Universidad Nacional de San Agustín
Escuela de Ciencia de la Computación

Computación Gráfica

Escáner 3D para modelado de objetos

Presentado por:
Melvin Salcedo Almiron
Enrique Gutierrez Salazar

Índice

1. Introducción

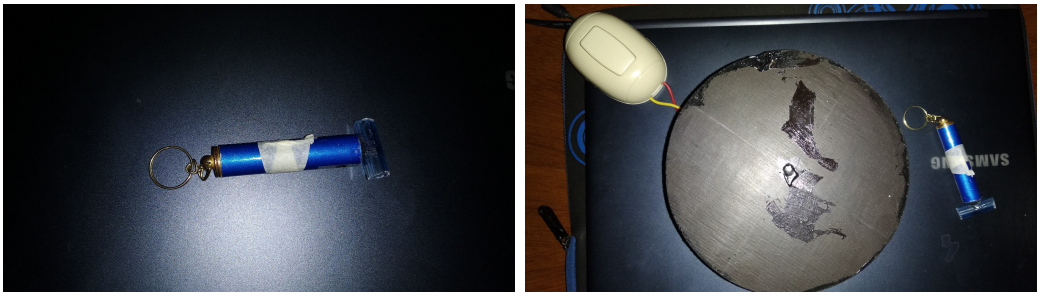
El escáner 3D es un dispositivo que analiza objetos o entornos para poder obtener los datos de sus formas y su apariencia. Los datos obtenidos son utilizados para la reconstrucción digital del objeto, crea un modelo tridimensional que puede ser utilizado en diversas aplicaciones: industriales, entretenimiento, videojuegos, ingeniería inversa, control de calidad, arqueología, entre otros. El objetivo del escáner 3D es crear una nube de puntos de muestras geométricas sobre la superficie del objeto. Mediante estos puntos puede extrapolarse la forma del objeto mediante un proceso de reconstrucción digital. [2]

2. Construcción básica de un escáner 3D

Para la construcción básica de un escáner 3D está conformada principalmente por un sensor, un emisor de señales y un sistema de procesamiento de datos. Los dispositivos láser son los más utilizados para emitir la señal que se va a capturar y procesar, siendo los láseres de línea los más populares. Se utilizan cámaras como sensores gracias a que son muy eficientes para la captura en datos. Por último, el sistema de procesamiento de datos es un software diseñado para transformar y procesar las señales provenientes de la cámara, las cuales vienen en forma de punto de nubes de puntos.

3. Planteamiento

Requisitos necesarios para armar el escenario.



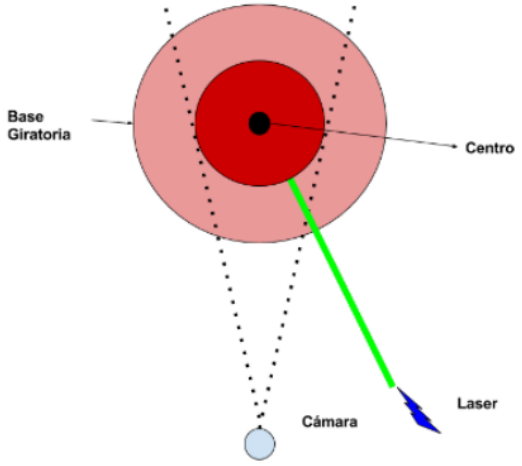
(a) Laser usado. (b) Base de rotacion

- Cámara de un celular
- Un puntero láser de color rojo
- Plataforma de rotación
- un lapicero: nos ayudara a generar una linea para la segmentación.

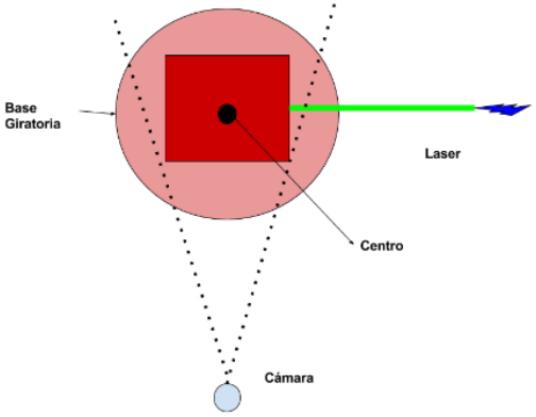
La cámara de celular lo utilizaremos como una cámara web cam gracias a una aplicación llamado IP Webcam, esta aplicación lo podemos encontrar en PlayStore, la finalidad de este programa es poder acceder a la cámara bajo una dirección IP generada por la aplicación en nuestro caso nos generó http://192.168.42.129:8080

4. Fijando variables

Para comenzar, requerimos frames suficientes, que nos den la información de 360 grados de vistas alrededor del objeto. Calibramos la velocidad de rotación de la base y alineamos la cámara mirando de frente al centro del objeto a escanear.



