Máster en Sistemas Inteligentes

Trabajo Final de Máster

Estimación del mapa de profundidad de un panorama esférico mediante interacción del usuario

Autor:
Miguel Saura Herreros

Tutorizado por:
José Ribelles Miguel
M.Ángeles López Malo

Curso académico 2017/2018

Resumen

En este trabajo, se aborda el problema de obtener un mapa de profundidades de un panorama esférico a partir de la información proporcionada por un usuario de manera interactiva. En primer lugar se estudian distintas formas de interacción con un panorama esférico, para lo que se ha desarrollado una aplicación Web que muestra distintos tipos de presentación de un panorama y permite introducir la información sobre la profundidad de los elementos presentes en la escena. La aplicación Web se ha implementado mediante el uso de las tecnologías HTML5, Javascript y WebGL. Esta información se utilizará para obtener un panorama de profundidad mediante la extensión de un método de estimación de profundidad, desarrollado para imágenes normales (proyección en perspectiva), a panoramas esféricos.

Palabras clave

Panorama esférico, Mapa de profundidades, Mapa de distancias, Interacción hombre-ordenador.

Keywords

Spherical panorama, Depth map, Range map, Human-computer interaction.

Índice general

Índice de figuras

Capítulo 1

Introducción

Un panorama esférico es una fotografía que captura una vista de 360º en horizontal y 180º en vertical. Se trata de una imagen que almacena la proyección del mundo 3D en una esfera cuyo centro coincide con la posición del observador. Se utiliza a menudo para que el usuario se pueda sumergir en un entorno y en ocasiones se le conoce con el nombre de Virtual Reality Photography [1]. Quizá, el ejemplo más conocido en el que se utilizan este tipo de fotos sea el de la aplicación Google Street View [2] que permite a un usuario conocer a pie de calle ciudades de todo el mundo (ver Figura 1.1).

Figura 1.1: Muestra de un panorama esférico (Fuente: Google Street View [2]).

El mundo de los videojuegos también hace uso de este tipo de panoramas. En la figura 1.2 se muestra la imagen panorámica como una esfera alrededor del observador. Generalmente se implementa como texturas de un *Skybox*, que no es más que un cubo que se utiliza para representar el fondo de una escena. En este caso concreto el panorama

captura el paisaje incluyendo cielo, montañas y, en general, todo aquello que resulta lejano al usuario. Sin embargo, en un videojuego es raro el uso de panoramas esféricos que capturan escenas de interior, como el vestíbulo de un edificio, el patio de un teatro o simplemente la habitación de una vivienda. El problema es que este tipo de escena muestra elementos cercanos al usuario y por lo tanto requiere que los elementos virtuales 3D del juego (como otros personajes, balones, mascotas, etc.) puedan interactuar con los elementos capturados en el panorama que carece de información de profundidad.

Una manera de resolver este problema pasaría por recuperar la información de profundidad de la fotografía panorámica permitiendo convertir el escenario 2D en un mundo 3D. De esta forma los objetos sintéticos sí que podrían interaccionar (como rodear, rebotar, tropezar, golpear, etc.) con los elementos capturados en el propio panorama. Pero no solo los videojuegos se podrían beneficiar de este proceso, recuperar información de profundidad permitiría también poder editar con facilidad cualquier panorama, eliminar elementos, moverlos, cambiar sus dimensiones, o incorporar elementos extraídos de otros panoramas para crear nuevas escenas que muestran lugares que no existen manteniendo un alto realismo visual.

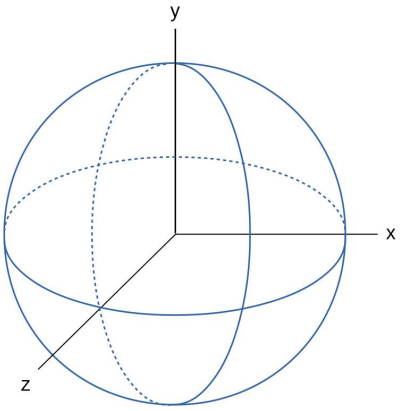


Figura 1.2: Sistema de coordenadas cartesianas de la proyección de una imagen 360º en una esfera. El observador está situado en el centro de la esfera, donde también situamos el origen del sistema.

1.1. Objetivos

El principal objetivo de este trabajo es estimar la información de profundidad de un panorama esférico. Este problema ha sido estudiado con imágenes que representan vistas en una sola dirección (la habitual fotografía). En particular, algunos de los métodos propuestos en la literatura [3] se basan en que sea el propio usuario el que proporcione algo de información acerca de la profundidad.

Al usuario le resulta sencillo especificar dicha información ya que es capaz de comprender y reconocer la escena que está observando con facilidad. Además, lo realiza de

manera interactiva con herramientas tan simples como el dibujo de puntos, líneas o garabatos. Sin embargo, un panorama esférico se puede mostrar al usuario de al menos tres formas diferentes: como una única imagen panorámica, como seis imágenes en forma de mapa de cubo, y también como resultado de una proyección esférica en la que el usuario puede además cambiar la dirección de la vista de manera interactiva. Cada una de estas formas de representación, plantea dificultades para la interacción a la hora de introducir información sobre la profundidad de distintos punto o elementos de la escena.

En este trabajo se abarca tanto el análisis de la interacción con el usuario, como la extensión del método de estimación de profundidad, con el objetivo de estimar las profundidades de un panorama esférico. Este objetivo principal se puede desglosar en los siguientes subobjetivos:

- Representar el panorama esférico en cada tipo de vista.
- Implementar las herramientas para proporcionar información de profundidad por parte del usuario, en cualquier tipo de vista.
- Conseguir una interacción coherente entre las herramientas y las distintas vistas.
- Analizar las ventajas y los inconvenientes de cada tipo de representación.
- Extender el método de estimación de profundidad para imágenes en perspectiva, para poder aplicarlo sobre panoramas esféricos.

Capítulo 2

Trabajo previo

2.1. Obtención de profundidades de una imagen

Una imagen es una proyección de una escena tridimensional sobre un plano. Al proyectar la escena en un plano 2D se pierde la tercera dimensión, esto es, la profundidad a la que están los objetos respecto de la posición del observador.

Existen muchos métodos para obtener o recuperar la profundidad de una imagen. En primer lugar, están los que utilizan sistemas de múltiples cámaras o cámaras especiales de modo que capturan profundidades o distancias a la vez que capturan la imagen o información extra que permita calcularlas [4]. Entre estos están los sistemas basados en luz estructurada, los basados en el tiempo de vuelo de la luz y los sistemas estereoscópicos.

Estos sistemas requieren de hardware específico: una o más cámaras adicionales (visión estereoscópica), un proyector de luz que aplica un patrón sobre la escena (luz estructurada), o un sistema de detección de rango basado en tiempo de vuelo de un haz de luz. La principal ventaja de estos sistemas es que suelen proporcionar una buena definición de la geometría presente en la escena, mientras que su principal desventaja es el tiempo de adquisición y el coste del equipamiento.

Este tipo de sistemas ya ha sido utilizado para capturar información de profundidad en el caso de imágenes panorámicas. Este es el caso de [5], donde utilizan un sistema basado en luz estructurada para adquirir la geometría al mismo tiempo que la imagen, con ayuda de un operador que inspecciona, durante la adquisición, el modelo 3D que se va obteniendo.

Sin embargo, en muchos casos no se dispone de un sistema de adquisición específico, o no es posible usarlo, por ejemplo si se trata de un dibujo. Hay métodos que abordan el problema de obtener profundidades a partir de una única imagen en color de manera

automática: unos sistemas se basan en aprendizaje automático [6], y otros se basan en Gestalt [7]. Sin embargo, otros métodos abordan el problema de manera semiautomática, explotando la habilidad humana de interpretar las imágenes 2D. La extensión de las herramientas tradicionales de edición de fotografías a 3D [8], aunque es la solución más directa y proporciona una precisión excelente, requiere de mucha interacción por parte del usuario y de un nivel de destreza elevado. De modo que algunos trabajos han intentado reducir drásticamente este esfuerzo diseñando herramientas que se basan en bosquejos, garabatos o puntos proporcionados por el usuario [3, 9, 10] de manera que transforman esta información imprecisa en un mapa de profundidades relativas.

Los métodos que se enmarcan en esta última categoría son los más relacionados con este trabajo. Todos ellos utilizan imágenes en perspectiva y, hasta donde sabemos, no hay ninguno que trabaje con panoramas esféricos. Este trabajo, se basa en uno de estos métodos, que se detalla a continuación.

2.2. Profundidades a partir de interacción

En general, estos métodos utilizan la información que el usuario introduce interactivamente, para estimar los valores de profundidad de todos los píxeles en función del contenido de la imagen. El objetivo es obtener un mapa de profundidades relativas, esto es, más que valores exactos de profundidad se pretende obtener una ordenación en profundidad de los objetos presentes en la escena. En [9] utilizan garabatos para asignar valores de profundidad absolutos, que son propagados al resto de elementos de la escena mediante un algoritmo de optimización. En [10] utilizan pares de garabatos que representan desigualdades en profundidad, y que se incluyen como restricciones relativas para guiar la optimización.

En este trabajo se usa el método [3] que permite integrar restricciones de distintos tipos. Este método realiza un proceso de optimización al que se le aplican las restricciones derivadas de la información que proporciona el usuario. El usuario puede introducir los siguientes elementos:

- Puntos con un valor predeterminado de profundidad (llamados semillas). Se propone que el usuario pueda marcar puntos en la lejanía (valor predeterminado 0) y puntos cercanos al observador (valor predeterminado 255), pero el método admite asignar cualquier valor de profundidad a cualquier píxel. Las profundidades resultantes variarán típicamente entre estos dos valores, de modo que el método requiere que haya al menos dos puntos a los que se les asigne distinta profundidad.

- Pares de puntos con la misma profundidad. Cuando el usuario añade un par de puntos de este tipo, el método impone la restricción de que en ambos puntos resulte la misma profundidad.

- Pares de puntos con distinta profundidad. El usuario marca un punto y a continuación otro que está algo más lejos que el primero. El método impone la restricción de que la diferencia de profundidad entre ambos sea mayor que un umbral, que es un parámetro del método (en los experimentos se usa valor 20 cuando las profundidades varían en el rango $[0, 255]$).
- Plano del suelo. El usuario ha de indicar dos cosas: la línea del horizonte (bien mediante una línea recta, o bien a partir de 4 rectas que fugan) y una serie de puntos del suelo (mediante un garabato). El algoritmo incluye esta información en el proceso de optimización restringiendo las profundidades de estos puntos para que estén ubicados en un mismo plano que a su vez pase por el horizonte.

En este trabajo se extiende este método a imágenes panorámicas. Para conseguirlo, es necesario abordar tanto la interacción del usuario con la imagen panorámica como la propia extensión del método de modo que los resultados sean coherentes para toda la escena.

Capítulo 3

Interacción con panoramas esféricos

3.1. Coordenadas esféricas y notación

Las coordenadas esféricas es un sistema de coordenadas que permite describir de forma natural las posiciones en una esfera mediante tres parámetros:

$$(r, \theta, \phi)$$

Generalmente se define:

- la distancia r es la distancia de un punto al centro de la esfera (el radio).
- el ángulo θ es el acimut (o ángulo acimutal): el ángulo que nos da una idea de la posición del punto en el sentido horizontal (ver Figura 3.1). Se mide de 0^o a 360^o respecto del eje $+Z$.
- el ángulo ϕ es el ángulo polar (también llamado ángulo cenital o colatitud): el ángulo que nos da una idea de la posición en el sentido vertical (ver Figura 3.1). Se mide de 0^o a 180^o respecto del eje $+Y$.

Por ejemplo:

- (1, 0^o, 0^o), se corresponde con el eje $+Y$.
- (1, 0^o, 90^o), se corresponde con el eje $+Z$.
- (1, 90^o, 90^o), se corresponde con el eje $+X$.

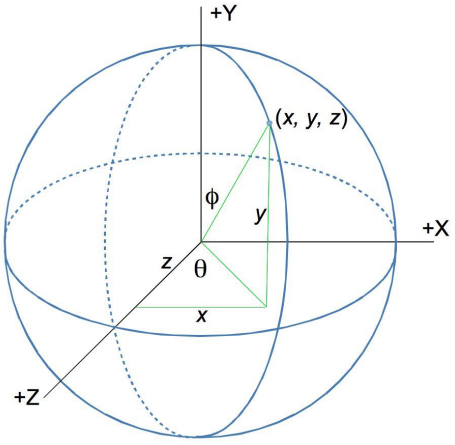


Figura 3.1: Coordenadas esféricas. El acimut o ángulo acimutal se representa mediante $\theta \in [0°, 360°]$. El ángulo polar se representa mediante $\phi \in [0°, 180°]$.

En ciertas disciplinas (como la geografía o la astronomía) los ángulos acimutal y polar se relacionan con la longitud y la latitud, respectivamente, y la distancia r con la altura. En nuestro caso, el observador está situado en el centro de la esfera, y consideramos un sistema de coordenadas con origen en el centro de la esfera. Por tanto, la distancia r es constante, pues todos los puntos de la esfera están a la misma distancia del origen, de modo que podemos fijar r a cualquier valor sin pérdida de generalidad, por ejemplo, $r = 1$.

