Informe proyecto final.

Daniel Méndez *Miembro IEEE,* Camila Sanhueza *Miembro IEEE*

***Resumen*—Basados en el paper de Hake [1], en este informe mostramos los resultados que obtuvimos al intentar hacer la reconstrucción 3D de un objeto, siguiendo lo expuesto por Hake.**

Para comenzar, introducimos explicando las dificultades que se nos presentaron durante el proceso. También definimos las diferencias del resultado final con lo que propusimos al comenzar el desarrollo del proyecto. Hemos de explicar también, el diseño y la implementación del código generado y utilizado.

Se muestran también en detalle los resultados obtenidos y un análisis de éstos.

I. Introducción

La reconstrucción en 3D de un objeto, es un área que tiene muchos, y muy variados, acercamientos. Desde 1980 con la introducción de métodos fotométricos, o a través de la técnica llamada *shape-from-shading*, se ha intentado construir en realidad aumentada los objetos en el computador.

Con la puesta en el mercado de la Kinect, se presentó una nueva forma de lograr el tan anhelado objetivo. Es así como nuevas mentes encontraron formas distintas de hacer una construcción en tres dimensiones.

Para éste proyecto, partimos basándonos en el paper de Hake [1], el cual plantea el uso del receptor infrarrojo para captar las diferentes imágenes del objeto. Además sugiere usar una fuente externa de luz infrarroja, para obtener mejores resultados de resolución a la hora de reconstruir. Los problemas al querer seguir este método, se presentaron a la hora de obtener las normales para hacer el mapa de profundidad, dado que la documentación de Hake es escasa al ser un método propuesto por él.

Para resolver este problema, combinamos el acercamiento de Hake con el de Schindler [2]. Usamos fotometría no calibrada para hacer el calculo de las normales a partir de 4 fotos tomadas con la Kinect y con una fuente externa de luz infrarroja. Luego, para el armar el mapa de profundidad, utilizamos el método de Gauss-Seidel con relajación.

Finalmente, las pruebas son generadas haciendo una modificación del código de Kai Wolf [3], para adaptarlo a nuestras condiciones de trabajo, es decir, utilizar la Kinect como cámara y la fuente externa de lux infrarroja.

II. Diseño e implementación

Para lograr el objetivo de realizar reconstrucciones $3D$, utilizamos el código de Kai Wolf, para las fotometría no calibrada [3], al cual le realizamos una serie de modificaciones y realizamos el código que permitiera utilizar la Kinect como cámara de fotografías. Para poder realizar las reconstrucciones en $3D$ utilizamos una luz infrarroja, como esta no entrego resultados favorables. Posterior a esto comenzamos a utilizar una luz halógena con filtro dicroico, este tipo de luces da paso a rayos infrarrojo visibles.

mail: daniel.mendezf@mail.udp.cl
mail: camila.sanhueza@mail.udp.cl

Código 1. Prototipo función `exportMesh`.

```
1 void exportMesh(cv::Mat Depth, cv::Mat Normals, cv::Mat
      texture);
```

A. Código que permite la utilización de la Kinect

- Diseño: Al momento de analizar como se utilizaría de la Kinect, se nos presentaron diferentes librerías (`OpenNi`, `Freenect`) que nos permitirían esto. Al momento de realizar pruebas con las librerías antes mencionadas ninguna resulto de la forma desea. Lo que nos obligo a utilizar `Openframenworks`.
- Implementación: para poder utilizar la Kinect como sensor de profundidad y de intensidades de la obtener las imágenes. Utilizamos las librerías de `Openframenworks`, mas específicamente `ofxKinect`. Este código nos permite obtener y almacenar las imágenes de intensidad y mapas de profundidad de las pruebas. Esto se realiza al presionar la tecla "S", podemos almacenar las intensidades del frame de ese instante que esta siendo visto por la Kinect. Al presionar "D.ªlmacenamos los mapas de profundidad. El funcionamiento de esta libreria consiste en que es necesario determinar las variables esenciales a utilizar en el archivo `ofApp.h`, el cual crea una clase. En el archivo `ofApp.cpp` se encuentra las siguientes funciones: setup, update y draw, que siempre se ejecutan el mismo orden mencionado, luego realizando ciclos entre update y draw. La función `setup` configura las opciones necesarias para iniciar la Kinect, `update` como su nombre lo indica renueva la información entregada por la Kinect y finalmente `draw` este dibuja la información necesaria en la pantalla. Además incluye la función `pressKey` el cual permite realizar acciones al presionar diferentes teclas.

B. Código de la fotometría no calibrada

1) Función `exportMesh`:

- Diseño: esta funcion nos permite transformar la informacion de las matrices de profunididdad y de normales, en una representacion $3d$. Para poder visualizar las reconstrucciones realizadas utilizamos la herramienta `meshlab`.
- Implementación: es función las variables `ofstream`, en donde esta crea un archivo con extensión `.obj` en donde los datos son ingresados escribiendo en el mismo archivo los valores necesarios de estas matrices.