(a) Angulo Theta demasiado pequeño



(b) Angulo Theta demasiado grande

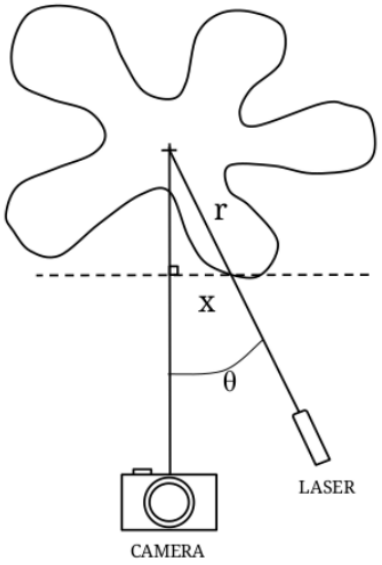
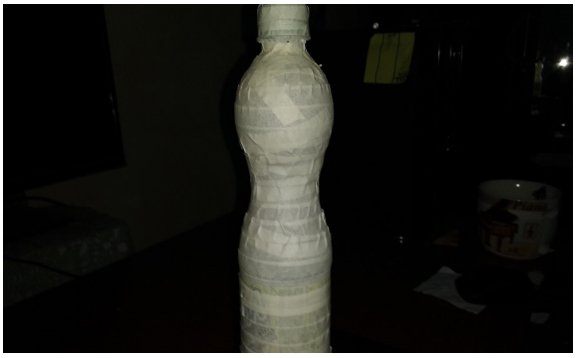
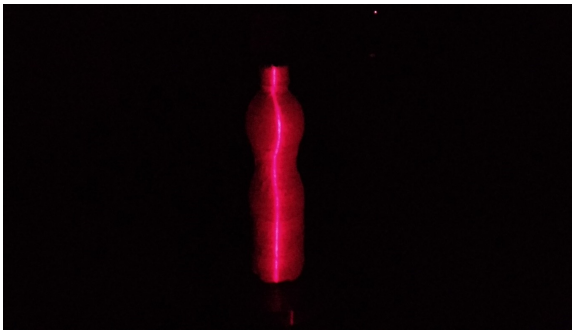


Figura 3: Esquema del espacio y variables a utilizar



(a) Objeto a escanear



(b) Proyección del laser en el objeto

5. Segmentación

Para extraer la distancia x del frame a tratar. Primero es necesario hacer una segmentación para ignorar todos los elementos que no sean ´útiles. En este caso solo requerimos la línea proyectada por el láser, ya que partimos asumiendo que el centro del frame coincide con el centro del objeto (eje y). Utilizaremos la librería OpenCV para capturar los frames. Esta librería nos provee varias funciones útiles para ejecutar la segmentación. [1]

- TheVideoCapturer: Detecta por defecto la cámara que tenemos en nuestra computadora para una mejor comodidad de la cámara utilizaremos la cámara de un celular es por ello que le indicamos la dirección URL que nos generó el IP Webcam.
- TheVideoCapturer.read(imgOriginal) Extrae cada frame y lo almacena en imgOriginal que es un tipo de Mat, la estructura que OpenCV utiliza para almacenar una matriz.
- cvtColor: Convierte una imagen de un espacio de color a otro debido a que la imagen generada es RGB y nosotros necesitaremos una imagen de escala de gris, se creó una variable Mat imgHSV que será la que contenga nuestra imagen de escala de gris

