Universidad Autónoma de Yucatán

Maestría en ciencias de la computación

Titulo:

Algoritmo de reconstruccion tridimensional

I.S.C. Javier Armando Jiménez Villafaña

Maestro:
Dr. Arturo Espinosa Romero

Índice

1. Resumen

La forma es una de las características mas importantes en la recuperación de información basada en el contenido de la imagen. Muchas representaciones y métodos de recuperación de información existen actualmente, y la mayoría de estos métodos no representan la forma de manera correcta o son difíciles de normalizar (haciendo la tarea del emparejamiento y comparación muy difícil).Recientemente han aparecido dispositivos que permiten obtener información tridimensional de una escena, y uno de los más utilizados es el constituido por un par estereoscópico de cámaras. En este trabajo los datos 3D son representados por nubes de puntos en un entorno gráfico virtual, ploteado por el software Matlab.

2. Introducción

La obtención de una reconstrucción tridimensional a partir de imágenes bidimensionales es un problema importante en diferentes campos, tales como: biología, medicina, microscopía electrónica, topografía, diseño asistido por computadora, simulación por computadora, visualización científica, etc.
Las técnicas para obtener una reconstrucción 3D pueden dividirse en:

1. Reconstrucción del volumen: tomografía, microscopía electrónica.
2. Reconstrucción de la superficie: visualización de terrenos, robótica, arqueología.
3. Realidad aumentada y la realidad virtual.

3. Planteamiento del problema

El problema que existe en las tareas de reconstrucción tridimensional es precisamente la cantidad puntos que intentamos buscar en otra imagen, que son las correspondencias, es decir un punto $p(x, y)$ en la imagen 1, que exista en la imagen 2, y este punto será el punto $p'(x', y')$. Hacer una comparación de manera manual por un método de fuerza bruta es una opción, pero consumiría un tiempo tremendo en calculo computacional. En cambio si buscamos en regiones, zonas o puntos de interés relevantes en la imagen, disminuimos enormemente la cantidad de puntos que debemos probar para verificar si precisamente existe una correspondencia en una segunda imagen.

4. El modelo de cámara pin hole.

El modelo de cámara pin hole es uno de los mas sencillos que podemos plantearnos, pero que es de mucho interés, ya que modela razonablemente bien una cámara común con una formulación matemática muy simple.

El modelo de cámara pin hole supone que para todo punto que impacta en la película, el rayo de luz que sale rebotado de un cuerpo y llega a la cámara, atraviesa un único punto, independientemente del punto de origen y del punto de impacto en la película. Esto hace el modelo matemático de la cámara muy simple, ya que de una aplicación directa del teorema de Thales de Mileto, podemos sacar las ecuaciones:

$$\begin{cases} Zx &= -fX, \\ Zy &= -fY \end{cases}$$

Suponiendo que:

1. Las coordenadas del punto en el espacio que se proyecta serán $(X, Y, Z)^T$ donde el eje de la coordenada Z es la linea que pasa por el centro óptico y es perpendicular al plano de proyección, y las coordenadas $(0, 0, 0)$ están situadas en el centro óptico.
2. Las coordenadas del punto proyectado serán $(x, y)^T$
3. f es la *distancia focal* es decir la distancia que separa el centro óptico del plano de proyección.

Si queremos operar con álgebra matricial, lo que es mas cómodo para trabajar computacional mente hablando, las ecuaciones pueden reformularse en coordenadas homogéneas como:

$$Z \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -fX \\ -fX \\ Z \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Sabiendo que hemos deifinido el punto proyectado m como:

$$\begin{bmatrix} -fX \\ -fY \\ Z \end{bmatrix}$$

El punto M se proyectara como:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Y la matriz de proyección P como:

$$\begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A lo que la ecuacion de coordenadas homogeneas puede quedar reformulada como:

$$\lambda m \ = \ PM$$

Donde λ es un factor de escalamiento y coincide con la coordenada Z del punto M. A esto le hemos ganado bastante, ya que esta reformulación matemática a pesar de seguir siendo matricial, el hecho de operar con coordenadas homogeneas, supone que siempre tratamos con un sistema lineal de ecuaciones, por lo que podemos usar numerosas técnicas numéricas sencillas, que pueden ser resueltas computacionalmente hablando.

La formula matemática de la matriz de proyección P puede ser más complicada en caso de que no empleemos el modelo pin hole, al modelar la imagen. También puede involucrar más parámetros que apenas la distancia focal f. Al proceso de cálculo de estos parámetros, que nos permiten deducir la matriz de proyección, lo denominamos calibración de la cámara.

5. Visión estéreo y geometria epipolar

Definimos visión estéreo como aquella en la que empleamos mas de una imagen para obtener una idea de tridimensionalidad. Según el numero de imágenes que empleemos, hablaremos de visión $bifocal$, dos imágenes, $trifocal$, tres imágenes, $cuadrifocal$, cuatro imágenes, o $N\ focal$, N imágenes.

Para nuestro caso, estudiaremos los conceptos de un sistema bifocal; aunque lo estudiado es extrapolable a cualquier sistema N focal.

Dado un sistema bifocal, definimos como epipolo e_i de una de las dos imágenes del sistema a la proyección del centro óptico de la otra imagen. Un sistema bifocal tiene exactamente dos epipolos, uno por cada imagen. Los dos epipolos, pueden tener las mismas coordenadas, o coordenadas distintas.

