Estéreo Visión

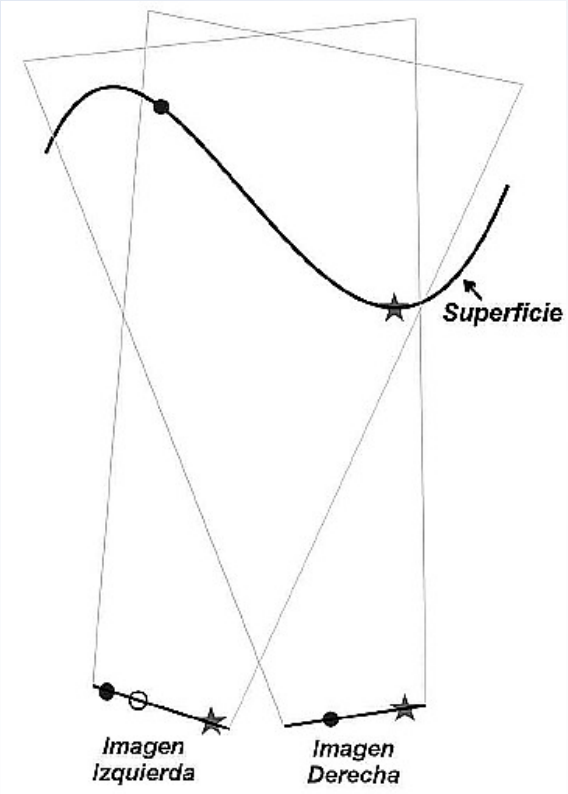
Pablo Morales Pimentel
Estudiante Ingeniería Civil Electrónica
UTFSM

Introducción

- **Visión estéreo:** Es el proceso mediante el cual se puede crear una representación en 3D a partir de dos imágenes en 2D.
- Es una característica muy común en el mundo animal.
- Al emular ésta característica, se puede por ejemplo calcular mapas de elevación, dotar de visión a robots, generar modelos tridimensionales, etc.
- En éste trabajo se investigó e implementó una manera de hacer el cálculo.

Marco Teórico

- Criterio de Correlación
  - Imagenes acotadas epipolarmente
  - Permite implementar la visión estéreo
  - Función costosa computacionalmente

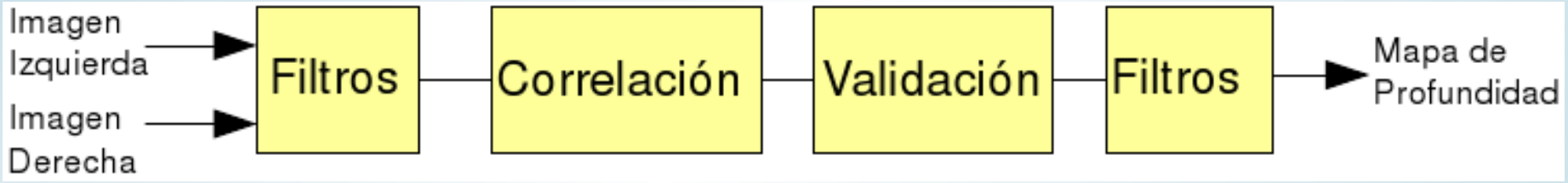


$$F(I,D) = \max_{d=-\delta_l \ldots \delta_h} \frac{\sum_{i,j} I(x+i,y+j)D(x+d+i,y+j)}{\sqrt{\sum_{i,j} I(x+i,y+j)^2}\sqrt{\sum_{i,j} D(x+d+i,y+j)^2}}$$