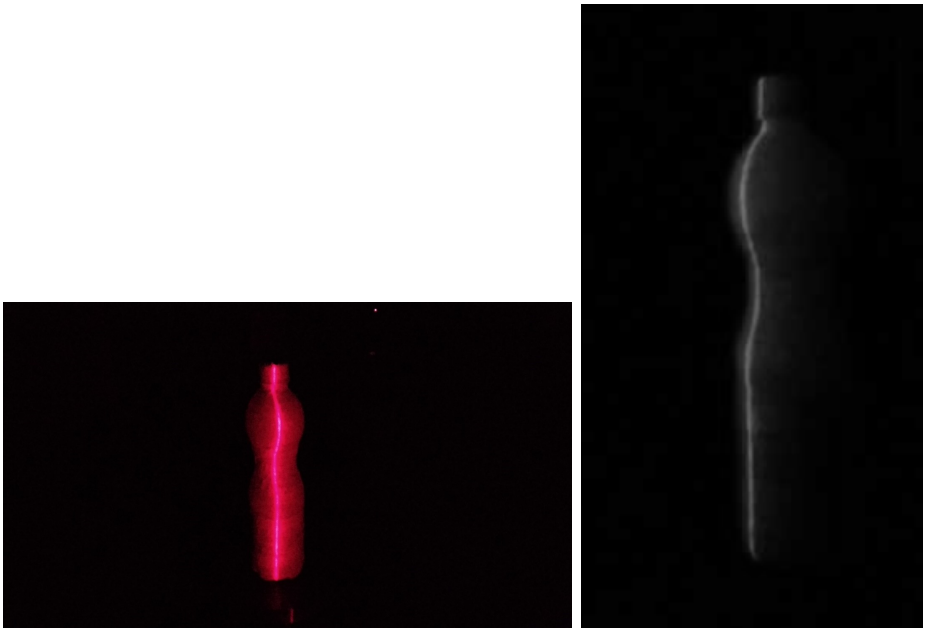


Figura 5: Segmentación

6. Calculando distancias

En la figura 3 se muestra la estructura de la que disponemos para el cálculo. Teniendo como base el frame segmentado. Tomaremos cada punto en Y y aproximaremos la coordenada en X de acuerdo a la distancia x mostrada en el esquema. Esta distancia será obtenida en pixeles. Se pretende escalar esta distancia con la equivalencia:

$$100px = 1,0$$

en caso que la cámara sea de 480px de ancho. Este depende y se ajusta de acuerdo a la resolución de la cámara. La coordenada Y toma la misma escala. Para conseguir la coordenada Z, que representa la profundidad, de acuerdo al esquema es necesario hallar la distancia r, desde el punto actual al eje y que coincide con el centro del cuerpo, y teniendo en cuenta que conocemos x, luego de la aproximación visual. Aplicamos la siguiente formula:

$$R = (C1 - width/2)$$

$$C1 = constante$$

Una vez obtenida el radio, solo queda calcular la coordenada Z. tenemos:

$$Z = width(imagen)/C2$$
$$C2 = constante$$

'C1 y C2' están constantemente cambiando, esto varia de acuerdo a ala posición de los contornos capturados en la imagen. por ejemplo 'C1': si el contorno del objeto esta mas allá del centro la 'C1' es mas próximo mayor a 0.5 caso contrario es menor que 0.5. y 'C2' cambia entre 0 y 1 mientras mas cercano a 0 la altura del objeto escaneado es mas ancha y cuanto mas próximo a 1 la altura del objeto es mas corta.

Ya tenemos el punto Z para un X y Y actuales. Por cada Ángulo recorreremos toda la línea del láser tanto en X y Y. Para completar todos los puntos alrededor de los 360 grados, será necesario aplicar rotación de puntos en 2D. Ya que la coordenada Y no se verá modificada. X y Z serán modificadas de acuerdo al ángulo α que ira de 1 a 360 grados. Luego:

$$X' = R.cos(rotacion)$$
$$Y' = R.sin(rotacion)$$

7. Explicación del código

7.1. Obtención de las imágenes

Hacemos una conexión con la aplicación ipcam instalada en el celular y una vez establecida la conexión capturamos las fotos necesarias para el escaneo 3d.

```
int main(int argc, char *argv[]) {
        char key = 0;


    TheVideoCapturer.open("http://192.168.1.46:8080/video?x.mjpeg")
        ;
    //TheVideoCapturer.open("http://10.42.0.199:8080/video?x.mjpeg
        ");

    if (!TheVideoCapturer.isOpened()) {
        std::cerr<<"\n\nConexion no encontrada con el dispositivo\n
            \n"<<std::endl;
        return -1;
    }

        std::thread fotografo;
        fotografo=std::thread(tomar_foto);

        while(TheVideoCapturer.grab() && cont<36){
        TheVideoCapturer.retrieve(bgrMap);
        cv::waitKey(20);
        }

        fotografo.join();

}
```

7.2. Obtención de los contornos

Recorremos por cada pixel de la imagen y cuando encuentre el promedio de un pixel de los valores RGB con un promedio no mayor de 60 guardamos esa posición en una

nueva imagen con los datos cambiados a 255 y 0 caso contrario.

```
void Obtener_contornos::transformar_imagen(){
  for(int i=0;i<tam_imagen.height;i++){
    std::vector<int> Temporal;
    for(int j=0;j<tam_imagen.width;j++){

      intensity = img_destino.at<cv::Vec3b>(i, j);
      r=intensity.val[0];
      g=intensity.val[1];
      b=intensity.val[2];
      prom=(r+g+b)/3;

      if(prom<60){
        intensity.val[0]=0;
        intensity.val[1]=0;
        intensity.val[2]=0;
        img_destino.at<cv::Vec3b>(i, j) = intensity;
      }
      else{
        intensity.val[0]=255;
        intensity.val[1]=255;
        intensity.val[2]=255;
        img_destino.at<cv::Vec3b>(i, j) = intensity;

        for(int jj=j+1;jj<tam_imagen.width;jj++){
          intensity = img_destino.at<cv::Vec3b>(i, jj);
          intensity.val[0]=0;
          intensity.val[1]=0;
          intensity.val[2]=0;
          img_destino.at<cv::Vec3b>(i, jj) = intensity;
        }
        break;
      }
    }
    //cout<<"\n"<<endl;
  }
  imwrite("contornos/"+nombre_imagen+".png",img_destino);
}
```