Definimos como linea epipolar l_i para uno de los dos planos de proyección a la linea que pasa por el epipolo e_i contenido en el plano de proyección y la proyección m_i. Observamos que la linea epipolar $e_i m_i$, es también la proyección de la linea que pasa por el centro optico C_i y la proyección m_i sobre el plano de proyección. Tenemos exactamente dos lineas epipolares, una por cada imagen. Las dos lineas epipolares pueden tener las mismas coordenadas o coordenadas distintas. Un ejemplo gráfico lo podemos ver en la figura 1.

Haz de planos.

La línea de base corta a cada (plano de la) imagen en los epipolos, **e** y **e'**.

Cualquier plano que contiene a la línea de base es un plano epipolar y corta a las imágenes en sendas rectas epipolares **l** y **l'**.

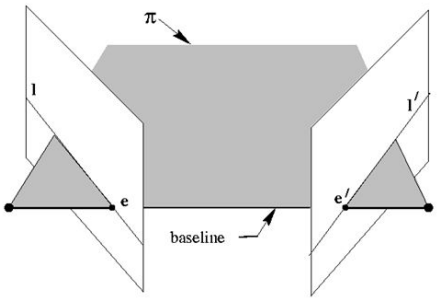

Figura 1: Haz de planos en un sistema bifocal.

Definimos como plano epipolar al plano definido por los centros ópticos de las dos imágenes, y el punto M del espacio tridimensional. Tanto las dos proyecciones m_1, y m_2 del punto M como sus epipolos e_1 y e_2 están contenidos en el plano epipolar. Si M está fuera de los dos planos de proyección y los planos de proyección no son paralelos, existe un único plano epipolar como se muestra en la figura 2.

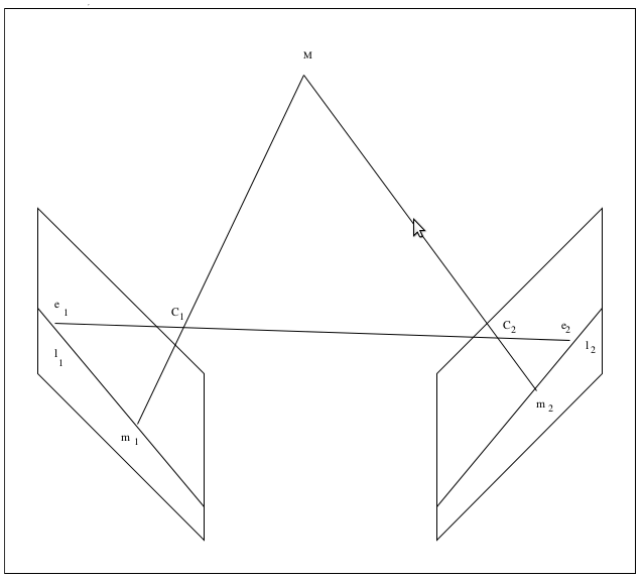

Figura 2: Plano epipolar.

6. Detector de esquinas de Harris

La detección de esquinas es un acercamiento usado en los sistemas de visión por computadora para extraer ciertos tipos de rasgos e inferir el contenido de una imagen. La detección de esquinas frecuentemente se usa en la detección de movimiento, análisis de imagen, rastreo en video, modelado 3D y reconocimiento de objetos entre otros. La detección de esquinas se solapa con un tema más abarcador: la detección de puntos de interés.

Definamos esta imagen por I. Tomemos un parche de la imagen encima del área (u, v) y cambiándolo por (x, y). La suma ponderada de las diferencias cuadradas (SDC) entre estos dos parches, denotada por S, viene dada por:

$$S(x,y) = \sum_u \sum_v w(u,v)\ \left(I(u+x, v+y) - I(u,v)\right)^2$$

$I(u+x, v+y)$ puede aproximarse por una serie de Taylor. Definamos a I_x y I_y como las derivadas parciales de I, tal que:

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

Esto produce la aproximación:

$$S(x,y) \approx \sum_u \sum_v w(u,v)\ \left(I_x(u,v)x + I_y(u,v)y\right)^2,$$

que puede escribirse en la forma de la matriz:

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix},$$

donde A es el structure tensor,

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

Esta matriz es la de Harris, y pudiéndose ver los promedios (es decir la suma sobre de (u, v)). Si una ventana redonda (o circularmente pesó la ventana, es usada como una función gaussiana, entonces la respuesta será isotrópica.

7. Selección de areas y reducción de puntos de interés.

Ahora bien el algoritmo de Harris, dependiendo del humbral hessiano que se le ponga como limite para detectar lo que es una esquina de lo que no lo es, puede arrojarnos un numero indefinido de puntos de interés, lo cual no es malo, el problema viene a darse cuando sobre un área de muy poca longitud existen múltiples puntos de interés a muy escasa distancia, lo cual dificulta mucho el proceso de encontrar correspondencias entre un par de imágenes, como se puede observar en la figura 3.

Figura 3: Esquinas aconglomeradas en muy poco espacio.

