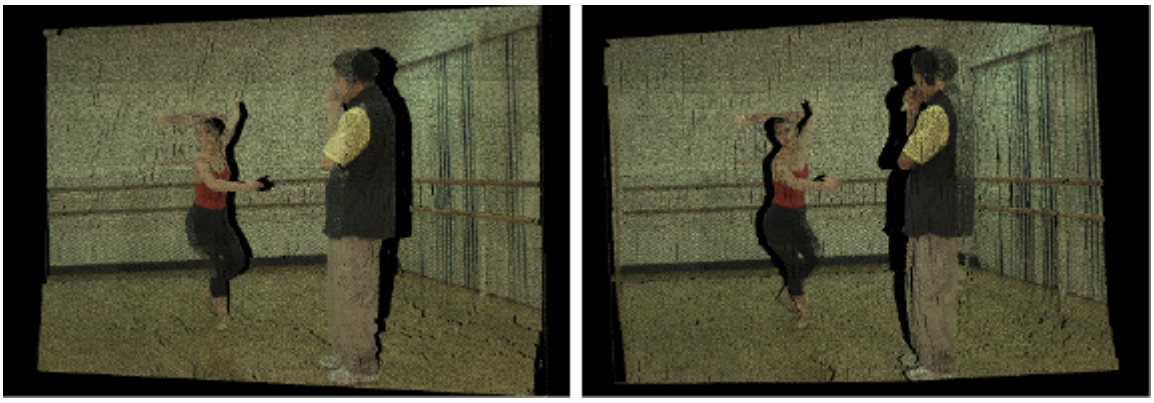


Fig. 2.20 Ejemplo de la reproyección directa de imágenes rectificadas

Como se puede apreciar la pérdida de información durante el proceso de rectificación de las imágenes de profundidad afecta seriamente a estos métodos basados en la rectificación.

CAPÍTULO 3. DESCRIPCIÓN DE LAS APLICACIONES

Se han desarrollado dos aplicaciones con el entorno de programación orientado a objetos Visual C++. Este entorno de desarrollo incluye las bibliotecas estándar del sector ATL (Active Template Library) y MFC (Microsoft Foundation Class), extensiones avanzadas del lenguaje y eficaces características del entorno de desarrollo integrado (IDE) que permiten a los programadores editar y depurar código fuente de un modo eficaz, mediante asistentes sencillos de utilizar.

A su vez se han incorporado las librerías de OpenCV (Open Source Computer Vision Library), desarrolladas por Intel, en las que se recopilan algunos de los algoritmos más conocidos dentro del mundo del procesado digital de imagen. Estas librerías son de código abierto y están disponibles para todo el mundo.

A continuación, se detallará el entorno visual utilizado para testar los métodos anteriormente descritos. Se han desarrollado dos aplicaciones, una que trabajará con las imágenes originales y una segunda que implementa los algoritmos descritos para las imágenes rectificadas.

3.1. Reproyección directa

Esta aplicación es sencilla, en la siguiente figura se muestra el entorno Visual.

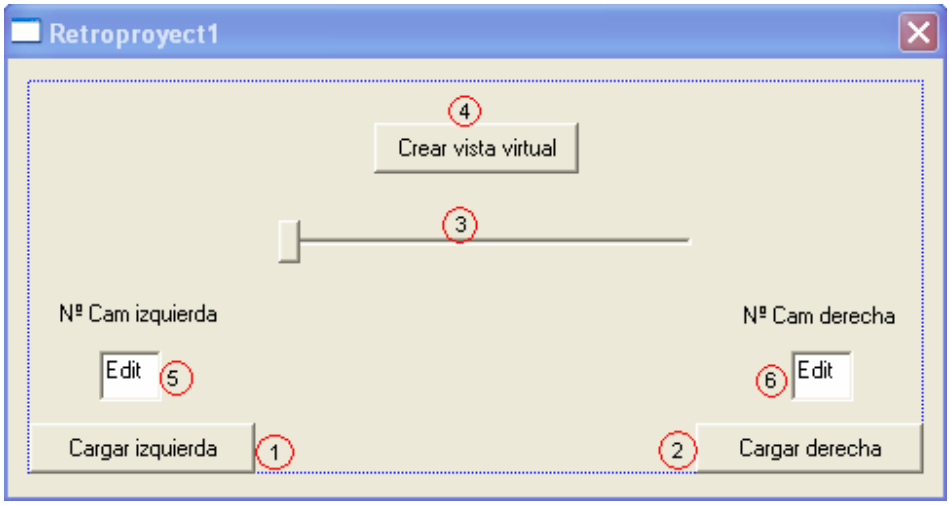


Fig. 3.1 Interfície visual de la aplicación de la reproyección directa

El primer paso es cargar las imágenes izquierda y derecha, los botones **Cargar izquierda** y **Cargar derecha** (1 y 2) realizan estás funciones, para cada uno se abrirá dos ventanas de apertura de fichero, donde deberemos seleccionar la imagen a cargar, primero la de color y luego la de profundidad.