Existen otras versiones de esta nomenclatura (p.e. cuando se refiere a longitud y latitud en lugar del ángulo suele utilizarse λ y δ). La nomenclatura que se utiliza en este trabajo es la que usan en [11] y [12], con una diferencia: en este trabajo se considera que en el eje de coordenadas cartesianas el cénit está en +Y (en lugar de +Z) y el plano ecuatorial lo forman los ejes X y Z (en lugar de X e Y). Esto es debido al modo en que las herramientas estándar de visualización gráfica, como es *OpenGL*, representan los escenarios tridimensionales que envuelven a un observador. En esta representación el eje Z representa delante/detrás, el eje X izquierda/derecha, y el eje Y, arriba/abajo. Asumir, pues, que el cenit está en +Y simplifica el cambio de coordenadas.

Por tanto, el cambio de coordenadas esféricas a rectangulares se realiza según la ecuación 3.1.

$$\begin{aligned} x &= r\ sin(\phi)\ sin(\theta) \\ y &= r\ cos(\phi) \\ z &= r\ sin(\phi)\ cos(\theta) \end{aligned} \tag{3.1}$$

Y el cambio de coordenadas rectangulares a esféricas se obtiene con la ecuación 3.2.

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= atan\left(\frac{x}{z}\right) \\ \phi &= acos\left(\frac{y}{r}\right) \end{aligned} \tag{3.2}$$

3.2. Presentación de un panorama esférico

Un panorama esférico se puede mostrar al usuario de diversas maneras. En este trabajo se tienen en cuenta los siguientes tres tipos de vista: panorámica, mapa de cubo y proyección esférica.

Vista panorámica. La escena completa se muestra desplegada en un área rectangular. El eje X de la imagen representa el ángulo θ, y el eje Y el ángulo ϕ (ver Figura 3.2, donde la esquina superior izquierda de la imagen se corresponde con el $(0°, 0°)$ y la inferior derecha con el $(360°, 180°)$).

Una característica de este tipo de visualización es que un gran número de líneas rectas de la escena aparecen como curvas en la imagen (ver por ejemplo las líneas de la calzada y las aceras en la figura 3.2).

Cuando la anchura de la imagen es exactamente el doble de la altura (la relación de aspecto es 2:1), también se le llama geometría equirectangular.

Mapa de cubo (*cube map texture*). La imagen panorámica se reorganiza en 6 imágenes cuadradas. Cada una muestra el resultado de la proyección de la imagen esférica en direcciones ortogonales entre sí. Las 6 vistas difieren únicamente en el eje de coordenadas utilizado como dirección de la vista. Realmente, este tipo de presentación es un tipo de textura muy utilizado en gráficos en tiempo real gracias a que está ampliamente soportado por el hardware gráfico. Es muy popular y resulta fácil encontrar en la red panoramas dispuestos como texturas de mapa de cubo.

En la figura 3.3 pueden verse las seis imágenes que conforman el mapa de cubo del panorama de la figura 3.2, y para cada una se indica la dirección de la vista correspondiente.

Proyección esférica (*skybox*). La escena se muestra parcialmente, depende de la dirección de la vista y el resultado es la proyección de la imagen esférica sobre un plano

Figura 3.2: Vista panorámica. Muestra la imagen esférica completa sobre un plano, de tal modo que cada coordenada representa un ángulo (θ y ϕ), que conjuntamente identifican un punto de la esfera en coordenadas polares (Fuente: Google Street View [2]).

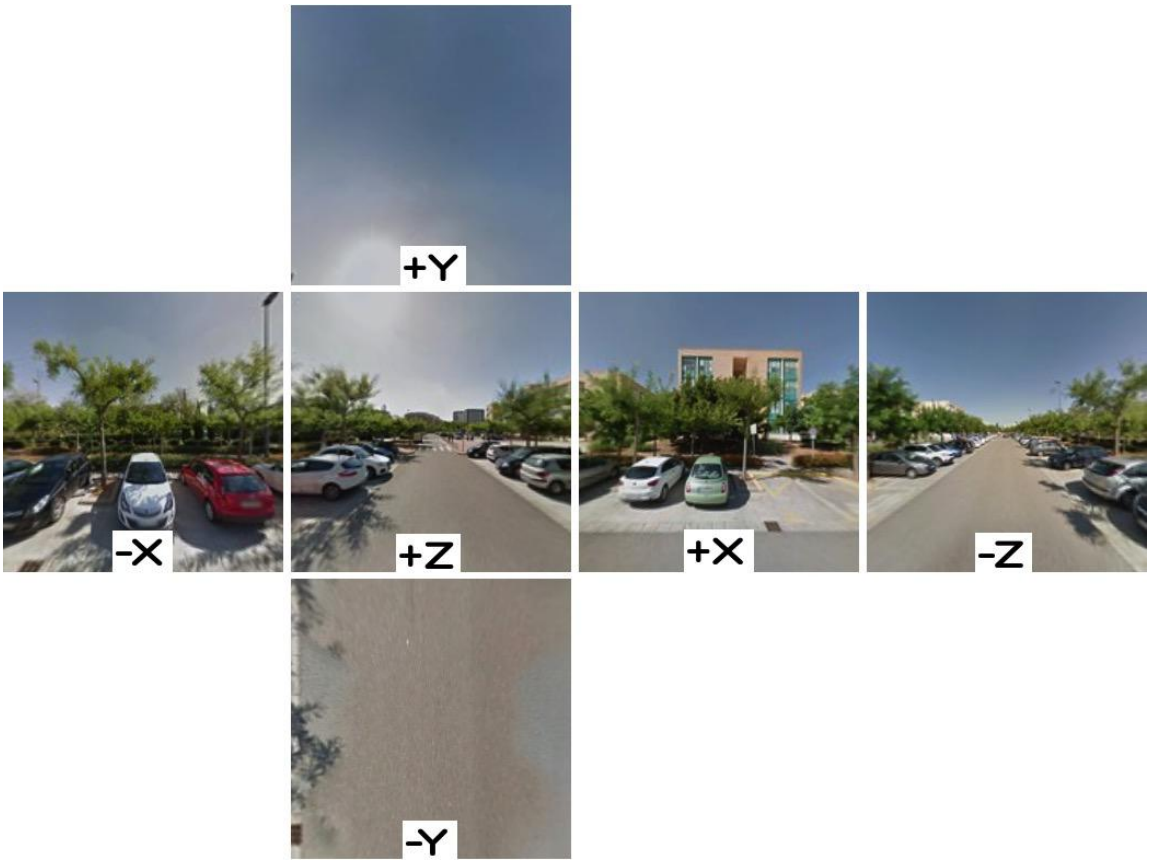


Figura 3.3: Mapa de cubo. El panorama esférico se presenta como seis imágenes, cada una resultado de la proyección sobre un plano en direcciones ortogonales entre sí.

perpendicular a dicha dirección. Este tipo de vista requiere interactuar para cambiar la dirección de la cámara y así poder observar el resto de la escena.

Se trata de un mecanismo muy utilizado en videojuegos y otras aplicaciones interactivas para dibujar el entorno lejano del observador utilizando un cubo que recibe el nombre de *skybox*.

En la figura 3.4 se muestran varios ejemplos obtenidos con diferentes direcciones de la vista. En el primer ejemplo la vista se dirige al $(1, 90°, 90°)$ y en los siguientes la dirección de la vista gira hacia la derecha y hacia abajo.

Figura 3.4: Proyección esférica. Cuatro vistas del panorama de la figura 3.2. Izquierda, arriba: vista en la dirección $(1, 90°, 90°)$, y el resto son el resultado de girar la vista un poco hacia la derecha y hacia abajo.

3.3. ¿Cómo obtener los diferentes tipos de presentación?

En la sección anterior se explicaron los tres tipos de presentación de un panorama que se van a utilizar en este trabajo: vista panorámica, mapa de cubo y proyección esférica. En Internet, es fácil encontrar tanto panoramas 360 como texturas de mapa de cubo. Sin embargo, en este trabajo se hace uso de ambos tipos de presentación por lo que se ha de poder obtener cualquiera de ellas a partir del recurso que se encuentre disponible, así

como también de generar proyecciones esféricas a partir de tanto una imagen panorámica como de una textura de mapa de cubo. Para estas operaciones se va a utilizar *OpenGL*, biblioteca estándar para gráficos en tiempo real que soporta tanto el uso de texturas 2D como el de texturas de mapa de cubo.

3.3.1. Proyección esférica

Para obtener una proyección esférica se utiliza un *Skybox*, que no es más que un cubo del cual solo se proporcionan las coordenadas de los vértices y los índices que permiten visualizarlo, por ejemplo, como una única tira de triángulos:

```
var skybox = {

  "vertices" : [-0.5, -0.5,  0.5,   // 0
                 0.5, -0.5,  0.5,   // 1
                 0.5,  0.5,  0.5,   // 2
                -0.5,  0.5,  0.5,   // 3
                 0.5, -0.5, -0.5,   // 4
                -0.5, -0.5, -0.5,   // 5
                -0.5,  0.5, -0.5,   // 6
                 0.5,  0.5, -0.5],  // 7

  "indices" : [ 3, 0, 2, 1, 7, 4, 6, 5, 3, 0,
                0, 1, 5, 4, 4, 7, 7, 6, 2, 3]

}
```

Ahora hay dos opciones posibles dependiendo de si la textura que se va a utilizar es:

- la propia vista panorámica o
- la textura de mapa de cubo.

A continuación se explica cómo proceder según el tipo de textura disponible. Pero antes, hay que señalar que a la hora de dibujar el cubo es necesario aplicar una transformación de simetría en X para observar la escena de manera correcta ya que esta se observa desde dentro del cubo hacia afuera (esto es independiente del tipo de textura que se vaya a utilizar).

A partir de un panorama

Para utilizar como textura del *Skybox* la propia imagen panorámica, es necesario obtener las coordenadas de textura $(u, v) \in [0, 1]$ por fragmento del cubo. Este cálculo se implementa directamente en el *shader*:

- En el *shader* de vértices se asignan como coordenadas de textura de cada vértice las propias coordenadas del vértice. Por lo tanto, las coordenadas de textura son en este momento coordenadas 3D.

```
in  vec3 VertexPosition;
out vec3 texCoords3d;
...
texCoords3d = VertexPosition;
```

- El procesador gráfico interpolará las coordenadas de textura especificadas en los vértices haciendo que cada fragmento reciba el resultado de la interpolación. Ya en el *shader* de fragmentos, primero se normalizan las coordenadas de textura interpoladas para que residan en la superficie de una esfera de radio unidad.

```
in vec3 texCoords3d;
...
vec3 spherePoint = normalize(texCoords3d);
```

  Después se convierten las coordenadas rectangulares en esféricas utilizando la ecuación 3.2 (con $r = 1$):

```
#define M_PI 3.141592
float zeta = atan (spherePoint.x, spherePoint.z);
float phi  = acos (spherePoint.y);
```

  Y ya solo queda normalizar el resultado en el rango $[0, 1]$:

```
float u = zeta / (2.0 * M_PI);
float v = phi  / M_PI;
```

  Una vez convertidas las coordenadas ya se puede acceder a la textura y obtener así el color del fragmento:

```
uniform sampler2D panorama;
out vec4 fragmentColor;
...
fragmentColor = texture(panorama, vec2(u, v);
```

A partir de una textura de mapa de cubo

Para obtener una proyección esférica a partir del mapa de cubo, el *shader* de vértices es igual que en el caso de la proyección esférica, es decir, para cada vértice se asignan como coordenadas de textura las propias coordenadas del vértice. Sin embargo, en el *shader* de fragmentos no es necesario que se calculen las coordenadas (u, v) ya que se puede acceder a la textura de mapa de cubo utilizando directamente las coordenadas 3d así:

```
uniform samplerCube mapaCubo;
in  vec3 texCoords3d;
out vec4 fragmentColor;
...
fragmentColor = texture(mapaCubo, texCoords3d);
```

3.3.2. Mapa de cubo a partir del panorama

La creación del mapa de cubo en este proyecto es importante por dos razones. La primera porque es una de las formas elegidas para presentar el panorama esférico al usuario y, la segunda, porque para presentar el panorama como proyección esférica mediante *OpenGL* es más eficiente utilizar una textura de mapa de cubo que directamente la imagen panorámica.

Para crear el mapa de cubo hay que generar las seis vistas de una escena formada únicamente por un *Skybox* y como textura la imagen panorámica, por lo que hay que proceder como se ha explicado en la sección 3.3.1.