Por esta razón, debemos realizar un paso adicional después de calcular los puntos de interés, y este es la reducción del numero de esquinas que hay por cada zona en la cual dividimos nuestra primera imagen, como se indica en el algoritmo siguiente:

Algoritmo para la reducción del número de esquinas:

1. *Escogemos un tamaño de mosaico de $m * n$, y dividimos la imagen en mosaicos.*
2. *Entre esquina y esquina dentro del mosaico hacemos una separación al menos de 10 pixeles.*
3. *El numero máximo de esquinas dentro de cada mosaico sera 5.*
4. *Por cada mosaico, Seleccionamos la esquina con mayor porcentaje de esquinidad y almacenamos las coordenadas de su ubicación (x,y).*
5. *Ordenamos la lista de esquinas en forma decreciente de mayor a menor, por cada mosaico tendremos una lista de esquinas.*
6. *Por cada esquina, si la esquina a seleccionar tiene al menos el mínimo de separación necesario de la esquina principal con mayor índice de esquinidad, si existe otra esquina que cumpla con el criterio de distancia y realizamos la misma comprobación, entonces guardamos esa esquina como punto de interés para el mosaico actual, para las iteraciones futuras, realizaremos esta misma comprobación.*
7. *El criterio de paro se cumple cuando el numero de esquinas seleccionadas es igual ó mayor al máximo permitido por cada mosaico ó cuando todas las demás esquinas se han descartado.*

8. *Repetir para cada mosaico, hasta completar el numero total de mosaicos*

Y obtenemos una imagen, con un numero eficiente de esquinas, las cuales son las que tienen mayor indice de esquinidad, y no hay mas de cierta cantidad por cada mosaico, como se puede observar en la figura 4.

Figura 4: Imagen con esquinas reducidas.

8. Método de búsqueda de correspondencias.

Una vez encontradas, las esquinas en la imagen uno, el siguiente paso es encontrar sus debidas correspondencias en la imagen dos, para este proceso debemos tener entonces una lista de puntos de interés que no excedan el numero máximo permitido por cada mosaico de todos los mosaicos en la imagen, y procederemos entonces a encontrar su debida correspondencia en la imagen dos como se indica a continuación:

Algoritmo para la búsqueda de correspondencias:

1. *Para cada punto (x,y) en la imagen uno, crearemos una ventana de busqueda de* $m * m$*, donde* m *es un numero impar.*
2. *Recorreremos esa ventana a n posiciones a la redonda, de donde se ubica nuestra correspondencia en la imagen uno, pero en la imagen dos.*
3. *Para cada posición desplazada en la imagen dos calcularemos el operador SSD "Sum of square diferences", o suma de diferencias cuadráticas.* $\text{SSD} = \sum_{i,j,\epsilon W}(I_1(i,j)–I_2(i,j))^2$.

4. *Seleccionaremos para la coordenada (x,y), la coordenada (x',y') de la imagen dos, donde la sumatoria de la diferencia de cuadrados sea mínima.*

Y como se muestra en la figura 5, al punto esquina con su correspondencia en la imagen numero dos.

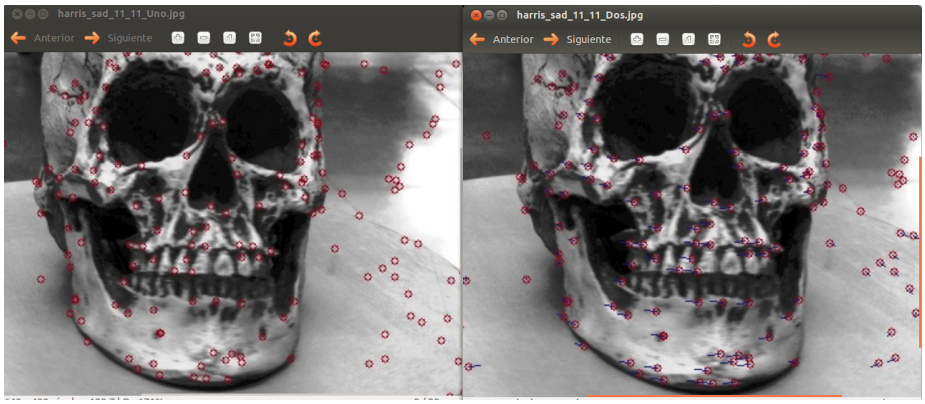

Figura 5: Correspondencias entre un par de imagenes.

9. Normalización de puntos.

Ya teniendo un conjunto de puntos de interés en nuestra imagen uno, y ahora mapeados en nuestra imagen dos, a continuación la idea principal de una reconstrucción tridimensional es encontrar una matriz fundamental F, que mapee los puntos P(x,y) de la imagen uno, a puntos P'(x',y') en la imagen dos. Para ello utilizaremos dos algoritmos dependientes uno de otro, primeramente el algoritmo de 8 puntos, y el algoritmo de Consenso de Muestras Aleatorias de Estimación Robusta o RANSAC. Pero para alimentar a ambos algoritmos, debemos normalizar previamente nuestros conjuntos de puntos para poder centrarlos en el origen del plano. Para ello utilizaremos el siguiente algoritmo:

Algoritmo para la normalización de puntos:

1. *Para cada vector de coordenadas (X, Y) calcularemos un centroide en $\bar{X}$, y un centroide en $\bar{Y}$, los cuales serán el promedio de la sumatoria de todas las coordenadas X e Y respectivamente.*

2. *A cada par de coordenadas (x, y) restaremos su centroide X e Y, para centrarlos en el origen.*

3. *Calcularemos la desviación estándar σ para los puntos normalizados en X, y en Y.*