Con las dos imágenes cargadas correctamente, el siguiente paso es elegir la posición de la cámara "virtual", de esto se encarga el dispositivo deslizante (3). Este divide el trozo de arco comprendido por las cámaras izquierda y derecha en diez posiciones, en las cuales se podrá colocar la "cámara virtual" y obtener una imagen. La posición original, será la más próxima a la cámara izquierda y a medida que aumentamos el deslizador, nos aproximamos a la posición de la cámara derecha.

El último paso es generar la nueva imagen, para ello seleccionaremos el botón **Crear vista virtual** (4), antes de esto es necesario introducir el número que identifica a cada cámara en el fichero de calibración proporcionado por Intel en las cajas 5 y 6. Una vez accionado el botón 4, por pantalla aparecerá una ventana de apertura de fichero, en la que deberemos seleccionar el fichero de texto en el que están guardados todos los parámetros de calibración de las cámaras, una vez se haya cargado el fichero, y generado la nueva imagen, el programa mostrará en una nueva ventana el resultado, es decir, la imagen vista desde la posición indicada.

3.2. Reproyección con imágenes rectificadas

El entorno Visual de esta aplicación es parecido al anterior, la siguiente figura muestra la interfaz de usuario.

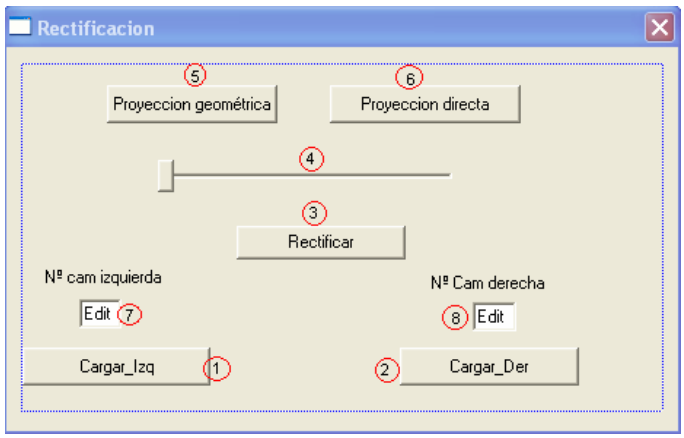


Fig. 3.2 Interfície visual de la aplicación de reproyección directa con rectificado

El funcionamiento de esta aplicación es parecido a la anterior, en primer lugar se cargan las imágenes izquierda y derecha, se introducen los números de las cámaras, y mediante el botón **Rectificar** 3, se ejecuta el proceso de rectificación. Una vez tenemos las imágenes rectificadas se elige, igual que en la proyección directa, la posición del nuevo punto de vista 4 y el método de rendering 5 o 6. La imagen interpolada del nuevo punto de vista aparecerá por pantalla.

CAPÍTULO 4. CONCLUSIONES Y LÍNEAS FUTURAS

La visión estereoscópica es una de las áreas del procesado de imagen que está despertando mayor interés tanto en el ámbito académico como en el de la industria. Se trata de un área bastante reciente y que a pesar de que en los últimos diez años ha evolucionado mucho, todavía existen varios problemas que no han conseguido resolverse satisfactoriamente debido a su complejidad. Son muchas las aplicaciones relacionadas con técnicas estereoscópicas como codificación y transmisión de sistemas multicamaras, teleinmersión, estudios 3D del relieve de los terrenos, etc.

En este trabajo, se ha realizado un estudio sobre dos métodos de rendering, con el objetivo de interpolar las vistas intermedias de un sistema de captación multicámara.

Uno de nuestros mayores obstáculos, como ya se ha comentado durante esta memoria, ha sido el no partir con una información propia, por lo que ha sido necesario dedicar un tiempo importante a definir y entender que era lo que teníamos y desde donde partíamos. Los sistemas de captación multicamara requieren un alto nivel de sincronización y un complejo proceso de calibración, con el que se caracterizan las cámaras. Todo esto requiere un gran esfuerzo y podría ser tratado como un proyecto independiente. De la misma manera el coste ecónomico que supondría construir un sistema de estas características queda fuera de nuestro alcance.

Los resultados obtenidos se pueden valorar teniendo en cuenta dos factores: la calidad de imagen y el tiempo de procesado.

Primero hablaremos de la calidad de imagen. El hecho de utilizar una etapa previa de rectificación reduce la dificultad en el proceso de búsqueda de correspondencias entre píxeles así como en el reconocimiento de objetos y en el rendering. Gracias a la rectificación conseguimos que todos los píxeles de una misma línea horizontal de la imagen se reproyecten sobre la misma línea en la imagen de la vista virtual.

Teóricamente, esto hace que el algoritmo de rendering posea mayor control sobre las posiciones reproyectadas de los píxeles, ya que siempre mantendrán la misma posición en el eje “y”. Esto hará más sencillo el filtrado de los errores producidos por las discontinuidades en los mapas de disparidad, así como el control de las oclusiones en la nueva vista interpolada.

Por lo tanto, el usar la etapa de rectificación, permite a nuestro algoritmo filtrar fácilmente posibles errores durante el rendering, y tener un mayor control del movimiento de los objetos. De ahí que los resultados obtenidos con estos métodos sean de mayor calidad.