$$i = -w \ldots w, \qquad j = -h \ldots h.$$

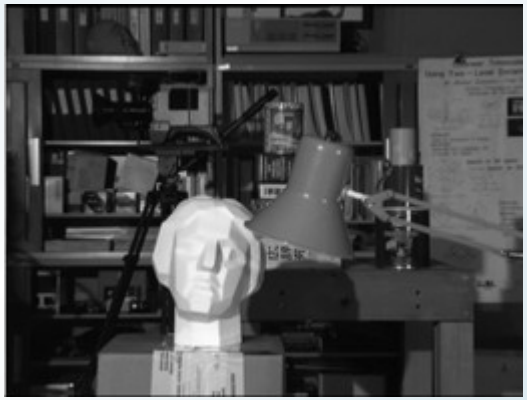
Implementación

- Diagrama



Implementación

- Adquisición de imágenes

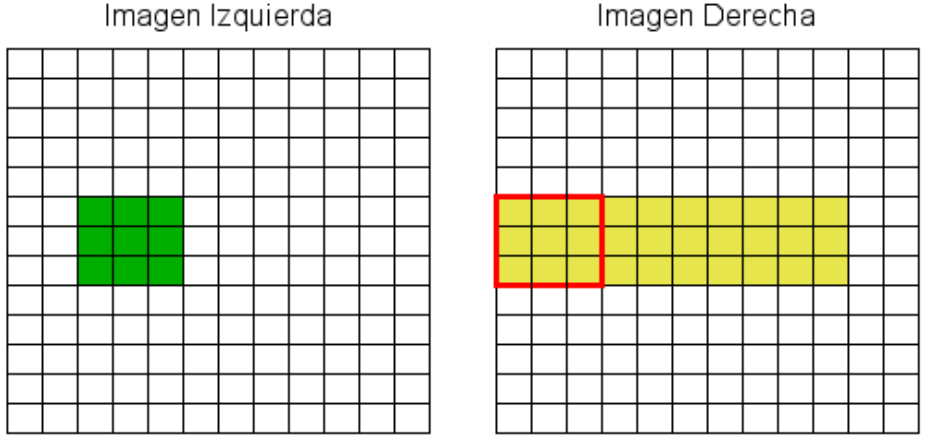


- Filtrado

Implementación

- Correlación

$$F(I,D) = \max_{d=-\delta_l \ldots \delta_h} \frac{\sum_{ij} I(x+i,y+j)D(x+d+i,y+j)}{\sqrt{\sum_{ij} I(x+i,y+j)^2}\sqrt{\sum_{ij} D(x+d+i,y+j)^2}}$$

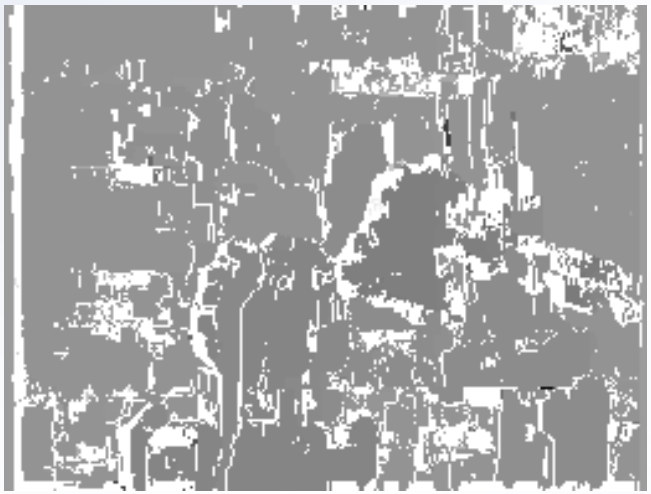


Tiempo promedio: 2 minutos

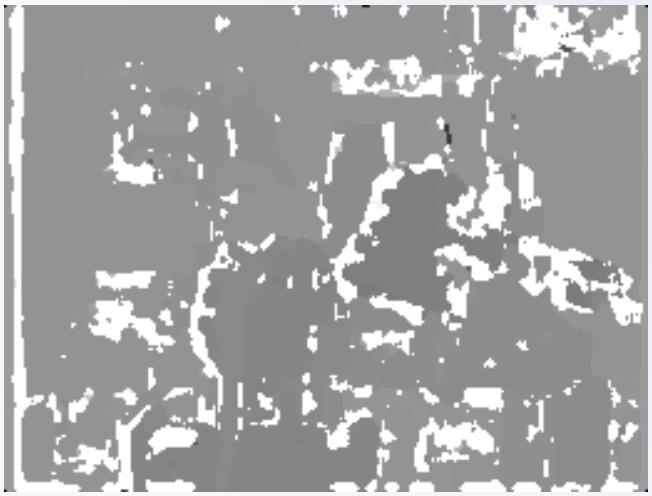
Tiempo promedio: 1' 15” aprox

Implementación

- Validación
  - Consiste en calcular la correlación de la imagen derecha respecto a la izquierda, es decir al revés del paso anterior. Los puntos que no coinciden se marcan de blanco y corresponden a los puntos de oclusión.