4. *Finalmente generaremos una matriz de escalamiento T, una para cada vector normalizado.*

$$T = \begin{bmatrix} \frac{1}{\sigma X} & 0 & \frac{-\bar{X}}{\sigma X} \\ 0 & \frac{1}{\sigma Y} & \frac{-\bar{Y}}{\sigma Y} \\ 0 & 0 & 1 \end{bmatrix}$$

10. Algoritmo de 8 puntos

Podemos calcular la matriz fundamental con 8 pares de puntos. Para ello desarrollaremos una matriz de ecuaciones, por asi decirlo, que contenga con ocho ecuaciones y y nueve incógnitas, en la que la novena ecuación la obtenemos sabiendo que el rango de la matriz fundamental es de dos. Por ello una de las filas debe ser linealmente dependiente de otra. Escogemos una fila al azar y decimos que es igual a otra fila al azar, multiplicada por la constante que queremos, exceptuando el cero. Esto nos dará una ecuación mas. Esto significa que con 8 puntos tendremos exactamente nueve incógnitas.

Algoritmo de 8 puntos normalizado:

1. *Normalización: Transformar las coordenadas de la imagen de la siguiente manera:* $\hat{u}1_i = T_1 u_{1i}$ *y de la misma manera para* v *con* T_2*; donde* T_1 *y* T_2 *son transformaciones de normalización que consiste en una translación y escalamiento. A partir de ahora* u_i *y* v_i *son las coordenadas de los puntos normalizados.*

2. *Encontrar la matriz Fundamental F que corresponde a los puntos correspondientes* $p_1 \leftrightarrow p_2$ *de la siguiente manera: a. Solución lineal: Determinar F del vector singular correspondiente al valor singular más pequeño de la matriz A. b. Cumplimiento de la restricción forzoso: Remplazar F por la matriz fundamental cuyo determinante sea cero; usando SVD.*

3. *Regresar las coordenadas de los puntos a su posición original.*

4. *Finalmente generaremos una matriz de escalamiento T, una para cada vector normalizado.*

La matriz de ecuaciones A se formara de la siguiente manera:

$$A = [x'.*x\ x'.*y\ x'\ y'.*x\ y'.*y\ y'\ x\ y\ 1];$$

Entonces realizaremos una descomposicion de valores singulares a la matriz A

La matriz de ecuaciones A se formara de la siguiente manera:

$$U, S, V^T = SVD(A)$$

Y ahora obtendremos de nuestro vector V transpuesto, la columna derecha, la cual es el espacio nulo derecho del vector V (endV) para formar nuestra matriz fundamental F para ese conjunto de 8 puntos.

$$endV = V[:, 8]$$
$$F = \begin{bmatrix} endV[0,0] & endV[0,1] & endV[0,2] \\ endV[0,3] & endV[0,4] & endV[0,5] \\ endV[0,6] & endV[0,7] & endV[0,8] \end{bmatrix}$$

Y realizamos una descomposicion en valores singulares de esta matriz F

$$U, S, V^T = SVD(F)$$

Y tomamos el vector de eigen valores S, y forzamos la norma de frobenius, limitando el rango de la matriz fundamental F a 2, convirtiendo el elemento inferior derecho de la matriz $S[2,2] = 0$

Volvemos a regenerar nuestra matriz fundamental F, realizando el producto punto:

$$F = U * S * V^T$$

y Finalmente denormalizamos la matriz fundamental multiplicando por las matrices de escalamiento T_1 y T_2

$$F = (T_2^T * F * T_1)$$

11. RANSAC

Random sample consensus (RANSAC) és un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos observados que contiene valores atípicos. Es un algoritmo no determinista en el sentido de que produce un resultado razonable sólo con una cierta probabilidad, mayor a medida que se permiten más iteraciones.

Los datos consisten en ïnliers", es decir, los datos cuya distribución se explica por un conjunto de parámetros del modelo, aunque pueden estar sujetos a ruido, y "valores atípicos", que son datos que no encajan en el modelo. Los valores atípicos pueden llegar, por ejemplo, de valores extremos del ruido o de mediciones erróneas o hipótesis incorrectas sobre la interpretación de los datos. RANSAC también asume que, dada un conjunto de inliers (generalmente pequeño), existe un procedimiento que puede estimar los parámetros de un modelo que explica de manera óptima o se ajusta a esta información.

Algoritmo de RANSAC:

1. *Seleccionar un subconjunto aleatorio de los datos originales. Subconjunto llamado ïnliers hipotéticos".*

2. *Un modelo se monta en el conjunto de inliers hipotéticos.*

3. *Todos los demás datos se prueban contra el modelo ajustado. Esos puntos que se ajustan al modelo estimado, de acuerdo con alguna función de pérdida de modelos específicos, se consideran como parte del conjunto de consenso.*

4. *El modelo estimado es bueno si se han clasificado suficientes puntos como parte del conjunto de consenso.*

5. *Después, el modelo puede ser mejorado volviendo a estimar usando todos los miembros del conjunto de consenso.*

Este procedimiento se repite un número de veces con un criterio de paro fijo, o dinámico, produciendo o bien un modelo que es rechazado porque no hay suficientes puntos en el conjunto de consenso, o un modelo aceptado con suficientes puntos en el conjunto de consenso. En este último caso, mantenemos el modelo si su conjunto de consenso es más grande que el modelo guardado previamente.