Para todas ellas, se define la misma transformación de proyección que se trata de una matriz de transformación de proyección perspectiva con $fovy = 90°$, y relación de aspecto 1 ya que la imagen a obtener es cuadrada. Utilizando la función *perspective* de la librería glMatrix [13] se construye la correspondiente matriz de transformación:

```
var projectionMatrix = mat4.create();
mat4.perspective(projectionMatrix, Math.PI/2.0, 1.0, 0.1, 10.0)
```

Respecto a la transformación de la cámara, para todas las vistas la cámara estará en el origen de coordenadas y para cada vista se fijará como punto de interés un punto del eje de coordenadas correspondiente a la vista que se desea obtener. Por ejemplo, para obtener la imagen en dirección $+X$ se utiliza el punto $(1, 0, 0)$ como punto de interés y el vector $(0, 1, 0)$ como vector de inclinación, y mediante la función *lookAt* de la librería glMatrix se construye la matriz correspondiente:

```
mat4.lookAt(cameraMatrix, [0,0,0], [1,0,0], [0,1,0]);
```

Para las seis vistas, estos son los respectivos puntos de interés, i, y vectores de inclinación, Up:

```
+X: i = [ 1, 0, 0], Up = [ 0, 1, 0]
-X: i = [-1, 0, 0], Up = [ 0, 1, 0]
+Z: i = [ 0, 0, 1], Up = [ 0, 1, 0]
-Z: i = [ 0, 0,-1], Up = [ 0, 1, 0]
+Y: i = [ 0, 1, 0], Up = [ 0, 0,-1]
-Y: i = [ 0,-1, 0], Up = [ 0, 0, 1]
```

Recordar también que, a la hora de dibujar el *skybox*, sigue siendo necesario aplicar una transformación de simetría en X para observar la escena de manera correcta ya que esta se observa desde dentro del cubo hacia afuera. De esta forma, la matriz de transformación M que se aplica a cada vértice del cubo se calcula de la siguiente forma:

```
var symInX = mat4.fromScaling (mat4.create(), [-1,1,1]);
mat4.multiply (modelViewMatrix, cameraMatrix, symInX);
mat4.multiply (M, projectionMatrix, modelViewMatrix);
```

Para este trabajo se ha utilizado la implementación del conversor de panorama a mapa de cubo que se puede encontrar en [14]. Donde se carga el panorama y se descarga el mapa de cubo como seis imágenes. Por ejemplo, la figura 3.5 muestra varios ejemplos de panoramas y las respectivas texturas de mapa de cubo creadas todas ellas con esta herramienta.

3.3.3. Panorama a partir de un mapa de cubo

Para obtener el panorama procede dibujar un rectángulo del tamaño del panorama. Para cada fragmento del rectángulo, se convierten sus coordenadas de textura (que varían entre 0 y 1) en *zeta* y *phi* así:

```
#define M_PI 3.141592
in vec2 texCoords;
...
float zeta = 2.0 * M_PI * texCoords.s;  // 0..360
float phi  =       M_PI * texCoords.t;  // 0..180
```

Figura 3.5: Ejemplos de panoramas y las texturas de mapa de cubo resultantes. Fuentes: la tercera, Emil Persson (Humus) [15] y el resto, Google Street View [2].

Después se transforman las coordenadas esféricas a rectangulares según la ecuación 3.1:

```
float x = sin(phi) * sin(zeta);
float y = cos(phi);
float z = sin(phi) * cos(zeta);
```

Y ahora se utilizan las coordenadas rectangulares para acceder a la textura de mapa de cubo y pintar el fragmento con el color devuelto:

```
uniform samplerCube mapaCubo;
out vec4 fragmentColor;
...
vec3 texCoordsCube = vec3 (x, y, z);
fragmentColor = texture(mapaCubo, texCoordsCube);
```

En este trabajo, se utilizó el conversor de mapa de cubo a panorama esférico que se puede encontrar en [16]. En este conversor, se cargan las seis imágenes que componen la textura del mapa de cubo y permite descargar la imagen correspondiente al panorama esférico. Por ejemplo, la figura 3.6 muestra varios ejemplos de texturas de mapa de cubo [15] y los respectivos panoramas.

3.4. Señalando un punto en la vista del *skybox*

A partir de un clic realizado sobre el canvas del *skybox* hemos de averiguar de qué punto de la superficie de la esfera centrada en el origen de coordenadas y radio unidad se trata.

Sea el punto p el punto señalado por el usuario en coordenadas de pantalla (en píxeles). Asumimos que el origen de coordenadas es el centro del canvas. En $Z = -1$ (cabe recordar que en el sistema gráfico la cámara está en el origen de coordenadas mirando en dirección $-Z$), la altura h que hay desde el centro del canvas hasta el tope superior depende del *fovy* y lo calculamos así:

$$h = tan\left(\frac{fovy}{2}\right) \tag{3.3}$$

Si *width* y *height* son el ancho y el alto del canvas en píxeles respectivamente, el punto 3D, q, lo obtenemos a partir de p así:

Figura 3.6: Ejemplos de texturas de mapa de cubo (Fuente: Emil Persson (Humus) [15]) y los panoramas esféricos resultantes.

$$\begin{aligned} q_x &= \frac{4\ h\ p_x}{width} \\ q_y &= \frac{2\ h\ p_y}{height} \\ q_z &= -1 \end{aligned} \tag{3.4}$$

Y para que q esté a una distancia de 1 del origen lo normalizamos:

$$q_n = normalize(q) \tag{3.5}$$

Sin embargo, el usuario puede interactuar con el *Skybox* y haberlo girado un ángulo *zeta* alrededor del eje Y, $R_Y(zeta)$, y un ángulo *phi* alrededor del eje X, $R_X(phi)$. Esto significa que cuando el usuario seleccionó el punto p al *Skybox* se le había aplicado una matriz de transformación R, tal que:

$$R = R_X(phi)R_Y(zeta) \tag{3.6}$$

Por lo tanto, el punto buscado q_f se obtendrá tras aplicar a q_n la transformación contraria aplicada sobre el *skybox*, es decir:

$$q_f = R_Y(-zeta)R_X(-phi)q_n \tag{3.7}$$

El mecanismo que se ha descrito en esta sección es igualmente válido en el caso de que el usuario interactúe con el mapa de cubo ya que esta textura se corresponde con seis vistas del *skybox* cada una obtenida en dirección de cada uno de los ejes de coordenadas:

$$q_f = R_{axis}(angle)q_n \tag{3.8}$$

donde $axis$ es el eje Y en el caso de las vistas obtenidas en dirección $\pm X$ y $\pm Z$, y X para las vistas en dirección $\pm Y$, y $angle$ es para la vista en dirección:

- $+X$: $+90°$
- $-X$: $-90°$
- $+Z$: $+180°$
- $-Z$: $+0°$
- $+Y$: $+90°$
- $-Y$: $-90°$

3.5. Interacción del usuario

El método de estimación de profundidad, necesita que el usuario aporte información sobre la profundidad de la imagen para empezar a resolverla. Esta información se basa en señalar los puntos que se encuentran más cerca o más lejos, pares de puntos que se encuentra a la misma o a distinta profundidad, e información sobre el plano del suelo.

Para introducir esta información, se ha desarrollado una aplicación Web sobre la que el usuario puede interactuar con el panorama. La aplicación muestra los tres tipos de visualización de un panorama al mismo tiempo. En la parte superior se encuentran la imagen panorámica y la proyección esférica, y en la parte inferior la vista de mapa de cubo y las herramientas para la interacción (ver Figura 3.7).

Figura 3.7: Vista de la aplicación Web para la interacción con los panoramas.

Se puede añadir información de profundidad desde cualquier tipo de visualización. Basta con seleccionar el tipo de información que se va a añadir y hacer clic con el botón izquierdo sobre cualquiera de las tres vistas.

En el caso de la proyección esférica, además, permite desplazarse por ella dejando el ratón sobre los lados del *canvas*, este desplazamiento es más rápido conforme el ratón se acerca a los extremos. De esta manera, los clics solo son necesarios para introducir información, quedando una interacción más consistente entre los tres tipos de vista.

El usuario puede aportar información de profundidad de la imagen a través de cinco tipos de entrada, realizando clic con el botón izquierdo del ratón sobre cualquiera de las tres vistas:

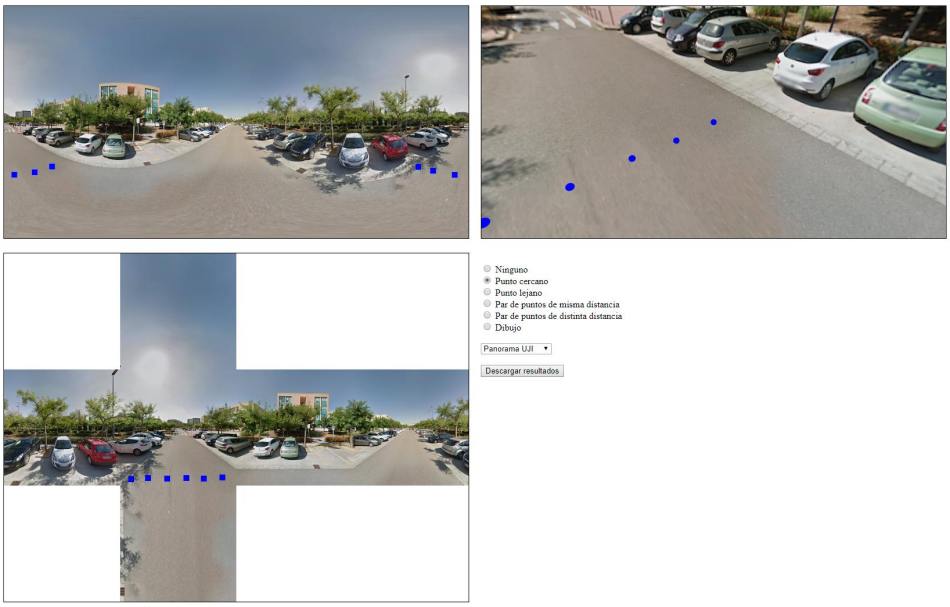


Figura 3.8: Puntos cercanos.

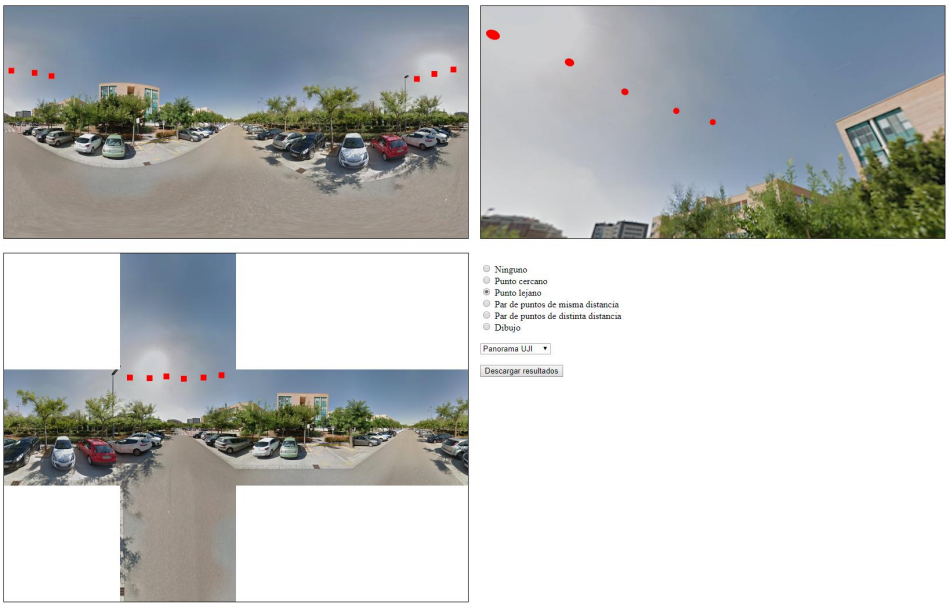


Figura 3.9: Puntos lejanos.

- Punto cercano (ver Figura 3.8), marca el punto más cercano de la escena con un punto azul. Se aconseja marcar más de un punto.
- Punto lejano (ver Figura 3.9), marca el punto más lejano de la escena con un punto rojo. Se aconseja marcar más de un punto.
- Pares de puntos a la misma distancia (ver Figura 3.10), marca dos puntos que se encuentran a la misma profundidad con puntos verdes y los une con una linea verde, para no confundir los puntos marcados.
- Pares de puntos a distinta distancia (ver Figura 3.11), marca dos puntos que se encuentran a distinta distancia. Primero se marca el punto más cercano con un punto amarillo, y después se marca el punto más lejano con un punto verde. Estos pares de puntos también están unidos por una línea verde.
- Trazo de puntos para marcar el suelo (ver Figura 3.12), en esta ocasión hay que hacer clic y sin soltar, arrastrar para marcar el trazo de color rojo.

Figura 3.10: Pares de puntos misma distancia.

Cada vez que se incluye un punto nuevo, este parpadea en las tres vistas, para que el usuario pueda ver lo que acaba de añadir fácilmente. Además, en la proyección esférica, el punto de vista cambia de manera automática para que se vea siempre el último punto marcado.

A la hora de marcar varios puntos, hay que tener cuidado si se marca el mismo punto con diferentes tipos de entrada, ya que, esto puede causar que el método de estimación de profundidad no encuentre una solución, al tener asignados dos valores distintos para un mismo píxel.

Figura 3.11: Pares de puntos distinta distancia.