Código 2. Prototipo función `imageMask`.

```
1 Mat imageMask(vector<Mat> camImages, int numPics, Mat
      ambient);
```

Código 3. Prototipo función `computeNormals`.

```
1 Mat computeNormals(vector<Mat> camImages,Mat Mask =Mat());
```

2) Función `imageMask`:

- Diseño: esta función consiste en la eliminar todo el fondo de la imagen transformándolo a color negro, y donde se encuentra el objeto de prueba esta la silueta del objeto, rellenándolo con color blanco.
- Implementación: en el código 2 se puede observar el prototipo de esta función. El parámetro `camUmagenes`, es un vector que posee las imágenes de las distintas iluminaciones, `numPics` son las cantidad de imágenes que posee este arreglo y finalmente el `ambiente` es una imagen mas oscura para obtener el fondo. Para obtener la mascara se sumaron todos las imágenes y a este se le resto el ambiente. Para no obtener valores fuera del rango de escala de grises se realiza un threshold.

3) Función `computeNormals`:

- Diseño: esta función consiste en calcular la normal para cada pixeles de las imágenes,y asignado a la matriz resultante un valor que es representado con diferentes colores. Como no se podía controlar o calibrar las fuentes de iluminación fue necesario utilizar un método de fotometría estero no calibrada.
- Implementación: como se puede observar en el código 3 que representa al prototipo de esta función. Al igual que en la función anterior se le entrega el arreglo de imágenes, mas la mascara calculada anteriormente. Inicialmente esta función crea una matriz que almacene la información de intensidad de las fotos y en las diferentes columnas de esta representan imágenes con diferentes iluminaciones. Posterior a crear esta matriz se le realiza una descomposición de valores singulares para obtener los valores propios y así poder obtener la información en una matriz de tres canales (RGB).

4) Función `cvtFloatToGrayscale`:

- Diseño: este funcion tal como su nombre lo dice transforma una matriz de valores flotantes a un matriz en escala de grises.
- Implementación: el código 4 es ingresar una matriz para que esta sea transformada a una escala de grises, principalmente a valores que no sobrepasen el valor de 255.

5) Función `localHeightfield`:

Código 4. Prototipo función `cvtFloatToGrayscale`.

```
1 Mat cvtFloatToGrayscale(cv::Mat F, int limit = 255);
```

Código 5. Prototipo función `localHeigthfield`.

```
1 Mat localHeightfield(cv::Mat Normals);
```

- Diseño: esta función nos permite a través del calculo de de normales, realizar una estimación de las profundidades de la imagen. Se utiliza el método de Gauss-Seidel con relajación, esta manera de calcular la profundidad es mediante un método iterativo actualizando el valor de cada pixel en relación a su vecindario. De esta forma se utilizan los valores de las normales y el mapa de profundidad comienza siendo plano a aumentar las diferentes profundidades mediante las iteraciones.
- Implementación: para estimar las profundidades se realizaron iteración con pirámides de la matriz de las normales, en estas iteraciones se calculan la profundidad de un píxel mediante el análisis del vecindario de este. Después de la estimación de esta matriz de profundidad buscamos los valores máximos y mínimos para que nuestra profundidad no supere estos valores.

6) Función `E_d`:

- Diseño: como menciona Haque [1] en su paper es necesario realizar diferentes funciones de costos, una de estas es calcular el costo para las normales. Este se rige por

$$E_d(\hat{Z}) = \sum_p w_p ||u_p||^2 (Z_p - \hat{Z}_p)^2 \qquad (1)$$

en donde $u_p = [\frac{-x}{f} \frac{-y}{f} 1]^T$, w_p representa los valores propios de $\sum_q N_q N_q^T$ y donde Z representa las profundidades para los distintos valores de los pixeles.

- Implementación: para este se implemento la ecuación 1, para cada valor de las distintas profundidades, ya que se ingresan las profundidades estimadas y las profundidades calculadas con la Kinect.

7) Función `E_n`:

- Diseño: para el costo de las normales, se utilizo la siguiente formula

$$E_n(\hat{Z}) = \sum_p (N_p T_x)^2 + (N_p T_y)^2, \qquad (2)$$

donde T_x y T_y, representan las derivadas parciales a sus respectivos ejes en Z y donde N_p es el valor de la normal para el píxel p.

- Implementación: para implementar esta función se utilizo la ecuación 2, realizando la sumatoria de todos los pixeles con los valores antes mencionados.

8) Función `E_s`:

- Diseño: del mismo modo que las funciones anteriores este se rige por $\bigtriangledown^2(\hat{Z})$, en cual es el operador Laplaciano para coordenadas cartesianas.
- Implementación:para realizar esto es necesario realizar las deribadas de Z con respecto a los ejes x e y. Para esto se utilizo la función de `sobel` implementada por `openCV`, se realizo dos veces cada derivada a los diferentes ejes y luego estas se sumaron.

9) Función `main:` en esta función se utilizan todas las funciones mencionadas anteriormente, el orden de esto generar las mascaras de las imágenes, se elimina el fondo de las imágenes con la mascara ,se calculan las normales, se calculan las profundidades y finalmente se realiza el mesh.

III. Experimentos

IV. Conclusión

Referencias

[1] S. M. Haque, A. Chatterjee, and V. M. Govindu, "High quality photometric reconstruction from a depth camera," 2014. [Online]. Available: http://www.ee.iisc.ernet.in/labs/cvl/papers/photometric_CVPR2014_paper.pdf

[2] G. Schindler, "Photometric stereo via computer screen lighting for real-time surface reconstruction."

[3] K. Wolf, "Uncalibrated-photometric-stereo." [Online]. Available: https://github.com/NewProggie/Uncalibrated-Photometric-Stereo