Para el caso fijo, basta con iterar el algoritmo de ransac un numero N finito de veces, y encontrar la F con la mayor cantidad de inliers. Para el criterio adaptativo de paro, repetiremos el ciclo hasta que el contador de iteraciones sea menor a N, donde N esta dado por:

$$N = \frac{log(1-p)}{log(1-(1-e)^s)}$$

Donde: e es el error actual de los outlayers de que sea outlayer, p es la probabilidad de que de los resultados obtenidos estan realmente bien clasificados, y s es un subconjunto de puntos que voy a tomar

Ahora el criterio para determinar que punto es un inlier, y que punto no lo es, viene dado por varias metricas. En nuestro caso, utilizamos el calculo de la distancia de *sampson*, para calcular la distancia que se proyecta de cada par de correspondencias, a la linea trazada por el conjunto de 8 muestras actuales. Como se muestra en la figura 6:

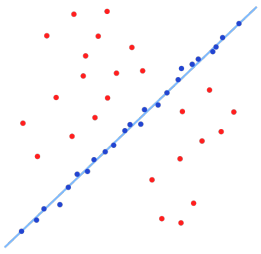

Figura 6: Calculo de inliers con la distancia de SAMPSON.

Donde la distancia de ransac se calcula mediante la siguiente ecuacion:

$$D = \frac{(x'^T Fx)^2}{(Fx_1)^2+(Fx_2)^2+(F^T x'_1)^2+(F^T x'_2)^2}$$

Lo cual nos genera un vector columna de 1 fila por m columnas, donde cada columna almacena la distancia del punto en la posicion i hasta la recta, y todos los puntos cuya distancia sea menor a un umbral u, en nuestro caso .0001, son considerados como inliers.

Al final de este proceso iterativo, que se detiene cuando el criterio adaptativo del numero de veces es menor N, obtendremos una matriz fundamental F la cual concentra la mayor cantidad de inliers en el conjunto de todas las iteraciones de ransac.

12. Cálculo de la matriz K

El modelo pinhole, como habíamos mencionado antes, Los puntos del espacio se proyectan a través de un punto en el plano proyectivo.

La proyección del espacio en R^3 al espacio R^2 se realiza como:

$$(x, y, z)^T \rightarrow (fx/z, fy/z)^T.$$

El centro de la proyección se denomina el *centro de la camara*. La linea perpendicular la plano de la imagen que pasa por el centro de la cámara se denomina eje o rayo principal, y el punto de intersección entre este y el plano de la imagen se llama el *punto principal*. El *plano principal* es el que pasa por el centro de la cámara y es paralelo al plano de la imagen. La forma de la matriz de la cámara K viene de forma explicita en la siguiente matriz:

$$F = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

en donde fx es el foco de X, cx es el centro de x, y de igual forma fy y cy, son el foco de y y el centro de y.

La importancia de esta matríz, que son los parametros intrínsecos de la cámara, es que permiten mapear el conjunto de coordenadas que originalmente se encuentran en pixeles, a metros, es decir la representación real de un pixel en el mundo real.

13. Cálculo de la matríz escencial

La matriz esencial es la especialización de la matriz fundamental al caso en que las coordenadas de la imagen están normalizadas. Coordenadas normalizadas significa que la matriz K, es igual a la identidad. Una matriz de cámara se puede descomponer como $P = K[R|t]$. Si conocemos la matriz de calibracion los puntos proyectados, $x = PX = K[R|t]X$, se pueden representar como

$\hat{x} = K^1 x$, con lo que tenemos $\hat{x} = [R|t]X$. Estos puntos estan en coordenadas normalizadas. Si consideramos el par de matrices $P = [I|0]$ y $P' = [R|t]$, la matriz fundamental, se denomina esencial, y tiene la forma:

$$E = [t]_\times R$$

Se cumple la relacion $\hat{x}^T E \hat{x} = 0$. Si sustituimos los puntos originales, obteemos $x'^T (K'^{-T} E K^{-1}) x = 0$, con lo que la relacion con la matriz fundamental es:

$$E = K'^T F K$$

La matriz esencial tiene 5 grados de libertad.

La cual fue calculada con la matriz K que son los parametros intrinsecos de nuestra camara que para ello utilizamos la utileria calibgui de matlab como se muestra en la figura 7:

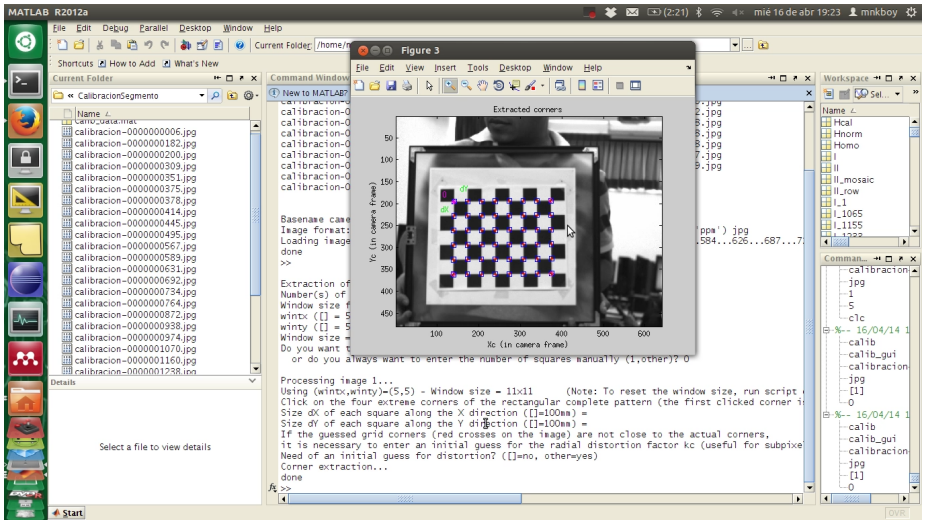

Figura 7: Calibracion de la Matriz K parametros intrinsecos de la camara.