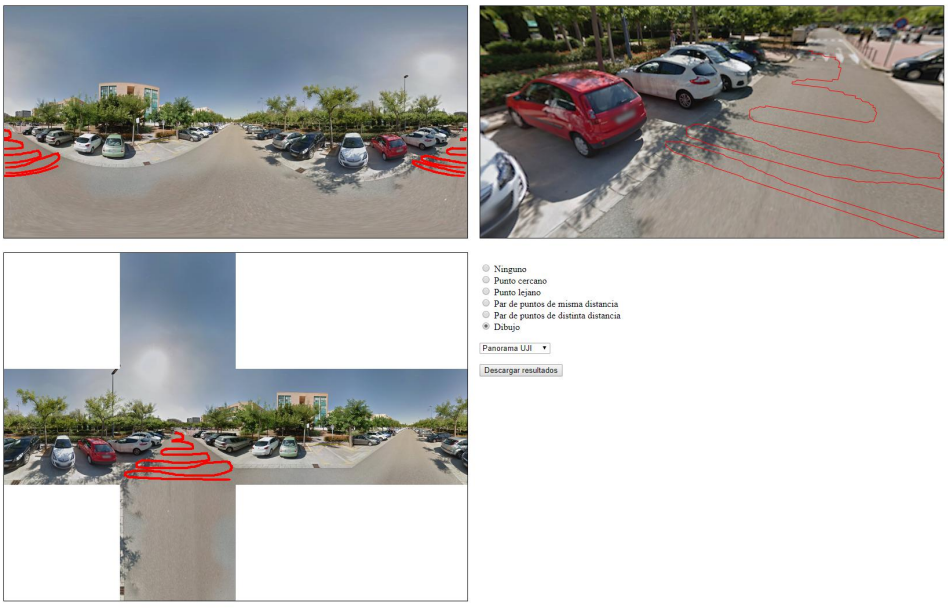


Figura 3.12: Trazo de puntos del suelo.

La información proporcionada por el usuario se almacena en el vector correspondiente al tipo de entrada que se utilice. Las coordenadas que se guardan corresponden a las de la imagen del mapa de cubo con la que se esté interactuando, de esta manera el algoritmo del método de estimación de profundidad no debe realizar transformaciones de coordenadas.

Además, para poder mostrar las interacciones en la proyección esférica, se ha creado una primitiva que aumenta el valor de sus índices y las coordenadas de los vértices, cada vez que se añade nueva información. Esto, es especialmente útil a la hora de mostrar el dibujo que marca el suelo, ya que en lugar de crear nuevas primitivas que vayan de un punto a otro, se crea una única primitiva que aumenta su número de vértices conforme dibujamos, consiguiendo que no dupliquemos información de los puntos y el movimiento de la cámara sea más fluido.

Con todos los puntos marcados, al pulsar el botón *Descargar resultados* (ver Figura 3.7) se descargan cuatro ficheros con la información en formato ASCII para que se puedan leer desde otra aplicación, en este caso Matlab.

- *inputData_seeds.mat*, en el que se almacenan las coordenadas X e Y, y la profundidad de los puntos cercanos (240) y lejanos (0). *(X,Y,Profundidad)*

```
309,279,240
...
307,126,0
...
```

- *inputData_pairs.mat*, para los pares de puntos de misma profundidad, donde se almacenan las coordenadas X e Y de los dos puntos. *(X1,Y1,X2,Y2)*

```
468,228,466,247
450,221,448,232
...
```

- *inputData_separators.mat*, para los pares de puntos de distinta profundidad, donde se almacenan las coordenadas X e Y del punto más cercano y después del más lejano. *(X1,Y1,X2,Y2)*

```
333,225,344,216
350,220,358,216
...
```

- *inputData_groundPoints.mat*, para el vector, donde se almacenan las coordenadas X e Y pertenecientes a los puntos del plano del suelo de la imagen.*(X,Y)*

```
241,252
242,252
243,251
245,251
...
```

3.6. Análisis de la interacción

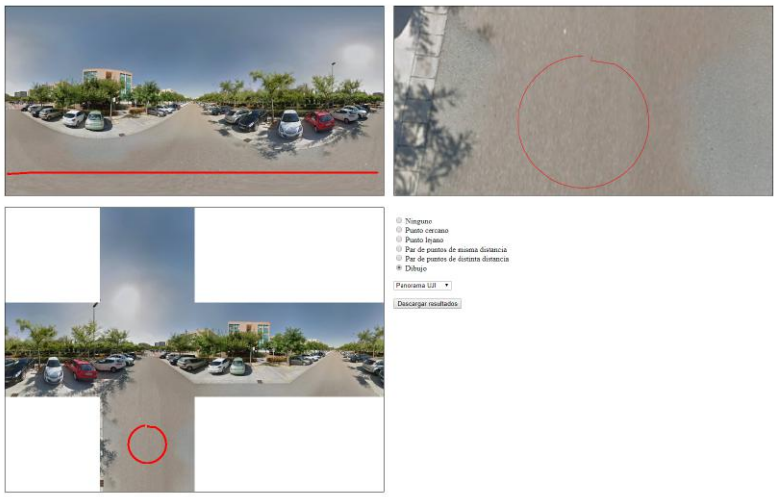
Tras realizar varias pruebas, se llegó a las siguientes conclusiones sobre lo que aportaba cada vista.

Vista panorámica En un primer contacto, puede llegar a ser la vista más útil, ya que es capaz de mostrar el panorama de una manera más continua/conectada. Pero, por el contrario, es la vista que más deforma la imagen, creando así sensaciones de profundidad erróneas. Entre estas deformaciones están:

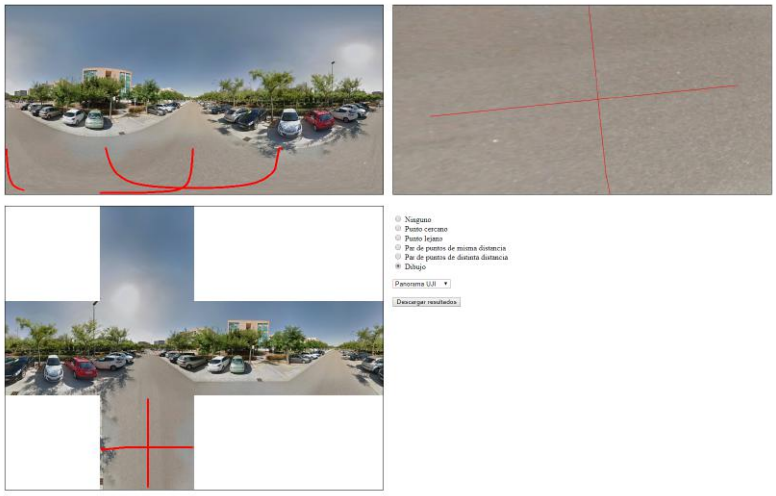
- Al dibujar una línea recta en la vista panorámica, esta se muestra como una curva en las demás vistas. En el caso de dibujar una fila entera del panorama, sería un círculo en la proyección esférica y en algunos casos, también en el mapa de cubo, tal y como se ve en el ejemplo (ver Figura 3.13(a)).
- Una recta en la proyección esférica, corresponde a una curva en el panorama. En el ejemplo se puede ver dos líneas rectas en forma de cruz que pasan a ser dos líneas curvas en el panorama (ver Figura 3.13(b)).
- En el caso de dibujar una fila entera del mapa de cubo, en la vista panorámica queda como una línea ondulada y en la proyección esférica se puede apreciar la forma del cubo que contiene la textura (ver Figura 3.13(c)).

Estas deformaciones dificultan la interacción del usuario en este tipo de vista, ya que, cuesta distinguir dos puntos que estén a la misma profundidad.

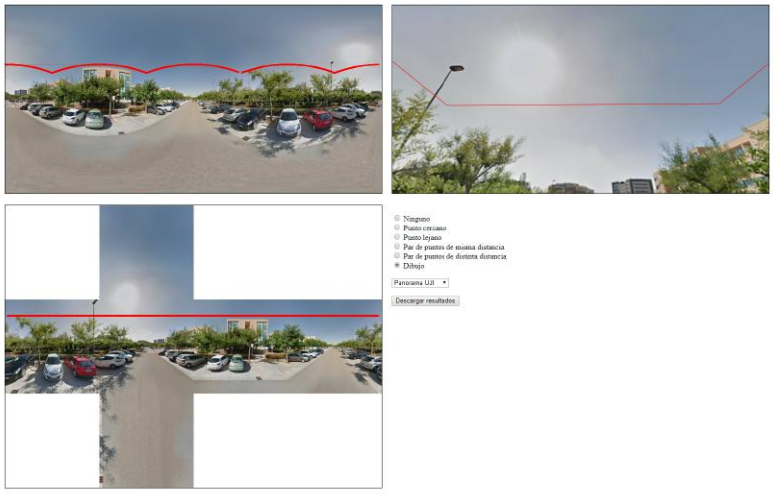
Mapa de cubo De esta vista, se consideró inicialmente, que podía presentar la desventaja de mostrar la imagen como un cubo desplegado, separando algunos puntos que son contiguos en el panorama, apareciendo estos en distintas caras. Esto, puede dificultar la introducción de información de profundidad de elementos que se encuentren en distintas caras. Pero sin embargo, como las caras del mapa de cubo son imágenes en perspectiva, permite definir las entradas de información de forma más natural, haciendo que sea la vista más cómoda.



(a) Una línea recta en la vista panorámica corresponde a una curva en las otras vistas.



(b) Una línea recta en la proyección esférica corresponde a una curva en el panorama.



(c) Una recta en el mapa de cubo produce un efecto de ondulación en el panorama y una línea poligonal en la proyección.

Figura 3.13: Deformación de las líneas.

Proyección esférica Esta vista puede ser de gran ayuda para entender mejor el panorama, ya que pretende simular cómo vería el panorama el usuario si se encontrara allí, mostrando la escena de manera más natural. Además, muestra el panorama más ampliado, pudiendo servir como zoom para ver los detalles de la imagen. Pero tiene el inconveniente de que no se puede mostrar todo el panorama al mismo tiempo y se necesita interacción para poder desplazarse por la vista.

Aunque gracias a esta aplicación el usuario puede interactuar con cualquiera de las vistas, se decidió utilizar el mapa de cubo en la siguiente etapa donde se estima la profundidad del panorama. Este es el motivo por el que la descarga se realiza en coordenadas de las distintas vistas del mapa de cubo. No obstante también se podría realizar la descarga en coordenadas esféricas de manera directa, puesto que estas se manejan en todo momento en la aplicación.

Capítulo 4

Estimación de la profundidad

4.1. Extensión del método a un mapa de cubo

La extensión al método de estimación de profundidades que se ha mencionado en el capítulo de trabajo previo [17], plantea varios inconvenientes a la hora de utilizar el método con imágenes de panoramas esféricos. Por una parte, en un panorama esférico la conectividad entre los píxeles es diferente que en las imágenes en perspectiva. La última columna está conectada con la primera columna y la primera fila conecta en un único punto (el cenit) y la última fila también (el nadir). Por tanto, hay zonas de la esfera más densamente representadas que otras. Estas diferencias se deberían tener en cuenta a la hora de propagar valores de profundidad a los píxeles contiguos. Por otra parte, las imágenes de panoramas esféricos crean deformaciones respecto a una imagen en perspectiva, por ejemplo, creando curvas donde hay líneas. Estas diferencias entre el panorama esférico y la imagen en perspectiva, implicarían tener que modificar las ecuaciones en las que se basa el método de estimación de profundidad, para adaptarlas a las coordenadas esféricas.

Para poder utilizar el algoritmo sobre imágenes panorámicas, se ha decidido utilizar las imágenes de mapa de cubo, ya que, cada cara del cubo es una imagen en perspectiva (ver Figura 4.1). Además, se ha realizado una modificación para compartir la información de profundidad que se ha obtenido en cada cara. Este proceso se explica a continuación.

Primero se procesará la imagen frontal ($+Z$), a partir de la información que el usuario ha marcado. El siguiente paso es analizar la imagen de la derecha ($+X$), en la que además de utilizar la información del usuario, también se utiliza la información de los píxeles de la última columna de la imagen frontal (ver Figura 4.2). Los píxeles de la última columna de $+Z$, marcados en verde, tienen como vecinos los píxeles de la primera columna de la imagen que está a su derecha, de modo que a estos se les otorgará la misma profundidad. Este proceso se repite para la imagen trasera ($-Z$), pero utilizando la información de la

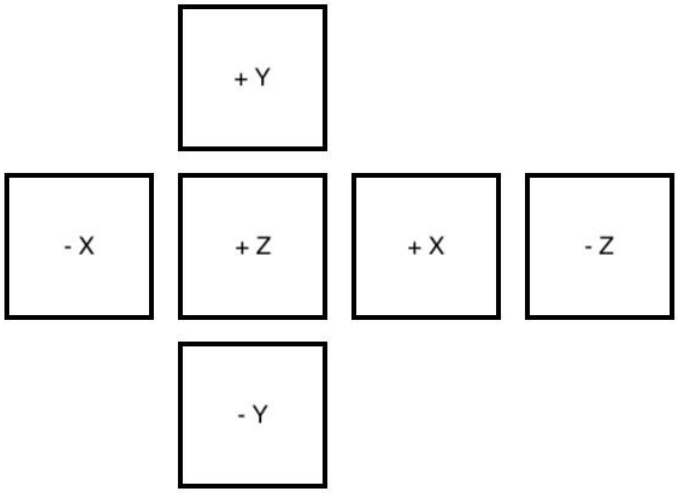


Figura 4.1: Vistas del mapa de cubo: izquierda (-X), frontal ($+Z$), derecha ($+X$), trasera (-Z), arriba ($+Y$) y abajo (-Y).