En contradicción con lo anterior, en nuestro caso no se han obtenido buenos resultados utilizando los métodos de rendering basados en imágenes rectificadas. Como se explica en esta memoria, el hecho de rectificar conlleva una pérdida de información de la imagen original. Esta pérdida no es apreciable visualmente en las imágenes de textura, en cambio, en las imágenes de profundidad esta pérdida de información es muy importante.

Las imágenes de profundidad expresan la información a través de sus niveles de gris, por lo que al modificar sus valores, estamos modificando la profundidad de los objetos de la escena. Los efectos que esto conlleva son claramente visibles durante el rendering debido a la gran precisión que se necesita a la hora de realizar los cálculos de las reproyecciones.

En segundo lugar esta el tema del coste computacional o tiempo se procesado. Es lógico pensar que un algoritmo con más etapas es a su vez más lento, pero esto no siempre es así.

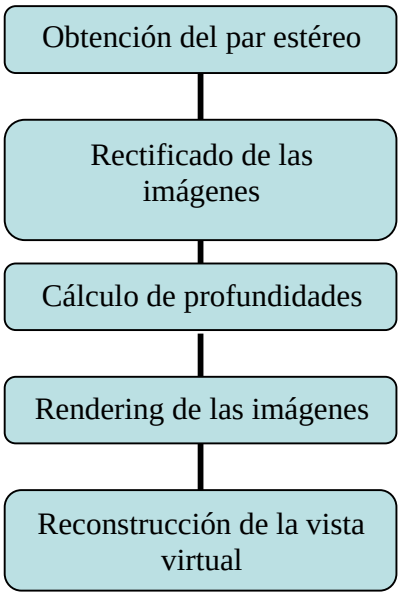
Como hemos visto anteriormente, el rectificado reduce el problema de la búsqueda de correspondencias entre píxeles, facilitando el cálculo de disparidades así como los procesos de rendering y de reconocimiento de objetos. Todo esto hace que los algoritmos utilizados sean más eficientes y rápidos, por lo que el coste computacional disminuirá notablemente. Por otro lado, decir que existen métodos de rectificación basados en la linealidad de esta operación, que reducen el tiempo de ejecución de este algoritmo.

En nuestro trabajo, el problema del tiempo no se ha tenido en cuenta, por lo que no se han optimizado los algoritmos desarrollados. Por este motivo y dado que la etapa del cálculo de disparidades no ha sido implementada (etapa en la que el cálculo de correspondencias es muy importante), el método de la reproyección directa es más rápido que el método de reproyección basado en imágenes rectificadas.

Aunque pueda parecer que los resultados obtenidos se contradicen con el estudio teórico, este es fácilmente explicable ya que el objetivo de este proyecto era analizar dos posibles métodos de rendering, ambos con

imágenes de profundidad ya calculadas. Esto sería totalmente diferente, en el caso de que hubiésemos tenido que encontrar las imágenes de profundidad, como suele ocurrir en proyectos de teleinmersión completos. Por este motivo en aplicaciones reales, donde el tiempo de cálculo es un factor tan importante como la calidad se suele trabajar con imágenes rectificadas.

El esquema que seguiría el proceso de interpolación de vistas en un proyecto de este tipo sería:



Como se ve en el esquema, el rectificado de las imágenes se realiza antes del cálculo de disparidades. Con esta distribución desaparece el problema de la pérdida de información en los mapas de profundidad, ya que no sería necesario rectificarlos. Los mapas de disparidad serían calculados a partir de las imágenes rectificadas.

Por lo que, la conclusión final para futuros proyectos comerciales es que el rectificado es una etapa importante y necesaria para temas de procesado de imagen 3D, ya que mejora los resultados y nos permite reducir el coste computacional de los algoritmos, haciendo posible aplicaciones en tiempo real.

Como proyectos futuros que continuen con el estudio realizado en este trabajo, podriamos contemplar la idea de desarrollar un proyecto de interpolación de vistas optimizando los algoritmos implementados, de manera que fuera posible trabajar con las imágenes captadas por un sistema multicámara a tiempo real.

La primera idea sería la optimización del algoritmo de rectificación. Como ya se ha comentado en la memoria, la rectificación es una operación lineal, por lo tanto, existe la posibilidad de rectificar de forma directa todos los puntos de la imagen partiendo de las posiciones rectificadas de los 4 extremos de la misma. Por lo que disminuiríamos notablemente el coste computacional de este proceso.

Otra idea sería solucionar los problemas de las zonas ocultas trabajando con algoritmos de reconocimiento de objetos. Con ello conseguiríamos segmentar la imagen por objetos, por lo que sería posible extraer la información de estas zonas mediante un análisis previo de la escena.

Dado que el tema de la teleinmersión es de los más novedosos dentro del procesado de imagen digital, otra de las líneas de investigación que podrían surgir tras este trabajo podría estar relacionada con la codificación y transmisión de sistemas multicamaras mediante técnicas estereóscopicas.

Muchos de los trabajos que se están realizando actualmente tienen como objetivo la transmisión entre puntos distantes de escenas tridimensionales. Lo que se pretende es transmitir la información estéreo captada por un sistema multicámara de manera que el receptor pueda generarse la escena tridimensional.