7.3. Obtención de los puntos

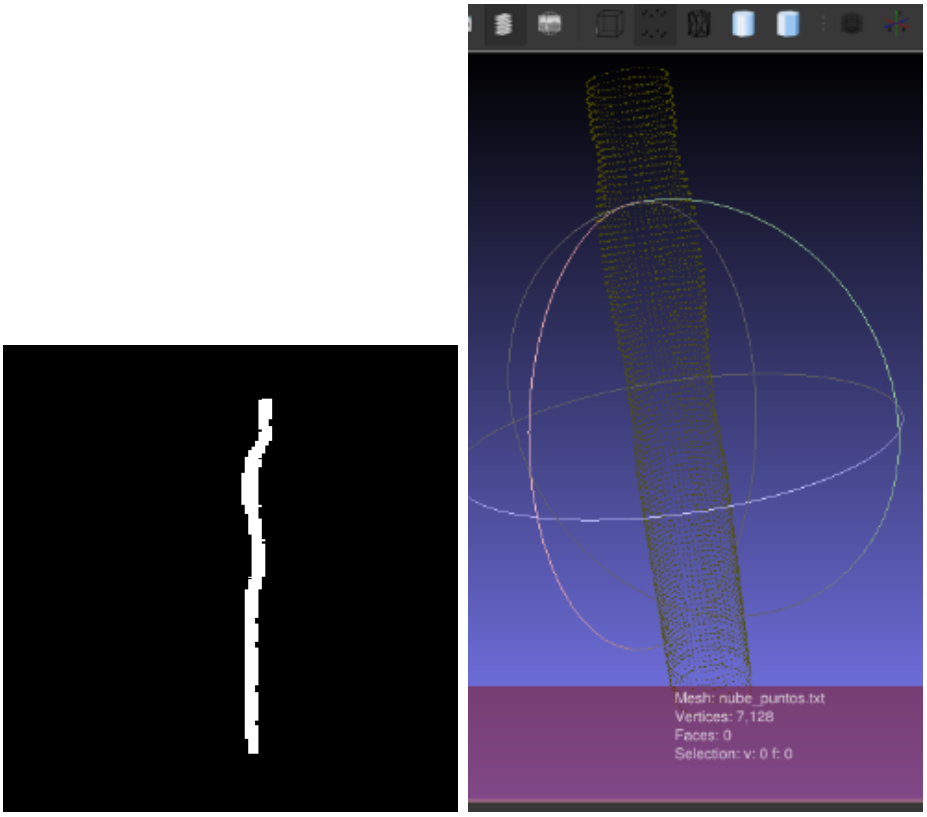
Al tener los punto x,y de la imagen calculamos z con el algoritmo mostrado anteriormente.

```
void obtener_puntos::obtener_xyz(int i, int j,int e){

    e*=0.4;
    float x = i;
    float y = j;
    float z = x;

    float d = (e-j);

    x = d * cos (angle);
    y = d * sin (angle);
    cont++;
    std::ofstream archivo ("nube_puntos.txt" , std::fstream::app);
    //std::ofstream archivo_opengl ("nube_puntos.txt" , std::
        fstream::app);
    int contador_lineas=0;
    if (archivo.is_open()){
      //cout<<x<< " " <<y<< " " <<z<<endl;
      archivo<<x<<";"<<y<<";"<<z<<endl;
      archivo.close();
    }
}
```

8. Ejecución del código

(a) Objeto a escanear (b) Objeto escaneado con laser

Figura 6: Ejecución del código.



(a) Objeto escaneado (b) Resultado

Figura 7: Ejecución del código continuación

9. Conclusiones

En un caso ideal, cada frame corresponde a un ángulo. Pero no será así necesariamente. Es importante calibrar la velocidad de la base giratoria antes de empezar a calcular los puntos. De acuerdo al ángulo θ elegido, se perderán datos para zonas cóncavas. Esto podría solucionarse usando dos láseres. El ángulo entre ambos láseres seria 90 grados para tratar de salvar estas zonas de perdida. En este caso solo presentamos la nube de puntos.

Referencias

[1] Miscellaneous image transformations. `https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html`. Accessed: 2018-12-8.

[2] Ernesto Javier Claudia Marcela. *Diseño e Implementacion de un escáner 3D para prototipado y modelado geométrico de objeto*. 2014.