última columna de la imagen de la derecha ($+X$), marcados en azul en la figura.

Para la imagen de la izquierda (-X), además de la información añadida por el usuario y de la información de la primera columna calculada anteriormente (-Z), marcada en rojo en la figura 4.2, también se tiene la información de la última columna (marcada en color naranja), ya que esta es vecina de la primera columna de la imagen frontal ($+Z$).

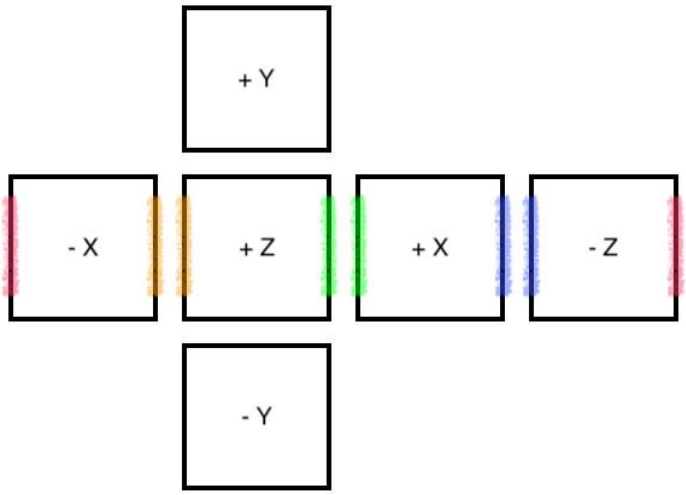


Figura 4.2: Propagación de la información de profundidad en el eje horizontal.

Una vez calculadas las imágenes de las caras del horizonte, se calcula la cara inferior (-Y) y la cara superior ($+Y$) (ver Figura 4.3). Estas caras tienen la particularidad de que, cada lado de estas, tiene la información de las demás caras vecinas ya calculadas. Sin embargo, hay casos en los que se debe tener en cuenta la orientación de la cara vecina.

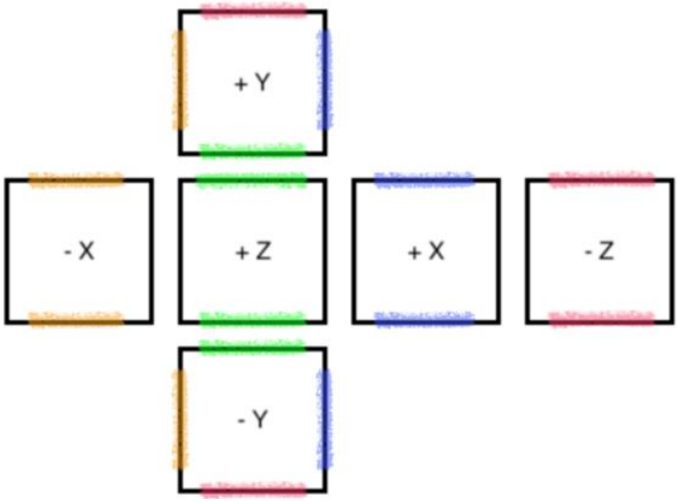


Figura 4.3: Propagación de la información de profundidad en el eje vertical.

Para calcular los valores de profundidad de la imagen inferior ($-Y$), se propagan los valores de la siguiente manera (ver Figura 4.4):

- Los valores de la primera fila se obtienen de los valores de la última fila de la imagen frontal ($+Z$).
- Los valores de la última columna se obtienen de los valores de la última fila de la imagen derecha ($+X$).
- Los valores de la primera columna corresponden a los valores de la última fila de la imagen izquierda ($-X$) pero en el orden contrario, ya que el primer píxel de la imagen inferior ($-Y$) es el vecino del último píxel de la imagen izquierda ($-X$).
- Lo mismo ocurre para la última fila, a la cual se le añaden los valores de la última fila de la imagen trasera ($-Z$) con el orden invertido.

Análogamente, en el caso de la cara superior ($+Y$) (ver Figura 4.5), la primera columna corresponde a la última columna de la imagen izquierda ($-X$) y la última fila a la primera fila de la imagen frontal ($+Z$). Los valores de la primera fila, corresponden a los valores de la primera fila de la imagen trasera ($-Z$), pero estos se deben añadir empezando por el último valor y acabando por el primero, este mismo proceso se realiza con los valores de la última columna correspondientes a los valores de la primera fila de la imagen derecha ($+X$).

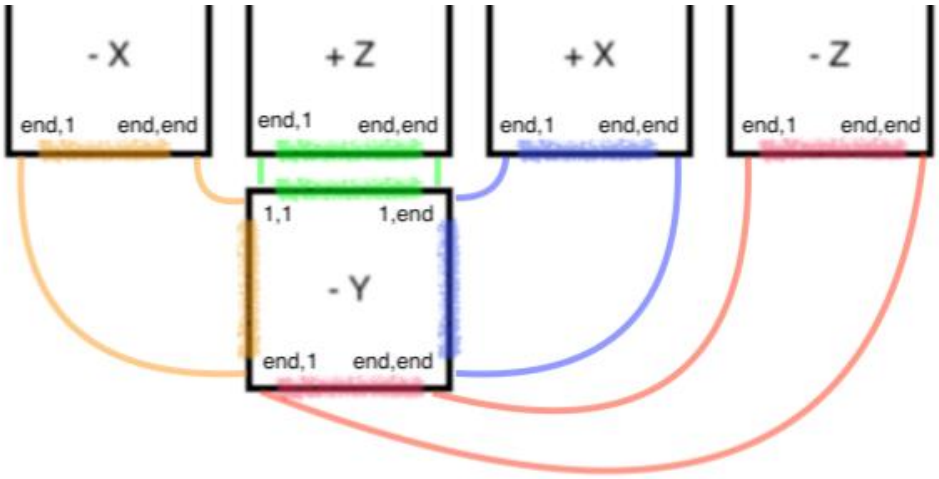


Figura 4.4: Propagación de la información de profundidad en la cara inferior.

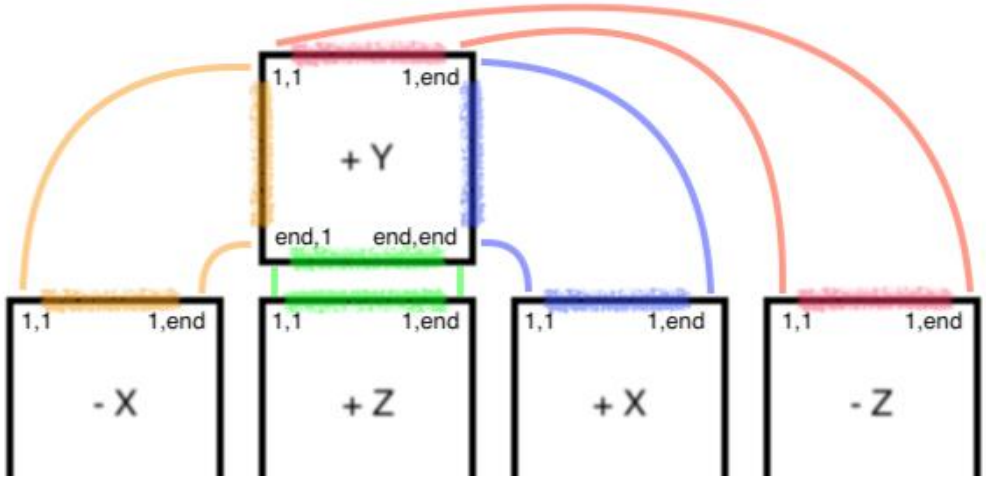


Figura 4.5: Propagación de la información de profundidad en la cara superior.

4.2. Obtención del mapa de profundidad

A continuación, se muestra el proceso de estimación del mapa de profundidad utilizando un panorama de la UJI, obtenido de Google Street View [2] (ver Figura 3.2).

La primera cara del mapa de cubo es la frontal ($+Z$). En esta imagen se puede ver unos edificios al fondo del horizonte, varios coches aparcados y la carretera que converge hacia la mitad de la imagen (ver Figura 4.6).

Figura 4.6: Cara frontal del panorama de la UJI.

Lo primero será marcar los puntos próximos y lejanos, en esta ocasión, los puntos más cercanos corresponden al trozo de carretera que se encuentra en la parte de debajo de la imagen, y los puntos más lejanos se encuentran en la parte superior de la imagen y corresponden al cielo.

Después se indican los puntos de misma distancia, se han añadido en los coches para que mantenga cada coche con la misma distancia. También se añaden algunos puntos de distinta distancia, para separar los coches y para separar los edificios que están al fondo del cielo.

Por último, queda indicar qué zona corresponde al suelo de la imagen, trazando de manera sinusoidal la línea que va desde los puntos más cercanos hasta el horizonte donde converge la carretera (ver Figura 4.7).

Una vez marcados todos los puntos de interés, se calcula la profundidad de la imagen frontal (+Z), ya que esta no necesita información de las demás caras del mapa de cubo.

En la imagen de profundidad obtenida (ver Figura 4.8), se puede comprobar cómo los valores de la carretera disminuyen conforme esta converge en el horizonte. También, se pueden distinguir los coches, cada uno con un valor de profundidad distinto. Y en el horizonte aparecen los edificios separados del fondo.

El siguiente paso, es acabar de marcar todos los puntos de interés en las demás caras para continuar con el proceso (ver Figura 4.9).

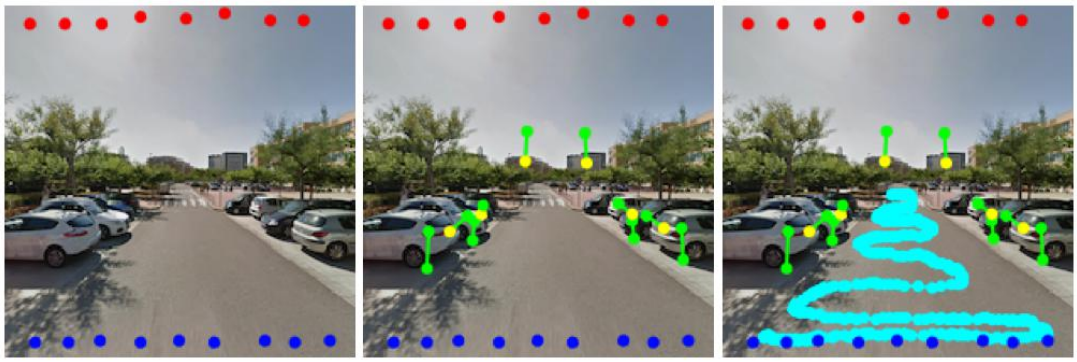


Figura 4.7: Entrada del usuario para la cara frontal del panorama de la UJI. Distintos momentos a lo largo de la interacción del usuario.

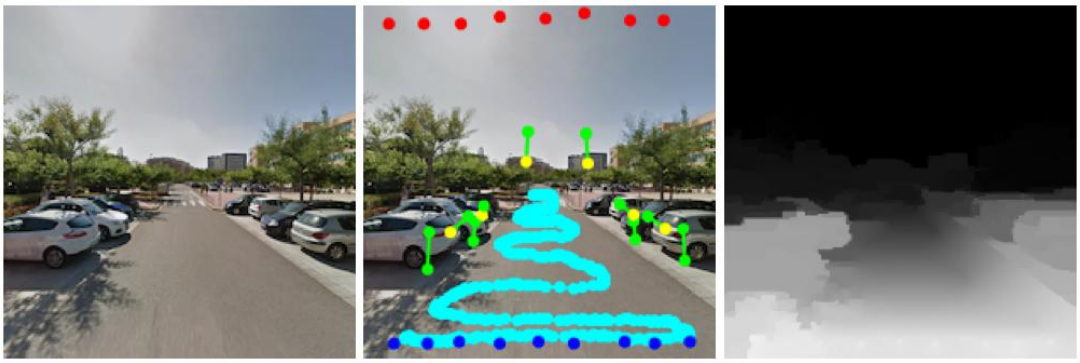


Figura 4.8: Datos de entrada y resultados obtenidos para la cara frontal del panorama de la UJI.