Y cuyos valores son los siguientes para este experimento:

$$K = \begin{bmatrix} 1636{,}54 & 0 & 249{,}71 \\ 0 & 1637{,}23 & 251{,}94 \\ 0 & 0 & 1 \end{bmatrix}$$

14. Calculo de R y t

Ahora mediante una descomposición de valores singulares (SVD), descomponemos nuestra matriz esencial E en nuestra matriz de rotacion R y nuestro vector de traslacion t para nuestra camara 1:

Una descomposición de valores singulares de $\mathbf{E}$ nos da:

$$\mathbf{E} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T$$

donde:

$\mathbf{U}$ y $\mathbf{V}$ son matrices ortogonales 3×3 y $\mathbf{\Sigma}$ es una 3×3 matriz diagonal con:

$$\mathbf{\Sigma} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Las entradas diagonales de $\mathbf{\Sigma}$ son los valores singulares de $\mathbf{E}$ Los cuales de acuerdo a las restricciones internas de la matriz esencial deben consistir en dos matrices idénticas de unos y ceros. Definidas como:

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ con } \mathbf{W}^{-1} = \mathbf{W}^T = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

y finalmente obtenemos R y t.

$$[\mathbf{t}]_\times = \mathbf{V}\,\mathbf{W}\,\mathbf{\Sigma}\,\mathbf{V}^T$$

$$\mathbf{R} = \mathbf{U}\,\mathbf{W}^{-1}\,\mathbf{V}^T$$

y al combinar nuestra matriz de rotacion R, y nuestro vector de traslacion t nos queda nuestra matriz [R—t] de la siguiente manera:

$$P = [R|t] = \begin{bmatrix} r1 & r2 & r3 & t1 \\ r4 & r5 & r6 & t2 \\ r7 & r8 & r9 & t3 \end{bmatrix}$$

Para nuestra cámara 2 no será necesario, al menos para este experimento calcular su matriz $[R|t]$ ya que su forma canónica viene dada por:

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Despues calculamos la inversa de nuestra matriz K osea K^{-1}, la cual nos da un valor de:

$$K = \begin{bmatrix} 0{,}0000611043 & 0 & -0{,}152586 \\ 0 & 0{,}000610785 & -0{,}1538837 \\ 0 & 0 & 1 \end{bmatrix}$$

Para continuar al siguiente paso que sera calcular los puntos tridimensionales atravez del algoritmo de triangulacion linear.

15. Triangulación Linear

Una vez teniendo nuestras matrices $P = [R|t]$ debemos a partir de nuestros vectores de puntos de interes triangularos, y obtener de dos coordenadas bidimensionales, una coordenada Tridimensional.

Esto último lo lograremos mediante el algoritmo de triangulación, que depende de dos puntos bidimensionales que son correspondencias entre la imagen uno, y la imagen dos, y nuestras matrices $P' = [R'|t']$ de la cámara uno, y $P = [R|t]$ de nuestra cámara dos para formar un punto tridimensional que formará parte de nuestra nube de puntos.

En cada imagen nosotros tenemos una relacion de $x = PX$, $x' = P'X$ estas ecuaciones pueden ser combinadas en una forma $AX = 0$ la cual es una ecuacion linear en X.

Por ejemplo para nuestra imagen, $x \times (PX) = 0$ y escribiendo esto nos da:

$$\begin{gathered} x(p^{3T}X) - (p^{1T}X) = 0 \\ y(p^{3T}X) - (p^{2T}X) = 0 \\ x(p^{2T}X) - y(p^{1T}X) = 0 \end{gathered}$$

donde p^{iT} son las filas de P. Estas ecuaciones son lineares en las componentes de X. Una ecuación de la forma AX = 0 puede ser compuesta con:

$$A = \begin{bmatrix} xp^{3T} & -p^{1T} \\ yp^{3T} & -p^{2T} \\ x'p'^{3T} & -p'^{2T} \\ y'p'^{3T} & -p'^{2T} \end{bmatrix}$$

Donde dos ecuaciones tienen que ser incluidas por cada imagen, dando un total de cuatro ecuaciones en cuatro homogeneas conocidas.