Basándonos en los fundamentos de la estereoscopía, se podría implementar un algoritmo de codificación de las imágenes del par estéreo. Sería posible generar una vista a partir de la otra, gracias a la redundancia de información que existe entre ambas. Por lo que se podría enviar una imagen codificada junto con la predicción de error que existiría al generar la nueva vista. Este error sería calculado por el codificador ya que conoce en todo momento la información de ambas imágenes. Por lo que, el mayor coste computacional lo tendría el codificador. El decodificador solo debería preocuparse de procesar la información recibida y corregir los errores.

Esto último sería una buena manera de transmitir un escenario 3D a cualquier punto, ya que el receptor solo necesitaría un sencillo decodificador y un sistema de proyección adecuado con el que generar la escena.

REFERÉNCIAS

[1] Gurewich O., Gurewich N., *Aprendiendo Visual C++ en 21 días*, Prentice Hall (1998).

[2] Zhang, Z, "A flexible new technique for camera calibration". IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 11, 1130-1334, 2000.

[3] Zitnick, L., Bing S., Uyttendaele, M., Winder, S., Szeliski, R., "Hight-quality video view interpolation using a layered representation". Interactive Visual Media Group, Microsoft Research, Redmond, WA.
http://research.microsoft.com

[4] Fusiello, A., Trucco, E,. Verri, A., " A compact algorithm for rectification of stereo pairs". *Machine Vision and Applications* 12: 16–22, 2000.

[5] Sonka, M., Hlavac, V., Boyle, R., *Image processing, analysis, and machine vision,* PWS (1998).

[6] Ayache, N., Hansen, C., " Rectification of images for binocular and trinocular vision". Rapports de recherché, Unité de Recherche Inria-Rocquencourt (1988)

[7] Wan, M., Zhang, N., Qu, H., Kaufman, A., "Interactive Stereoscopic Rendering of Volumetric Environments". IEEE Transactions on visualization and computer graphics, Vol 10, Nº 1, January/February 2004.

[8] Mark, W., McMillan, L., Bishop, G., "Post Rendering 3D warping". Departament of computer Science University of North Carolina at Chapel Hill (1997).

[9] Mandal, Ch., Zhao, H., Vemuri, B., Aggarwal, J., "3D Shape Reconstruction from multiples Views". CISE Departament, University os Florida, ECE Departament, University os Texas at Austin.

[10] Zhu, Z., Jiang, M., Wu, Ch., "3D Reconstruction". ECSE 6650 Computer Vision Project 2.