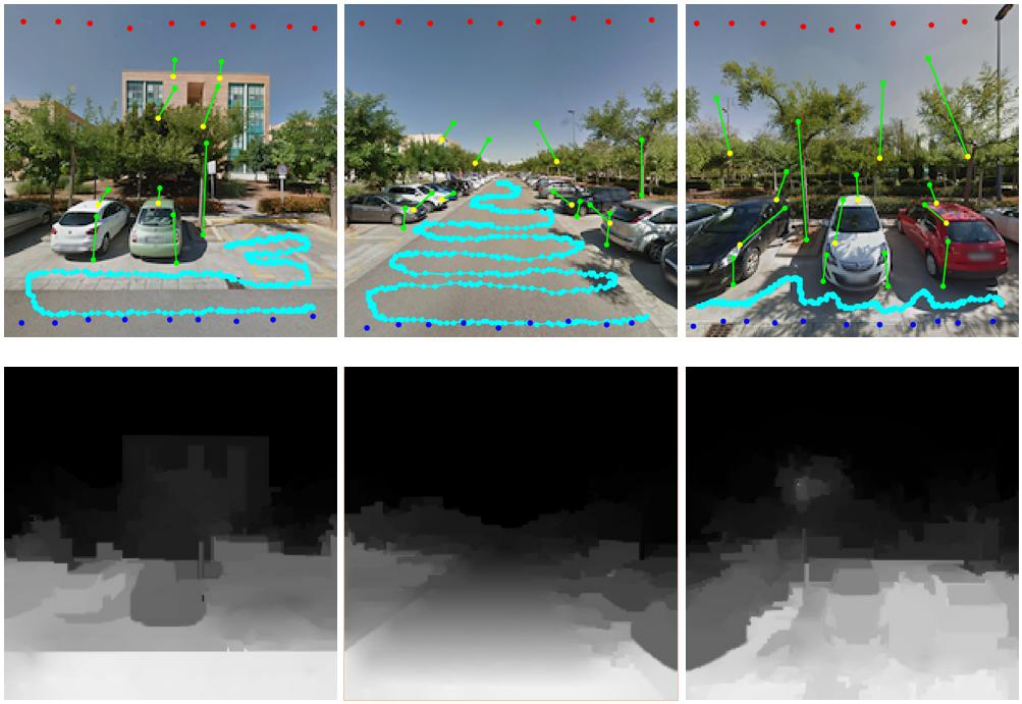


Figura 4.9: Entrada del usuario para las demás caras en el eje horizontal del panorama de la UJI y mapas de profundidad obtenidos.

En el caso de las caras superior (+Y) e inferior (-Y), no es necesario agregar información, ya que, en las imágenes solo se ve cielo o suelo y con la información que proporcionan las caras vecinas bastará para obtener la profundidad.

Por último, se transforman las imágenes de mapa de cubo a una sola imagen panorámica (ver Figura 4.10). En esta imagen, se puede observar que los valores de profundidad del suelo disminuyen conforme se aleja, y que cada coche aparcado, tiene uno o dos valores de profundidad y suelen diferenciarse unos de otros.

Estos valores de profundidad, equivalen a la coordenada z en un sistema de coordenadas *(x,y,z)*. Pero estas coordenadas no tienen en cuenta la posición del usuario en el caso de mostrar el panorama en una proyección esférica. Esto se puede apreciar mejor en objetos altos, como edificios o árboles, los cuales tienen el mismo valor en el panorama de profundidad, mientras que, la distancia a la que se encuentra el usuario es mayor cuanto más alto sea el objeto.

Por esto, se define el panorama de distancia (ver Figura 4.11), que representa la distancia a la que se encuentra el píxel respecto al centro del punto de vista en una proyección esférica. Al igual que en el caso de las profundidades, el valor de distancia se considera de manera inversa: el valor es mayor para distancias cortas y menor para distancias grandes.

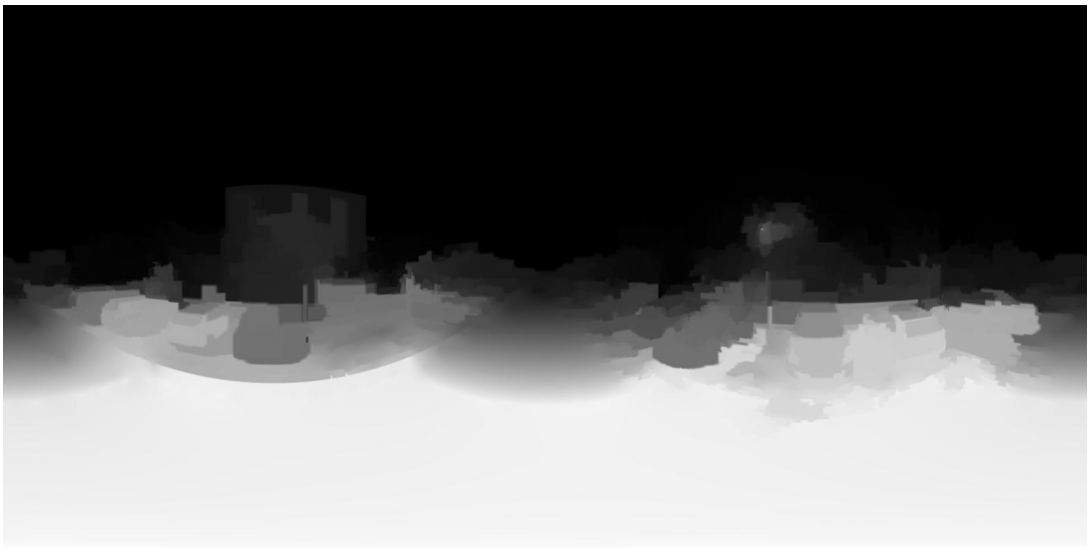


Figura 4.10: Imagen panorámica con la información de profundidad.

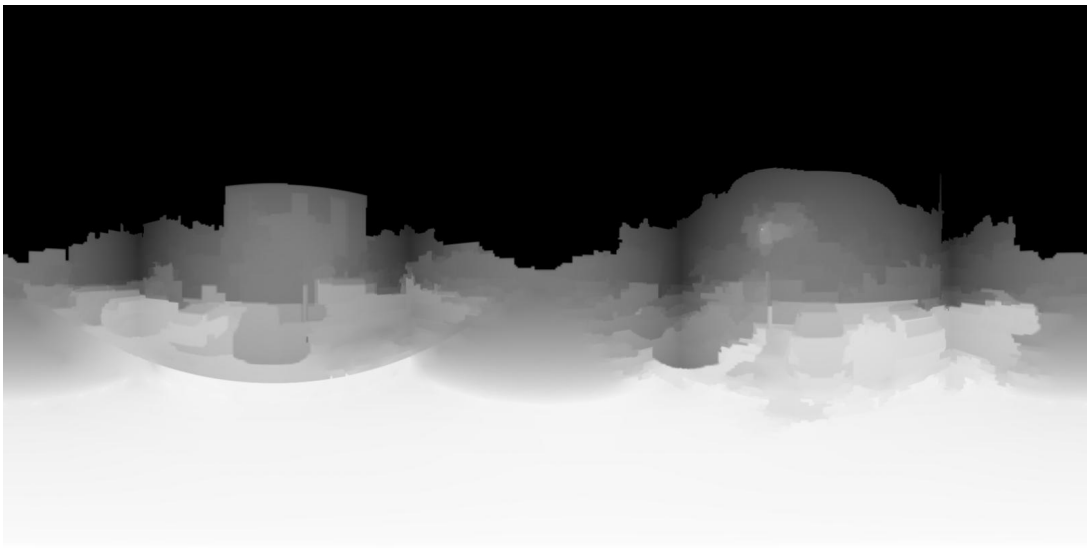


Figura 4.11: Imagen panorámica con la información de distancia.

4.3. Resultados para algunos panoramas de ejemplo

4.3.1. San Francisco

En este panorama (ver Figura 4.12) se ha introducido información de pares de puntos a la misma profundidad en las palmeras, y además, en algunos grupos de gente pares de puntos de distinta profundidad para destacarlos del plano del suelo.

En la figura 4.13 se muestra el resultado obtenido. La entrada de datos se muestra en la figura 4.14 y el resultado para cada una de las caras en la figura 4.15.

Figura 4.12: Panorama de San Francisco (Fuente: Emil Persson (Humus) [15]).

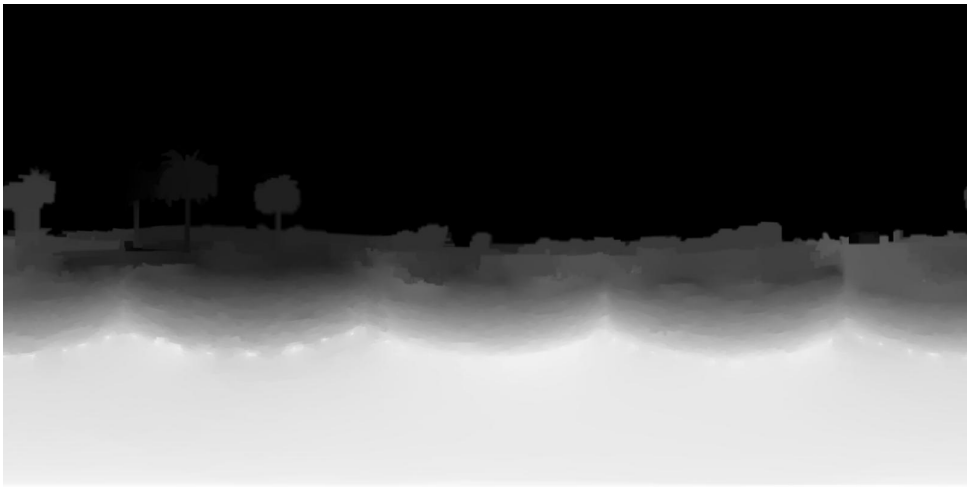


Figura 4.13: Mapa de profundidad del panorama San Francisco.

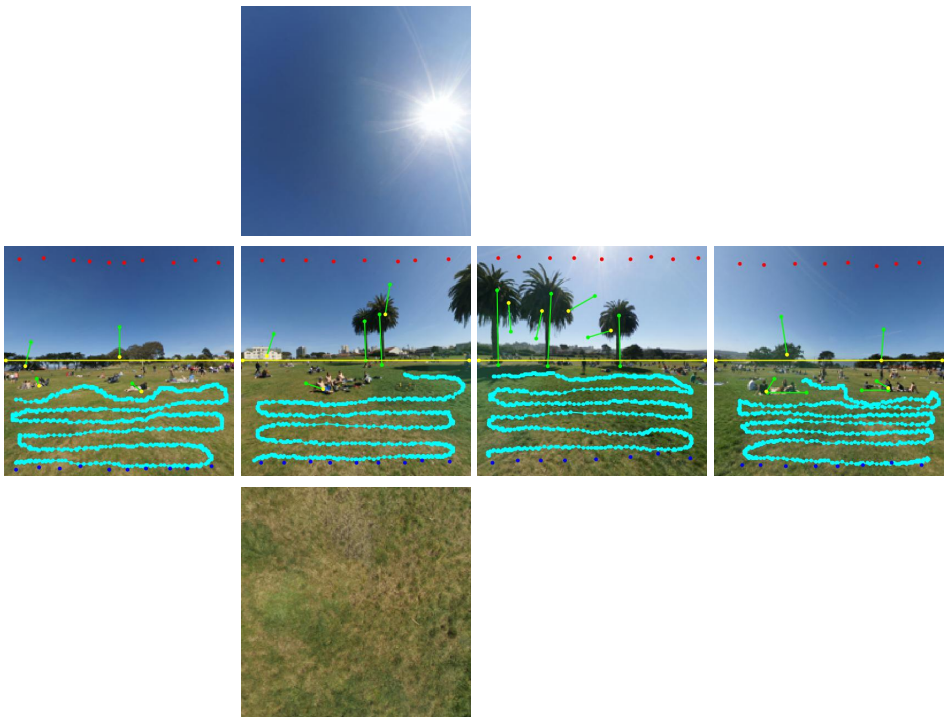


Figura 4.14: Entrada del usuario para el panorama San Francisco

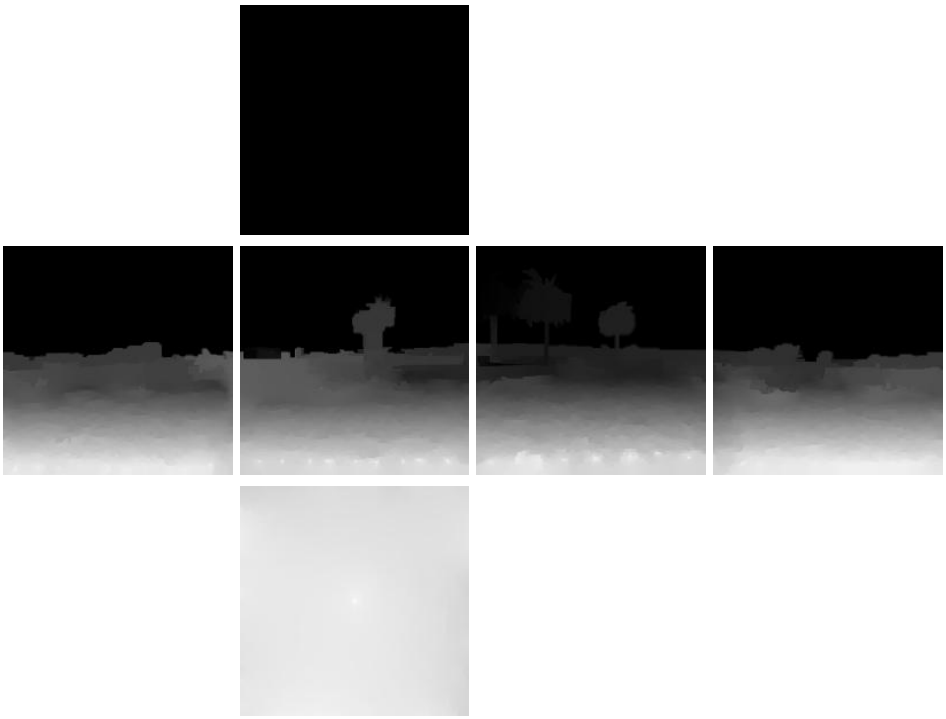


Figura 4.15: Mapa de cubo de profundidad del panorama San Francisco.