Basados en ese ejemplo, entonces nos disponemos a construir nuestro vector A y nuestro vector B, para obtener nuestro valor X. Para esto fijaremos a nuestro punto u como el punto de nuestra imagen 1 y u1 como el punto en nuestra imagen 2, y nuestra Matriz P como nuestra Matriz de rotacion y traslacion, es decir $P = [R|t]$ y $P1 = [R'|t']$ sera la matriz P de nuestra cámara dos.

$$A = \begin{bmatrix} [u_x * P(2,0) - P(0,0)] & [u_x * P(2,1) - P(0,1)] & [u_x * P(2,2) - P(0,2)] \\ [u_y * P(2,0) - P(1,0)] & [u_y * P(2,1) - P(1,1)] & [u_y * P(2,2) - P(1,2)] \\ [u1_x * P1(2,0) - P1(0,0)] & [u1_x * P1(2,1) - P1(0,1)] & [u1_x * P1(2,2) - P1(0,2)] \\ [u1_y * P1(2,0) - P1(1,0)] & [u1_y * P1(2,1) - P1(1,1)] & [u1_y * P1(2,2) - P1(1,2)] \end{bmatrix}$$

$$B = \begin{bmatrix} [-(u_x * P(2,3) - P(0,3)], & [-(u_y * P(2,3) - P(1,3)], \\ [-(u1_x * P1(2,3) - P1(0,3)], & [-(u1_y * P1(2,3) - P1(1,3))] \end{bmatrix}$$

Y aplicando la descomposicion de valores singulares a A, y a la Matriz B, obtenemos x donde:

$\text{SVD}(A) = [U_A, W_A, V'_A] =$

$$[U_A, W_A, V'_A] = \begin{bmatrix} \sigma & 0 & 0 & 0 \\ 0 & \sigma & 0 & 0 \\ 0 & 0 & \sigma & 0 \\ 0 & 0 & 0 & \sigma \end{bmatrix} = A^t = V * \begin{bmatrix} 1/\sigma & 0 & 0 & 0 \\ 0 & 1/\sigma & 0 & 0 \\ 0 & 0 & 1/\sigma & 0 \\ 0 & 0 & 0 & 1/\sigma \end{bmatrix} * U^t$$

Y obtenemos x luego de la descomposicion de valores singulares de A al multiplicarla por la matriz B:

$x = AB$

Donde x es nuestra coordenada en 3 dimensiones que servira para almacenarla en nuestra nube de puntos.

16. Resultados del experimento

Comenzando por lo resultados de ransac como se aprecian en la figura 8 con un total de 64 iteraciones con un error acumulado de 1.0873766076776592, y 324 inliers de 429 correspondencias.

```
('it: ', 56, 'inliers: ', 283, '/', 429, 'Error: ', 0.93479302804567743, 'N: ',
355.2197)
('it: ', 57, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 7
4.030495)
('it: ', 58, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 2
21.59891)
('it: ', 59, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 2
93.20847)
('iteracion numero: ', 60, 'Cantidad inliers / Tamano de coorrespondencias: ', 3
02, '/', 429, 'Error de la muestra: ', 1.2957986682271454, 'Criterio N: ', 14674
130.0)
('it: ', 60, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 1
4674130.0)
('it: ', 61, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 1
5060.547)
('it: ', 62, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 1
00844.77)
('it: ', 63, 'inliers: ', 302, '/', 429, 'Error: ', 1.2957986682271454, 'N: ', 2
35.64433)
('it: ', 64, 'inliers: ', 324, '/', 429, 'Error: ', 1.0873766076776592, 'N: ', 4
1.159977)
('iteracion numero: ', 64, 'Cantidad inliers / Tamano de coorrespondencias: ', 3
24, '/', 429, 'Error de la muestra: ', 1.0873766076776592, 'Criterio N: ', 41.15
9977)
```

Figura 8: Resultados de las iteraciones de ransac.

Obtuvimos una matriz Fundamental Denormalizada:

$$F = \begin{bmatrix} 9{,}82955271e-06 & 2{,}00005188e-04 & 1{,}04563508e-01 \\ -1{,}88690595e-04 & 1{,}07122429e-06 & 4{,}26846179e-02 \\ -1{,}12450448e-01 & -4{,}51450648e-02 & 1{,}00000000e+00 \end{bmatrix}$$

Obtuvimos una matriz Esencial Denormalizada:

$$E = \begin{bmatrix} 1{,}15428948e+00 & 2{,}34966571e+01 & 1{,}12948561e+01 \\ -2{,}21674160e+01 & 1{,}25900968e-01 & -2{,}98939448e-01 \\ -1{,}13039780e+01 & 3{,}63887610e-01 & -8{,}61957722e-03 \end{bmatrix}$$

Obtuvimos una matriz de Rotación positiva

$$R = \begin{bmatrix} -0{,}99967553 & 0{,}02418446 & 0{,}007997 \\ -0{,}02398283 & -0{,}99941418 & 0{,}0244154 \\ -0{,}00858279 & -0{,}02421569 & -0{,}99966991 \end{bmatrix}$$

Obtuvimos una matriz de T gorrito positiva

$$\hat{T} = \begin{bmatrix} 0{,}49533595 & -22{,}16087754 & -11{,}30904164 \\ 23{,}23480708 & -0{,}39851133 & 0{,}09278356 \\ 11{,}87002263 & -0{,}48605029 & -0{,}09682462 \end{bmatrix}$$

Vector de traslacion T

$$T = \begin{bmatrix} -0{,}48605029 & -11{,}30904164 & 23{,}23480708 \end{bmatrix}$$

Matriz RT $P0_1$

$$P0_1 = \begin{bmatrix} -9{,}99675527e-01 & 2{,}41844614e-02 & 7{,}99700244e-03 & -4{,}86050293e-01 \\ -2{,}39828255e-02 & -9{,}99414185e-01 & 2{,}44154025e-02 & -1{,}13090416e+01 \\ -8{,}58279103e-03 & -2{,}42156896e-02 & -9{,}99669914e-01 & 2{,}32348071e+01 \end{bmatrix}$$