[11] Manual de OpenCV
http://isa.umh.es/pfc/rmvision/opencv-0 9 5.pdf

[12] OpenCV Webpage
http://sorceforge.net/projects/opencvlibrary

ANEXOS

Anexo 1. Parámetros de calibración proporcionados por Microsoft

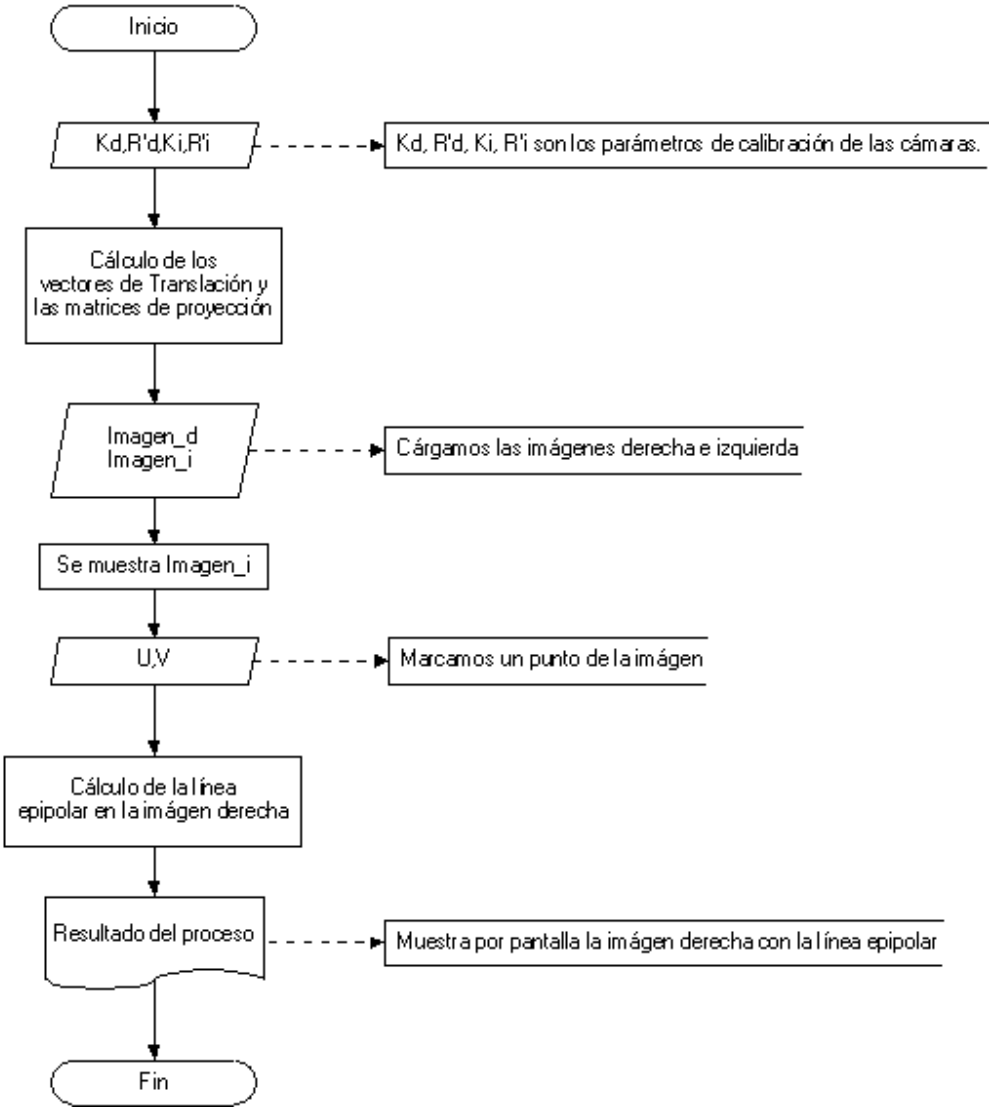
Estos son los parámetros de calibración de las 8 cámaras proporcionados por Microsoft. El primer número, indica el número de la cámara. La matriz [3x3], que hay debajo del número de la cámara, es la matriz intínseca. Los dos parámetros independientes son las distorsiones de barril o distorsiones de la lente, considerados como 0. Y por último la matriz [3x4] final es la matriz definida en la memoria como R'.

```
0
1918.270000    2.489820        494.085000
0.0     1922.580000    447.736000
0.0     0.0     1.0
0.000000        0.000000
0.949462        0.046934        0.310324        -15.094651
-0.042337       0.998867        -0.021532       0.189829
-0.310985       0.007308        0.950373        1.383263

1
1913.690000    -0.143610        533.307000
0.0     1918.170000    398.171000
0.0     0.0     1.0
0.000000        0.000000
0.972850        0.010365        0.231187        -11.589320
-0.012981       0.999864        0.009794        -0.355771
-0.231056       -0.012528       0.972852        1.045534

2
1914.070000    0.343703        564.645000
0.0     1918.500000    428.422000
0.0     0.0     1.0
0.000000        0.000000
0.989230        0.003946        0.146295        -7.784865
-0.004391       0.999983        0.002724        -0.431597
-0.146283       -0.003337       0.989230        1.392058

3
1909.910000    0.571503        545.069000
0.0     1915.890000    394.306000
0.0     0.0     1.0
0.000000        0.000000
0.996415        0.026023        0.080480        -3.903715
-0.026884       0.999591        0.009614        -0.040429
-0.080197       -0.011743       0.996707        0.168691
```