4.3.2. Skansen

En este panorama (ver Figura 4.16) se puede observar un edificio que se encuentra más elevado que el plano del suelo y varios árboles. Por lo que se introducirán pares de puntos de distinta profundidad en la escalera, para indicar la elevación. También se introducirán pares de puntos de misma profundidad en la base y en la copa de algunos árboles y pares de puntos de distinta profundidad entre ellos, para conseguir algo de separación entre ellos.

En la figura 4.17 se muestra el resultado obtenido. La entrada de datos se muestra en la figura 4.18 y el resultado para cada una de las caras en la figura 4.19.

Figura 4.16: Panorama Skansen (Fuente: Emil Persson (Humus) [15]).

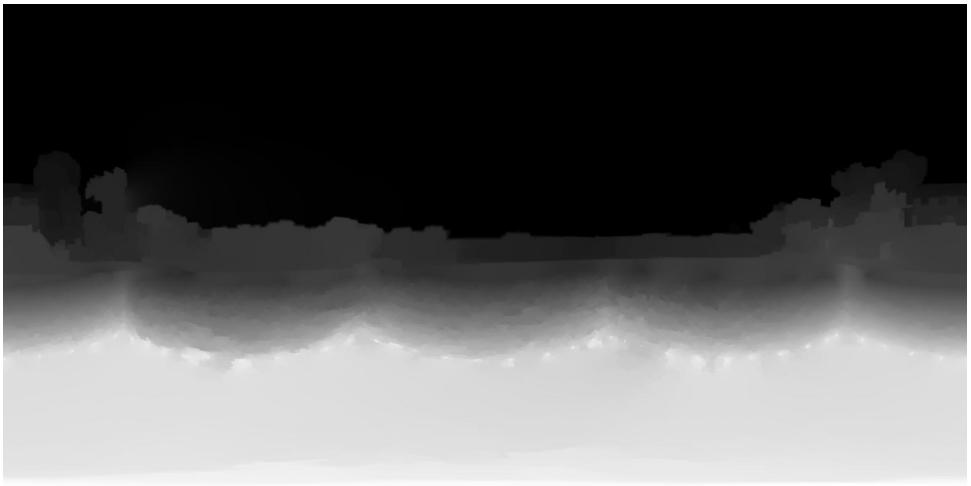


Figura 4.17: Mapa de profundidad del panorama Skansen.

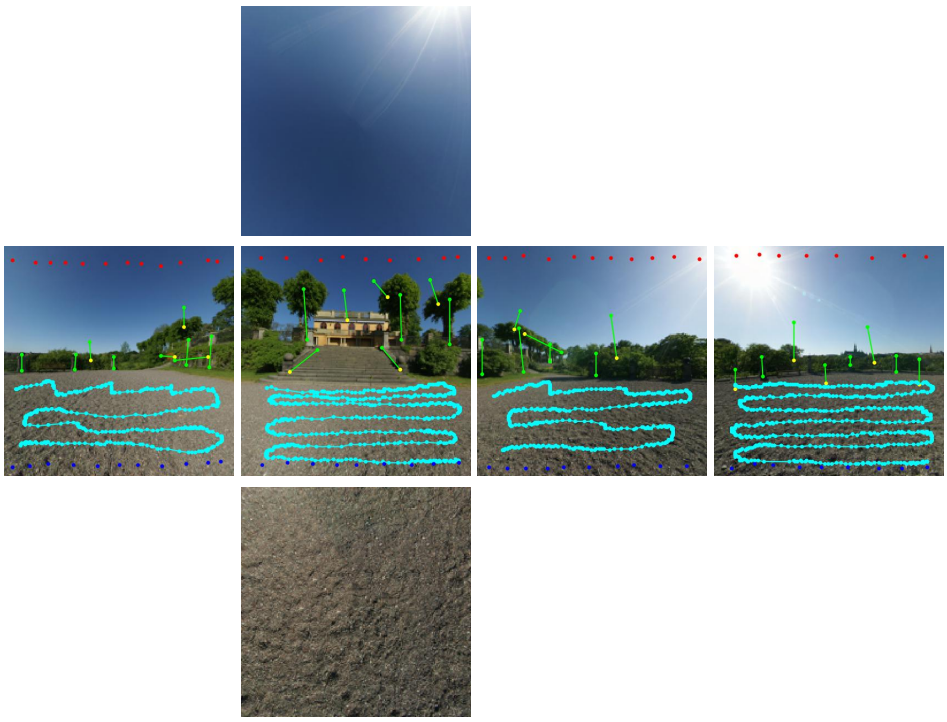


Figura 4.18: Entrada del usuario para el panorama Skansen.

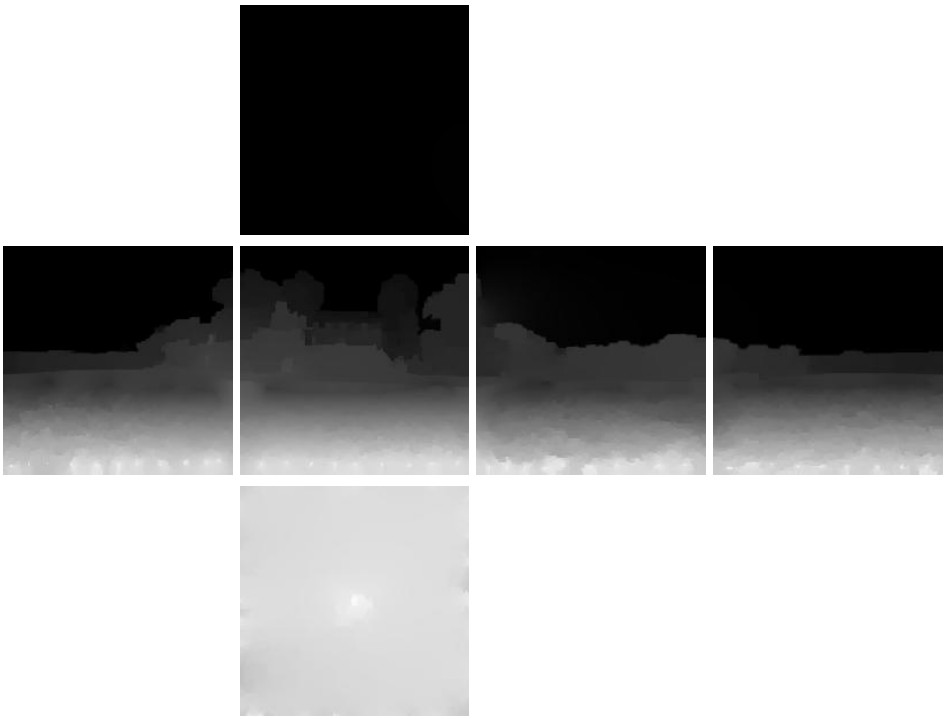


Figura 4.19: Mapa de cubo de profundidad del panorama Skansen.

4.3.3. Tantolunden

En este panorama (ver Figura 4.20) lo que más destaca es el árbol que sobresale de la isla, por lo que se introducirá información de profundidad utilizando pares de puntos de misma y distinta profundidad para conseguir este efecto.

En la figura 4.21 se muestra el resultado obtenido. La entrada de datos se muestra en la figura 4.22 y el resultado para cada una de las caras en la figura 4.23.

Figura 4.20: Panorama Tantolunden (Fuente: Emil Persson (Humus) [15]).

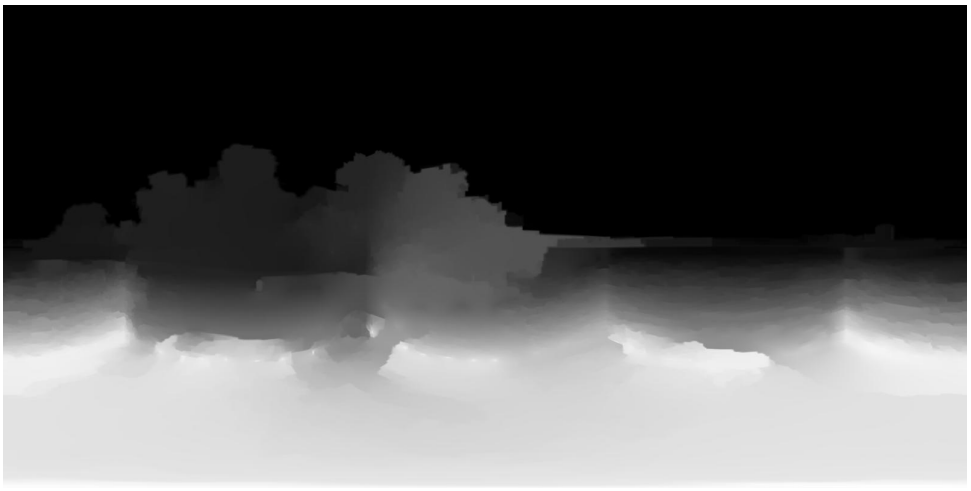


Figura 4.21: Mapa de profundidad del panorama Tantolunden.

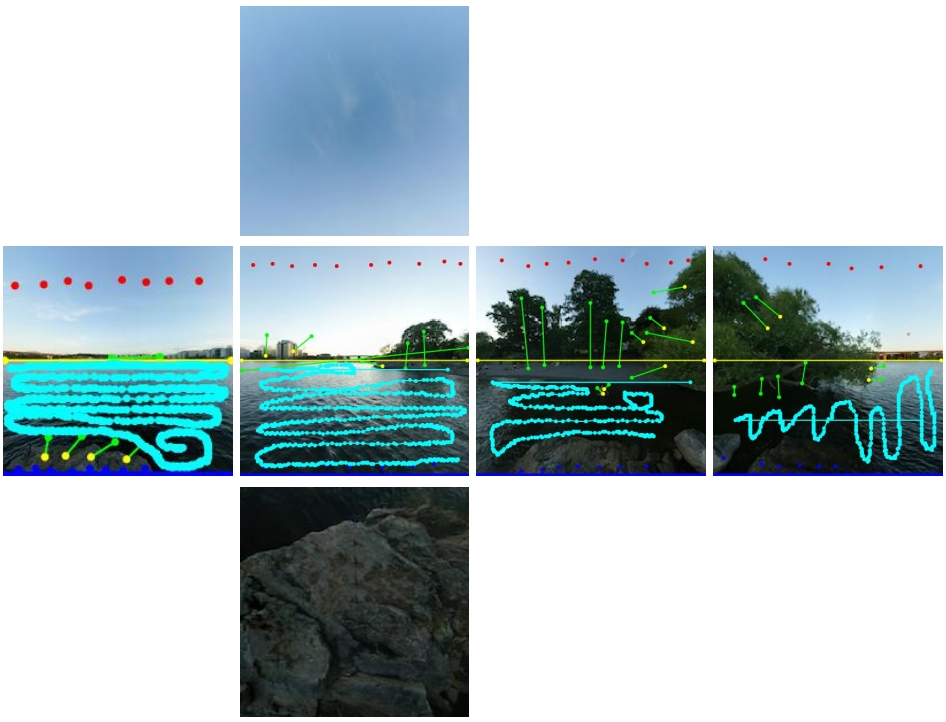


Figura 4.22: Entrada del usuario para el panorama Tantolunden

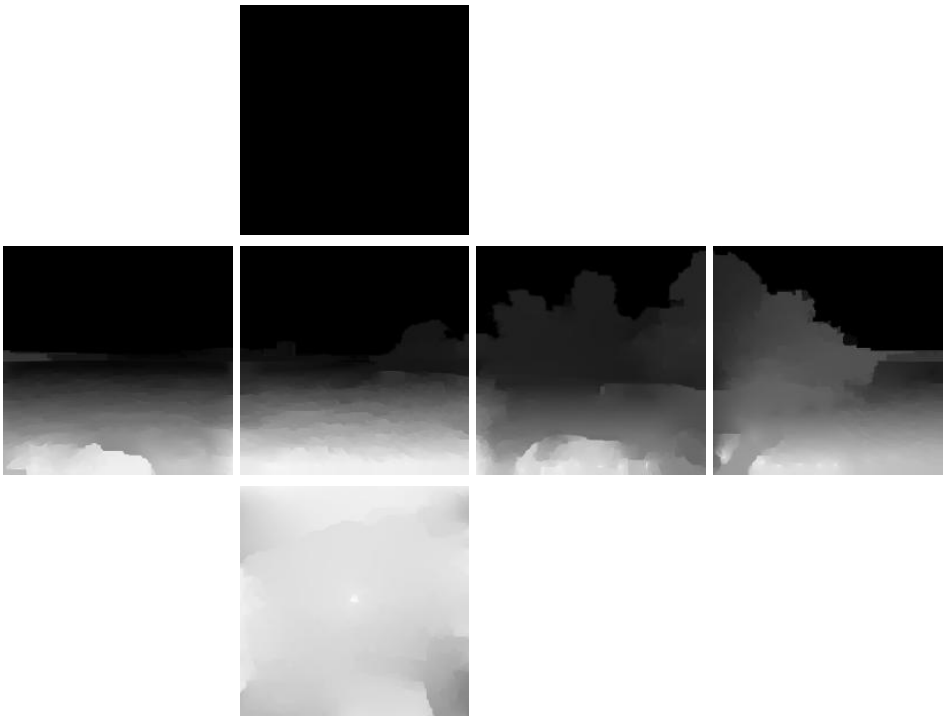


Figura 4.23: Mapa de cubo de profundidad del panorama Tantolunden.