Matriz RT $P0_2$ o tambien la llamada canonica $P[I|0]$

$$P0_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Al plotear la nube de puntos en el programa matlab obtenemos una reconstruccion parcial de nuestra imagen original del craneo, puesto que se aprecia la profundidad en los puntos como se muestra a continuación:

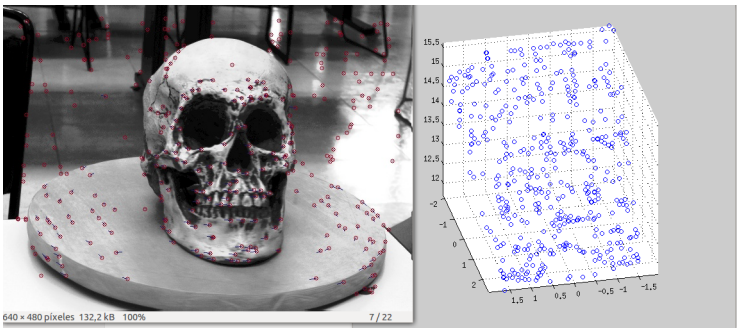

Figura 9: Nube de puntos producto de la reconstruccion 3D.

16.1. Prueba inversa

Para realizar la prueba inversa de nuestro sistema primero generamos 200 números aleatorios con los siguientes rangos:

$$\begin{gathered}\text{rango: X } -2, +2\\ \text{rango: Y } -1{,}5, +1{,}5\\ \text{rango: Z } 1{,}5, 5{,}0\end{gathered}$$

Lo cual nos dara un vector columna de muestras aleatorios (VMA) de las siguientes caracteristicas:

$$VMA = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Creamos un vector de traslacion T con .15 cm de moviento en Y

$$T = \begin{bmatrix} 0 & 0{,}15 & 0 \end{bmatrix}$$

Creamos una matriz de rotacion R con 15 grados de rotacion

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0{,}96592567 & -0{,}25881964 \\ 0 & 0{,}25881964 & 0{,}96592567 \end{bmatrix}$$

Creamos una matriz de $\hat{T}$ con 15 grados de rotacion

$$\hat{T} = \begin{bmatrix} 0. & -0. & 0{,}15 \\ 0. & 0. & -0. \\ -0{,}15 & 0. & 0. \end{bmatrix}$$

Creamos una $P0_1$ de $\hat{T}$ con 15 grados de rotacion

$$P0_1 = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & 0{,}96592567 & -0{,}25881964 & 0{,}15 \\ 0. & 0{,}25881964 & 0{,}96592567 & 0. \end{bmatrix}$$

Con las cuales realizamos la transformacion de los puntos 3D a 2D multiplicando la matriz $P0_1$ por el vector uno y la matriz $P0_2$ por el vector 2

$$\begin{gathered}P0_1 * puntosUno\\ P0_2 * puntosDos\end{gathered}$$

Y dividimos las 3 filas entre la fila 3 para obtener un vector columna de 3x200 donde la tercela fila para obtener un vector con X,Y,1

$$V2D = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Finalmente metemos nuestros vectores 2D en nuestro algoritmo de ransac, para obtener una matriz fundamental F que cumpla con la restriccion epipolar

Resultado Ransac desde la primera iteracion

('it: ', 1, 'inliers: ', 200, '/', 200, 'Error: ', 7,1848$e-49$, 'N: ', 1)

Con lo cual se comprueba que todos los puntos son inliers

La determinante de la matriz F:
np.linalg.det(FDenormalizadaDividida)
$Det(F) = 5{,}09950036094e-24$
La determinante de U: np.linalg.det(U)
$Det(U) = 1$
Con lo cual se comprueba que la matriz fundamental es, efectivamente una matriz fundamental.

Y finalmente y la prueba de fuego, la restriccion epipolar:

```
Comprobando la restriccion epipolar
-3.12250225676e-17
-9.36750677027e-17
-6.76542155631e-17
2.53269627493e-16
1.64798730218e-16
-2.04697370165e-16
1.5092094241e-16
3.98986399475e-16
-5.20417042793e-18
1.05818132035e-16
```

Figura 10: Restriccion epipolar de la prueba inversa.

Donde $X_2^T F X_1 = 0$

Referencias

[1] Hartley, Zisserman - Multiple View Geometry in Computer Vision

[2] Kenichi Kanatani Statistical Optimization for Geometric Computation Theory and Practice 2005

[3] An Invitation to 3D Image - Ma, Soatto

[4] Mastering OpenCV with practical computer vision proyects, Daniel Lelis Baggio

[5] Linear Algebra, Stanley Grossman

[6] Visión por computador, Departamento de Inteligencia Artificial Universidad politecnica de Madrid

[7] RANSAC, Václav Hlavác, Czech Technical University

[8] OpenCV Computer Vision with Python, Joseph Howse

[9] HOMOGRAPHY-BASED PLANE IDENTIFICATION AND MATCHING, Amirhasan Amintabar and Boubakeur Boufama

[10] HOMOGRAPHY-BASED PLANE IDENTIFICATION AND MATCHING, Amirhasan Amintabar and Boubakeur Boufama