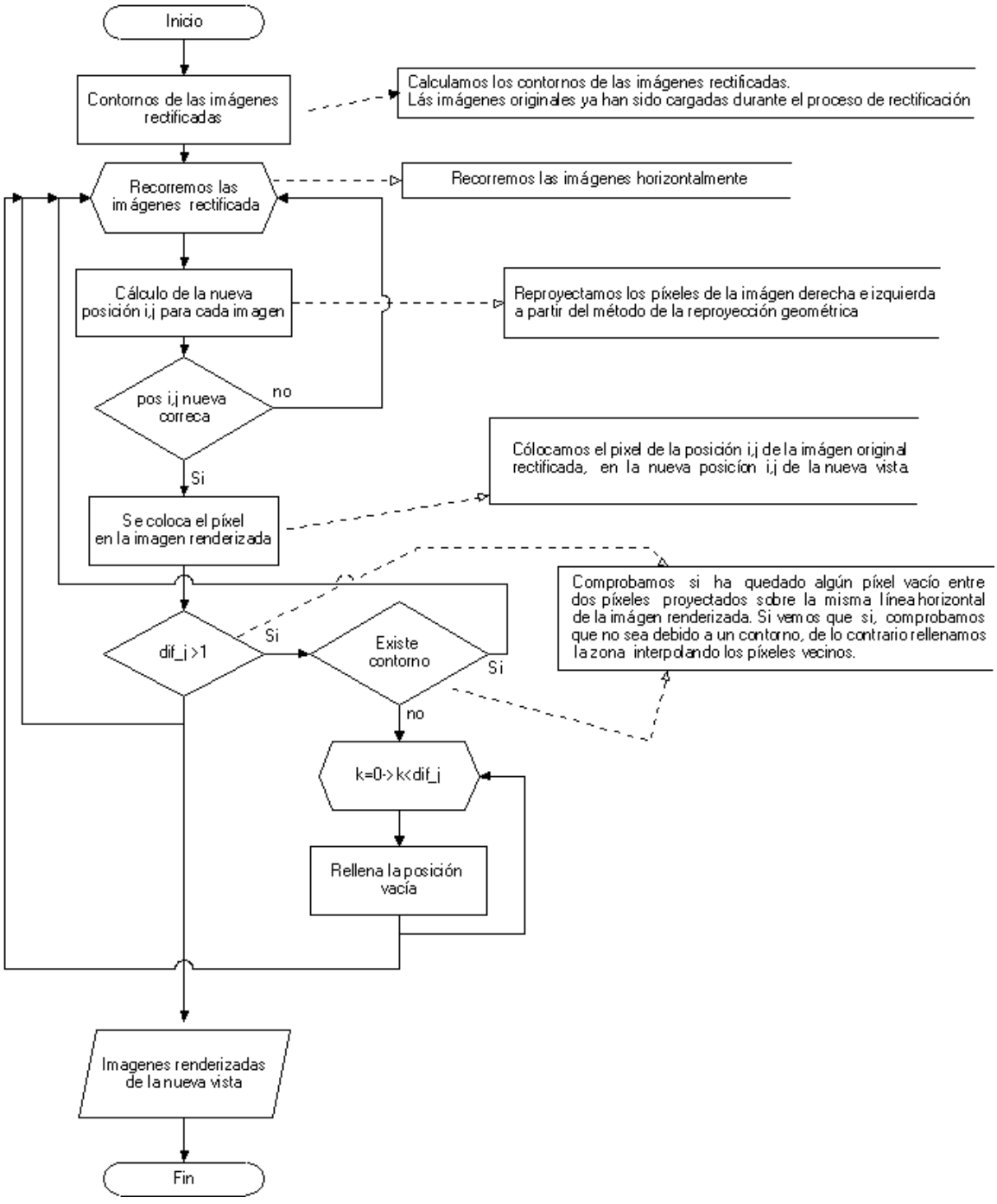
```
4
1908.250000    0.335031        560.336000
0.0     1914.160000    409.596000
0.0     0.0     1.0
0.000000        0.000000
1.000000        0.000000        0.000000        -0.000002
0.000000        1.000000        -0.000000       0.000006
0.000000        -0.000000       1.000000        0.000000


5
1915.780000    1.210910        527.609000
0.0     1921.730000    394.455000
0.0     0.0     1.0
0.000000        0.000000
0.998175        0.028914        -0.053000       3.849864
-0.028594       0.999567        0.006786        0.041657
0.053173        -0.005258       0.998570        0.428967

6
1910.570000    0.786148        578.134000
0.0     1916.270000    404.469000
0.0     0.0     1.0
0.000000        0.000000
0.988494        0.037674        -0.146458       7.602324
-0.037105       0.999288        0.006622        -0.045578
0.146603        -0.001111       0.989188        -0.044837

7
1929.090000    0.831916        585.520000
0.0     1937.210000    416.944000
0.0     0.0     1.0
0.000000        0.000000
0.975422        0.032363        -0.217910       11.142041
-0.033721       0.999425        -0.002516       0.200655
0.217705        0.009803        0.975952        -0.230057
```