4.3.4. Buddha

Este panorama (ver Figura 4.24) tiene varios detalles por destacar. Por un lado, encontramos la figura de un Buddha en la cara $+Z$, se ha introducido un par de puntos de misma profundidad en la base y en la cabeza para mantenerlo unido, y pares de puntos de distinta profundidad para separarlo del fondo. En las caras $-X$ y $+X$ hay que indicar las escaleras, para esto se utilizan puntos de distinta distancia. Y ha diferencia de los demás ejemplos mostrados, donde la información de profundidad de la cara $-Y$ se completaba con la información de las caras colindantes, es necesario introducir información adicional, ya que en esta cara se encuentran las escaleras.

En la figura 4.25 se muestra el resultado obtenido. La entrada de datos se muestra en la figura 4.26 y el resultado para cada una de las caras en la figura 4.27.

Figura 4.24: Panorama Buddha (Fuente: Emil Persson (Humus) [15]).

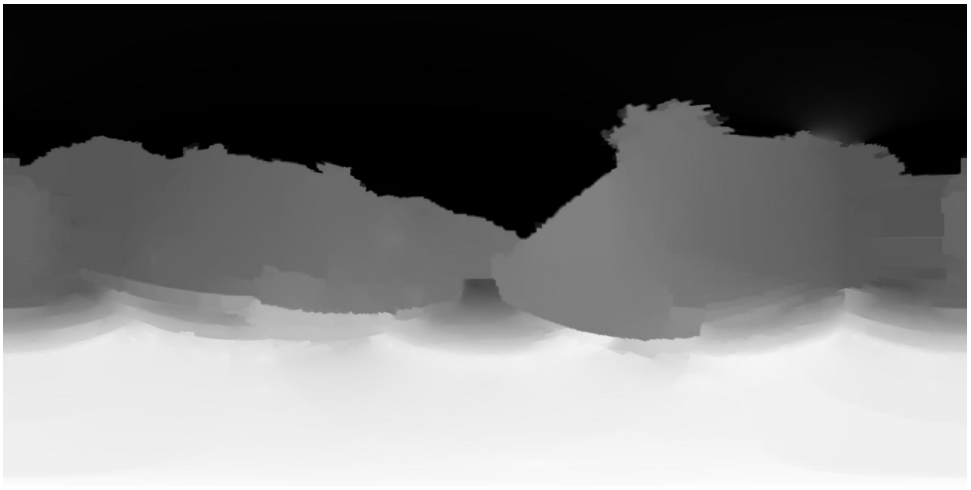


Figura 4.25: Mapa de profundidad del panorama Buddha.

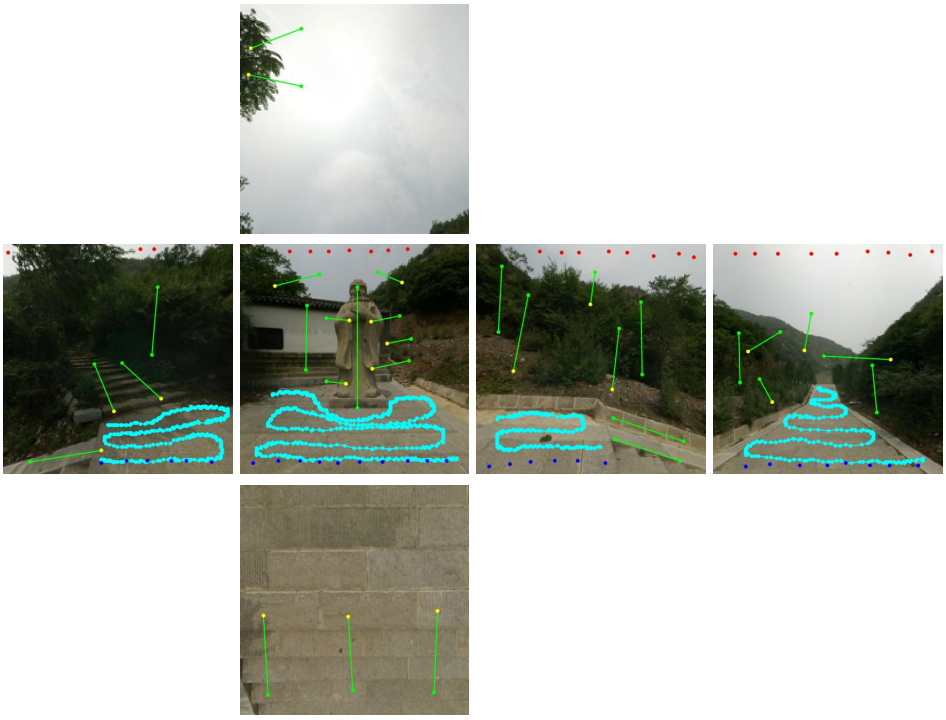


Figura 4.26: Entrada del usuario para el panorama Buddha

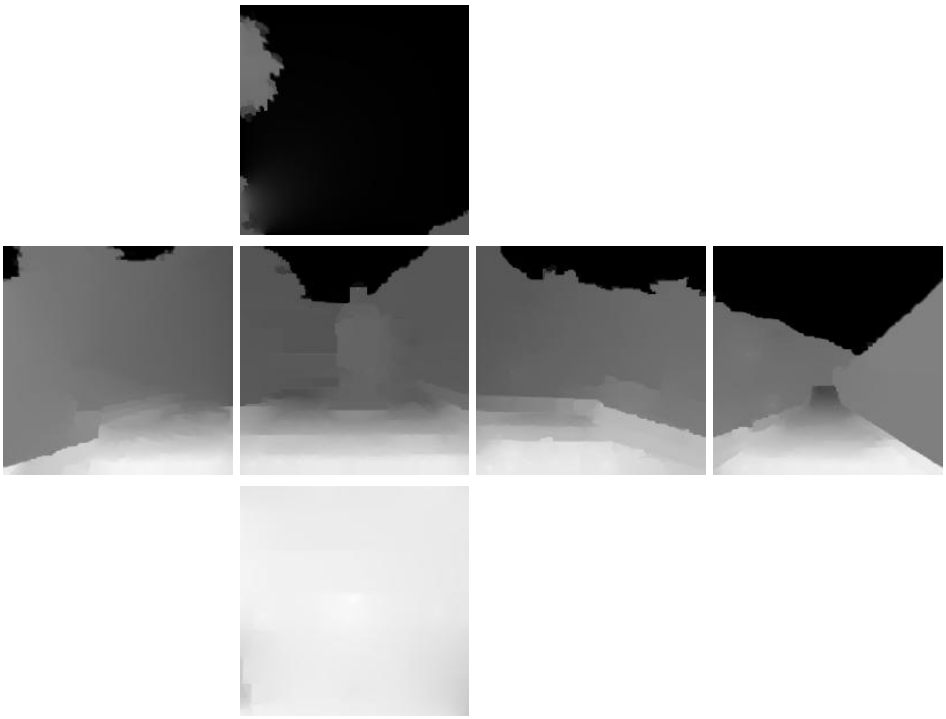


Figura 4.27: Mapa de cubo de profundidad del panorama Buddha.

Capítulo 5

Conclusiones

En este trabajo, se ha explicado qué es un panorama esférico, sus diferentes representaciones y cómo mostrarlas. Se ha desarrollado una aplicación Web, donde el usuario es capaz de añadir información de profundidad del panorama, sobre los tres tipos de representación. Se ha extendido el método de estimación de profundidad para aplicarlo a imágenes de mapa de cubo y obtener la profundidad de un panorama esférico.

Los resultados de los panoramas de profundidad obtenidos son positivos y pueden ser de utilidad en aplicaciones que no requieran precisión en la información de profundidad. Por ejemplo, en aplicaciones que simulen el efecto de profundidad de campo (ejemplo de aplicación implementada para imágenes en perspectiva [17]), o para agregar objetos o niebla. Sin embargo, hay panoramas en los que el usuario debe conocer qué información debe dar y cómo, para poder obtener unos resultados aceptables. Si no se aporta la información adecuada, puede ser necesario repetir el procedimiento.

Como trabajo futuro, queda implementar el método en *Python* y poder procesar la información desde el servidor, sin necesidad de descargar la información introducida desde la aplicación Web, y así conseguir observar el resultado interactivamente, permitiendo que durante el proceso de introducción de los datos pueda hacerse de modo incremental, en función de los resultados que se van obteniendo.

Otra línea de trabajo futuro consistiría en adaptar el método de estimación de profundidad a coordenadas esféricas, de modo que se pueda obtener directamente un mapa de distancias a partir del panorama.

Además, se podría realizar una evaluación exhaustiva de la interacción con los distintos tipos de presentación, incluyendo estudios con usuarios, que permitan averiguar qué elementos o vistas son más adecuados. Esto a su vez podría servir para desarrollar nuevas herramientas para introducir la información de profundidad de una manera más sencilla y que requiera menos conocimientos y habilidades por parte del usuario.

Bibliografía

[1] Scott Highton. *Virtual Reality Photography*. 2003 (last visited June 21, 2018). URL: http://www.vrphotography.com/.

[2] Google. *Google Street View*. 2018 (last visited June 21, 2018). URL: https://www.google.es/intl/es/streetview/.

[3] Angeles Lopez, Elena Garces y Diego Gutierrez. "Depth from a Single Image Through User Interaction". En: *Spanish Computer Graphics Conference (CEIG)*. Ed. por Adolfo Munoz y Pere-Pau Vazquez. The Eurographics Association, 2014. ISBN: 978-3-905674-67-5. DOI: 10.2312/ceig.20141109.

[4] R. I. Hartley y A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, 2004. ISBN: 0521540518.

[5] Gleb Bahmutov y col. "Depth Enhanced Panoramas". En: *Vision, Video, and Graphics (2005)*. Ed. por Mike Chantler. The Eurographics Association, 2005. ISBN: 3-905673-57-6. DOI: 10.2312/vvg.20051017.

[6] Ashutosh Saxena, Min Sun y Andrew Y Ng. "Make3D: learning 3D scene structure from a single still image". En: *PAMI* 31.5 (2009), págs. 824-840.

[7] Felipe Calderero y Vicent Caselles. "Recovering Relative Depth from Low-Level Features Without Explicit T-junction Detection and Interpretation". En: *International Journal of Computer Vision* 104 (1 feb. de 2013), págs. 38-68.

[8] Byong Mok Oh y col. "Image-based modeling and photo editing". En: *SIGGRAPH* (2001), págs. 433-442.

[9] O. Wang y col. "StereoBrush: interactive 2D to 3D conversion using discontinuous warps". En: *Proc. of the 8th Eurographics Symposium on Sketch-Based Interfaces and Modeling (SBIM)*. Vancouver, British Columbia, Canada: ACM, 2011, págs. 47-54.

[10] Kaan Yücer, Alexander Sorkine-Hornung y Olga Sorkine-Hornung. "Transfusive Weights for Content-Aware Image Manipulation". En: *Proc. of the Vision, Modeling and Visualization Workshop (VMV)*. 2013, págs. 57-64.

[11] Eric W. Weisstein. *Spherical Coordinates. From MathWorld–A Wolfram Web Resource*. 1995 (last visited June 21, 2018). URL: http://mathworld.wolfram.com/SphericalCoordinates.html.

[12] D. Zwillinger. *CRC Standard Mathematical Tables and Formulae, 31st Edition*. Advances in Applied Mathematics. Chapman & Hall/CRC, 2003. ISBN: 1-58488-291-3.

[13] Brandon Jones y Colin MacKenzie. *Javascript Matrix and Vector library for High Performance WebGL apps*. 2018 (last visited June 21, 2018). URL: `http://glmatrix.net/`.

[14] Jose Ribelles y Angeles Lopez. *Panorama to Cubemap*. 2017 (last visited June 21, 2018). URL: `http://cphoto.uji.es/grafica/demos/panoramaCubemap/`.

[15] Emil Persson. *Humus*. 2018 (last visited June 21, 2018). URL: `http://www.humus.name/index.php?page=3D`.

[16] Jose Ribelles y Angeles Lopez. *Cubemap to Panorama*. 2018 (last visited June 21, 2018). URL: `http://rubi.dlsi.uji.es/~ribelles/cm2pan/`.

[17] Jose Ribelles y Angeles Lopez. *Depth of Field*. 2018 (last visited June 21, 2018). URL: `http://cphoto.uji.es/demos/depthoffield/`.