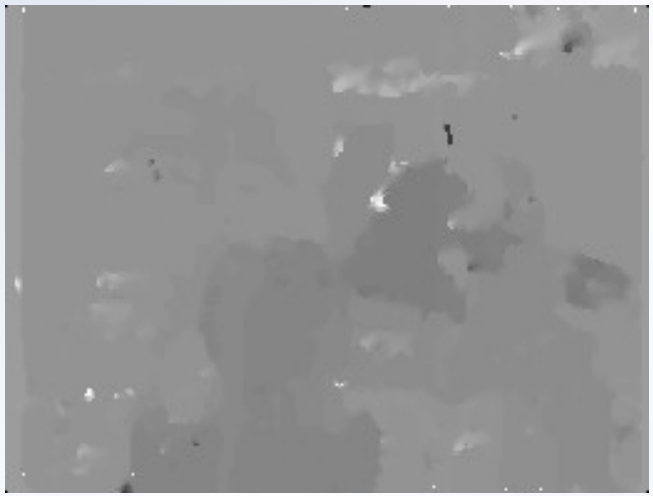


Filtro de Mediana

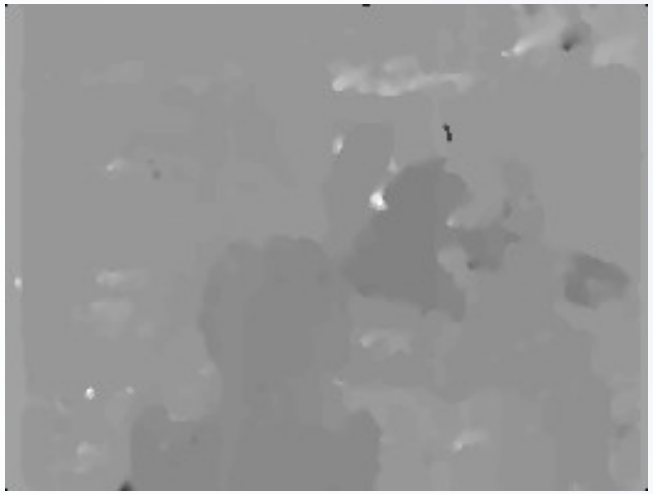


Implementación

- Relleno



- Filtros y ajuste de contraste:



Implementación

- C++

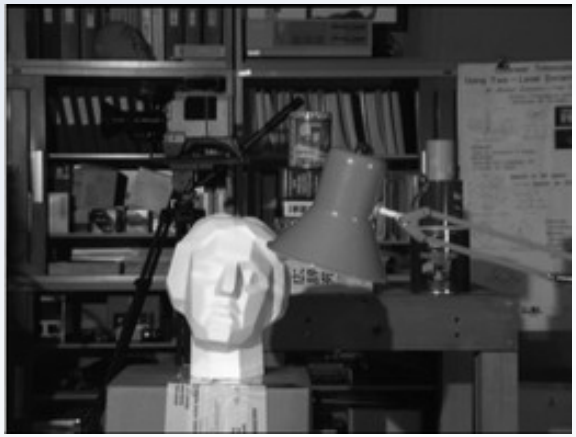
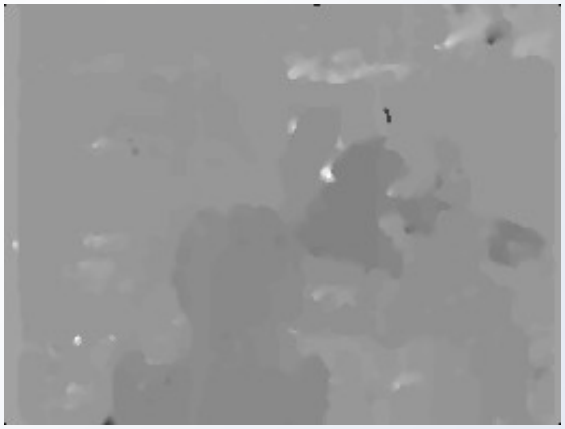
```
class CORRELATION
{
    public:
        CORRELATION(const char *izq, const char *der, const char *output);
        void maximizar();                           // maximizar C2
        void minimizar();                           // minimizar C1
        void mediana();
        void fillin();
        void fillin2();
        void contraste();
        void filtraInputs();
        void setsize();
        void setsize(int a, int b);
        void guardar();

    private:
        void filtro(pngwriter &im, string name);
        float calcularC1(int x, int y, int d, pngwriter &I, pngwriter &D);  // basada en diferencia
        float calcularC2(int x, int y, int d, pngwriter &I, pngwriter &D);  // basada en multiplicacion
        pngwriter Imagen, izq, der;
        int X;
        int Y;
        string nizq, nder;
        const static int WSIZEX = 7;                // Tamaño de ventana de correlación
        const static int WSIZEY = 7;                // Tamaño de ventana de correlación
        const static int WSEARCH = 10*WSIZEX;       // Tamaño de ventana de busqueda
};
```

Implementación

- C++
  - PNGwriter
    - Permite manipular imágenes de manera intuitiva, por ejemplo:
      - imagen.plot(x, y, R,G,B);
      - imagen.read(x,y);
      - imagen.readfromfile("archivo.png");
      - imagen.scale_k(int k);
      - Figuras básicas como líneas, triangulos, cuadrados, diamantes, flechas, cruces, poligonos, textos, etc...
    - Requiere la librería `libpng`
    - Más información en http://pngwriter.sourceforge.net/

Resultados


Conclusiones

- Problema costoso computacionalmente, ya que es $O(n^3)$.
- Trabajar basado en librerías (Clases) ya creadas facilita bastante la tarea.
- Trabajo a futuro
  - Realizar el cálculo del mapa de profundidades en tiempo real, es decir a 25 ó 30 cuadros por segundo aproximadamente. Ésto se podría realizar utilizando redes neuronales trabajando en forma paralela.
  - Convertir el mapa de profundidades en estímulos auditivos estéreos.
  - Teóricamente, una persona no vidente podría “ver” de ésta manera.