Anexo 2. Diagramas de flujo de los algoritmos

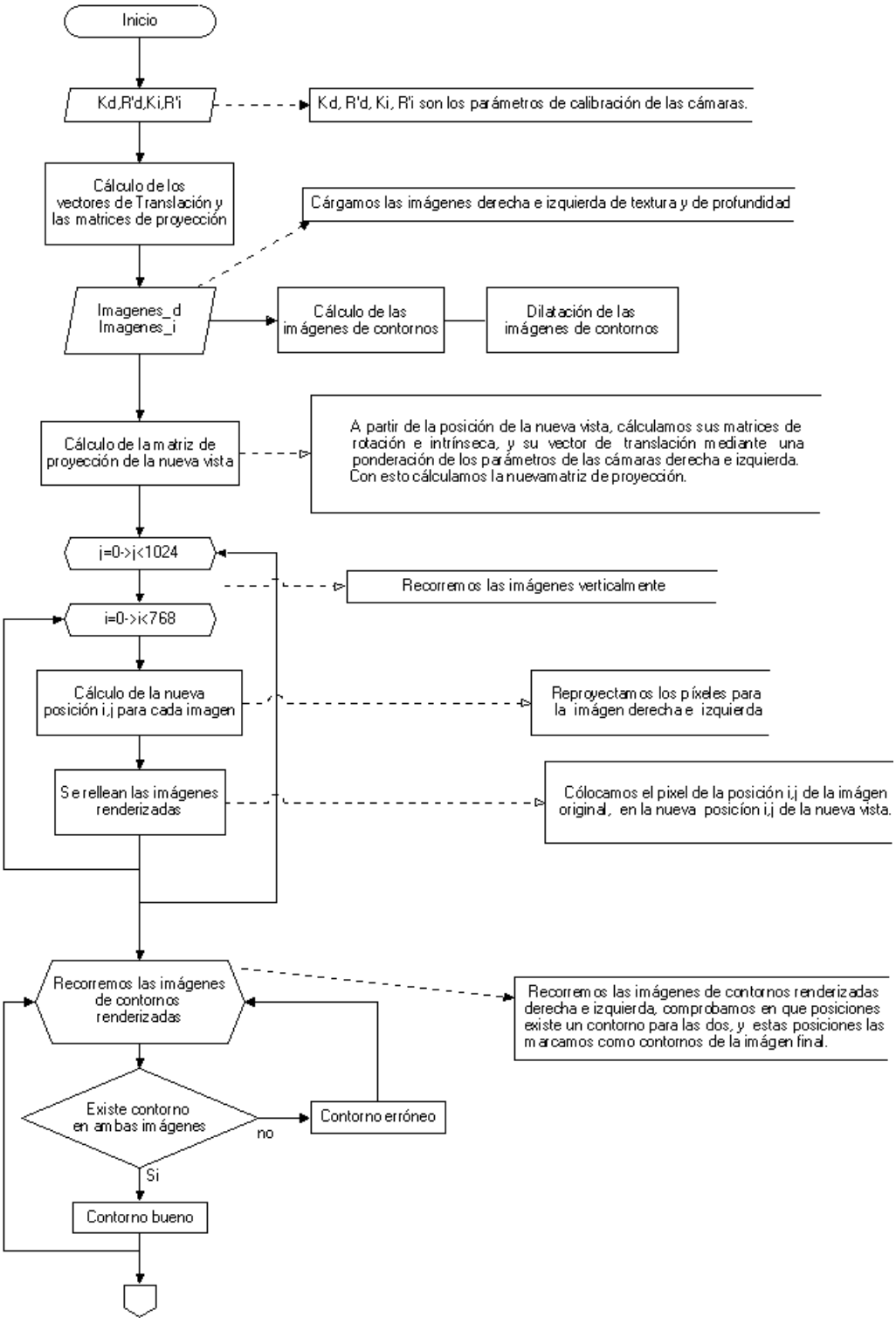
Cálculo de las líneas epipolares

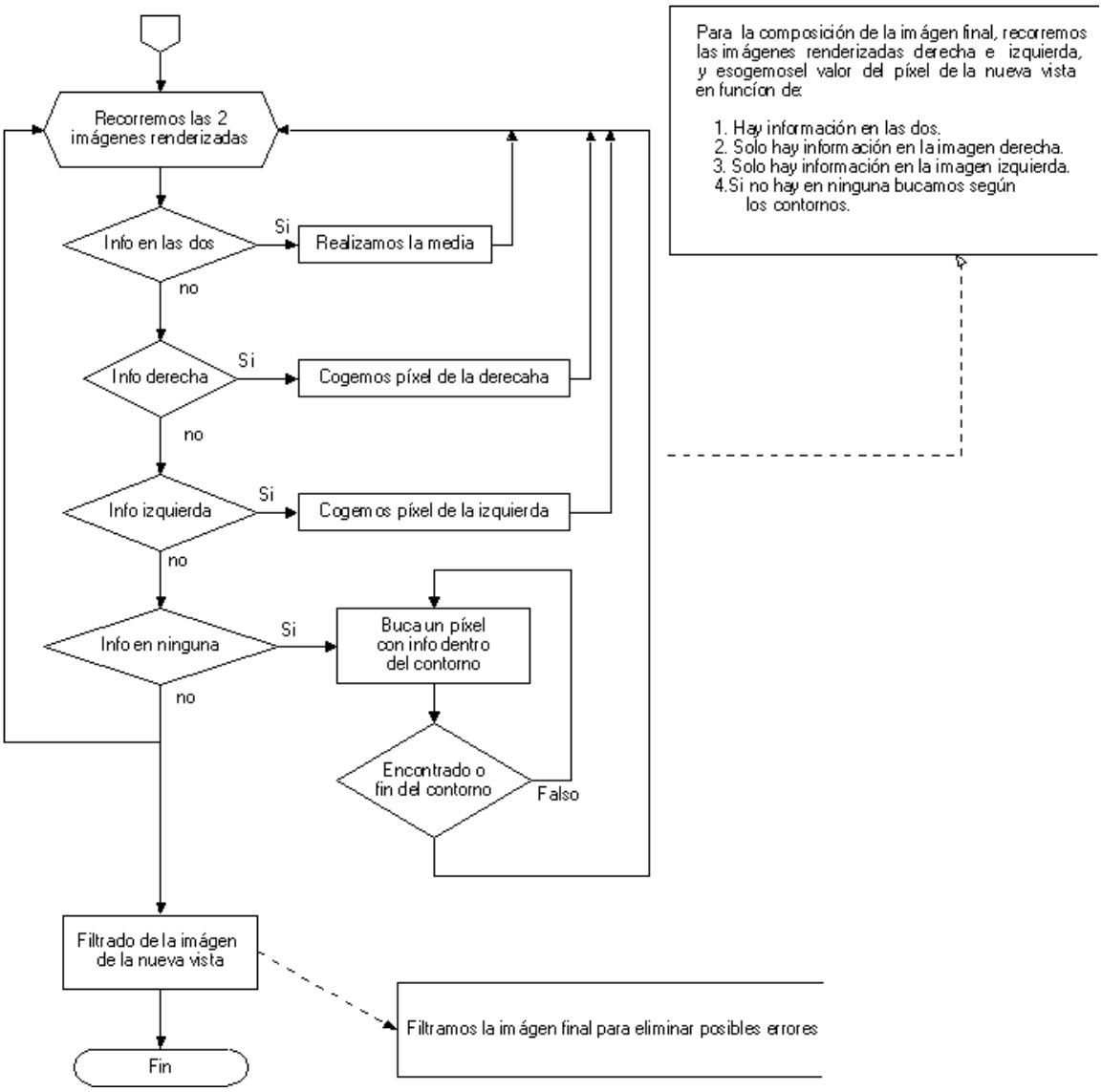


Rectificación del par estéreo

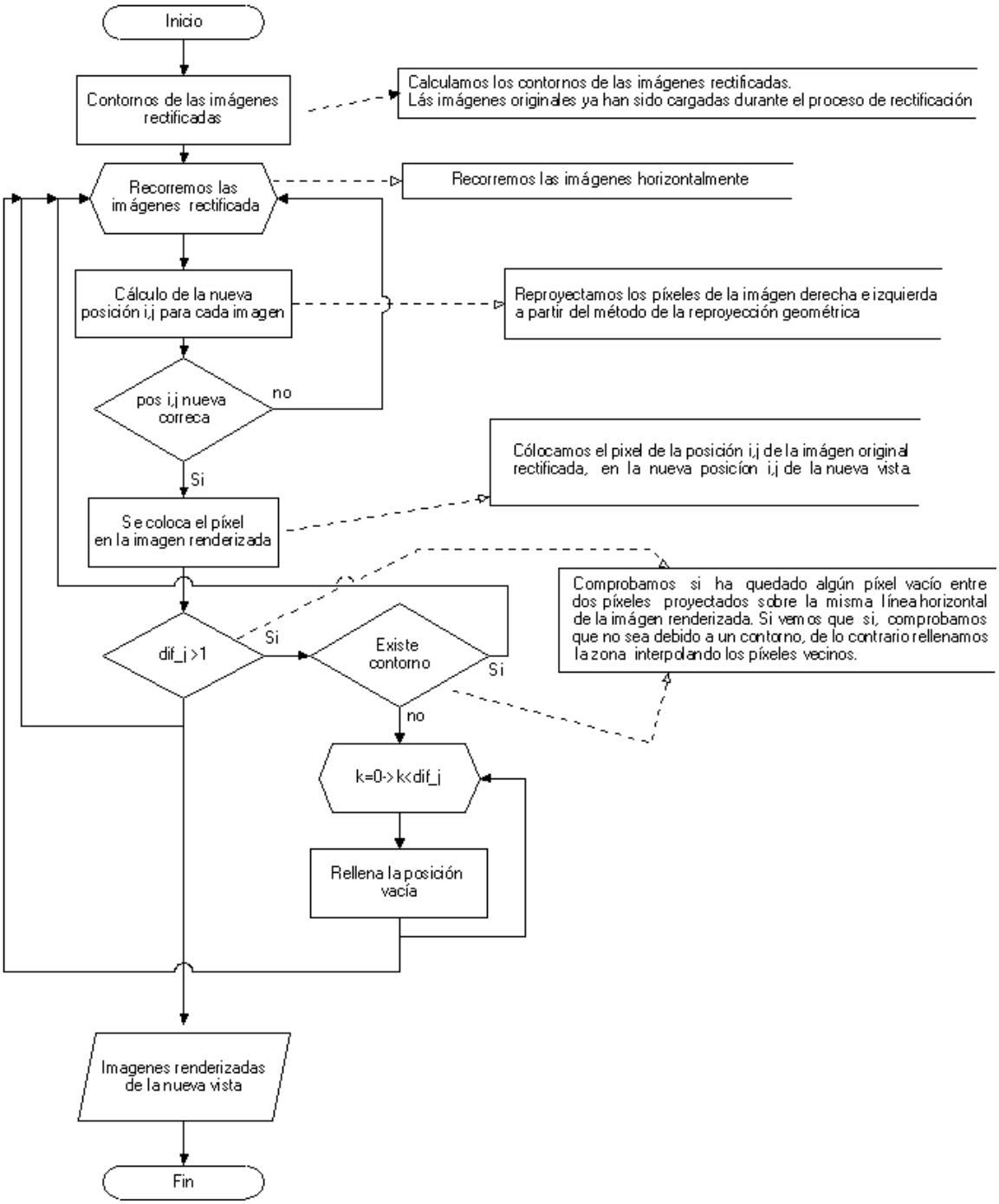


Reproyección directa




Recorremos las 2 imágenes renderizadas
Info en las dos
Si
Realizamos la media
no
Info derecha
Si
Cogemos píxel de la derecaha
no
Info izquierda
Si
Cogemos píxel de la izquierda
no
Info en ninguna
Si
Buca un píxel con info dentro del contorno
no
Encontrado o fin del contorno
Falso
Filtrado de la imágen de la nueva vista
Fin
Para la composición de la imágen final, recorremos las imágenes renderizadas derecha e izquierda, y esogemosel valor del píxel de la nueva vista en funcíon de:
1. Hay información en las dos.
2. Solo hay información en la imagen derecha.
3. Solo hay información en la imagen izquierda.
4.Si no hay en ninguna bucamos según los contornos.
Filtramos la imágen final para eliminar posibles errores

Reproyección geométrica de imágenes rectificadas



Reproyección directa de imágenes rectificadas