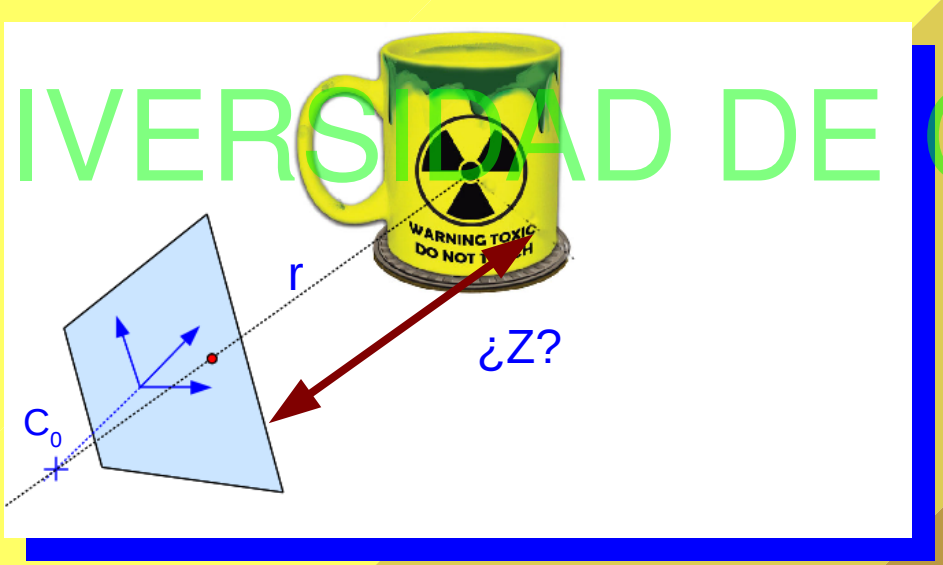
FSIV

Reconstrucción 3D: Visión Estéreo.

Reconstrucción 3D

- Problema: al proyectar un punto 3D, se perdió la profundidad.



$$x = P X$$
$$\vec{v} = P^{+} x$$
$$r = C_0 + \lambda \vec{v}$$

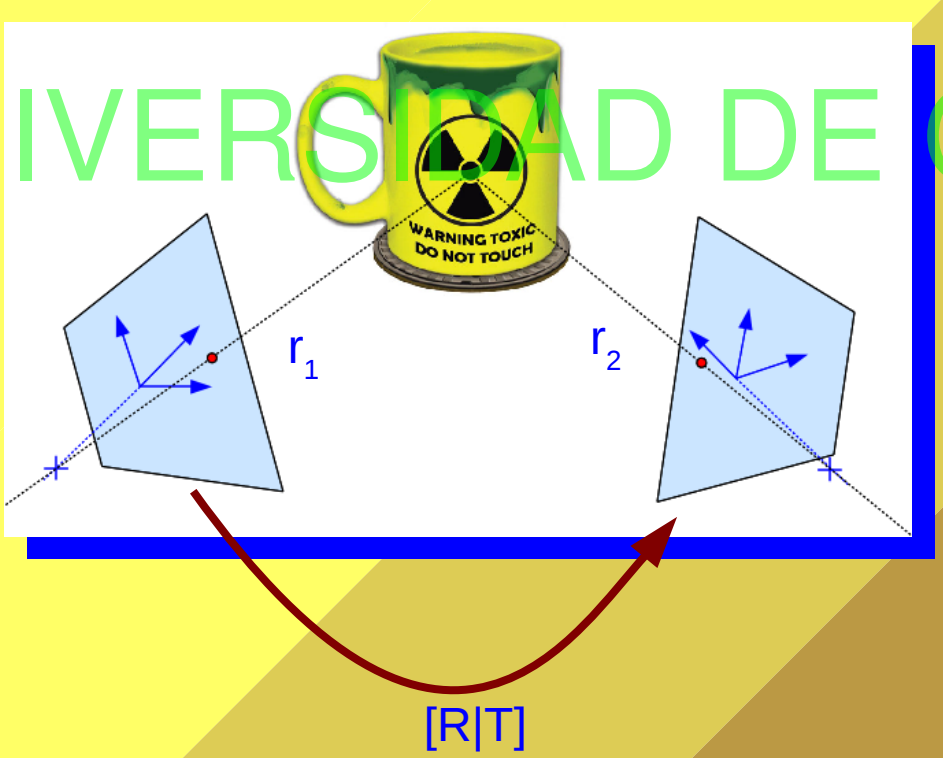
La matriz P^{+} se denomina pseudo-inversa:

$$P^{+} P = I$$

Con la calibración, sabiendo que x es dónde se proyectó un punto 3D X, solo puedo recuperar el rayo 3D r dónde estaría el punto.

Reconstrucción 3D

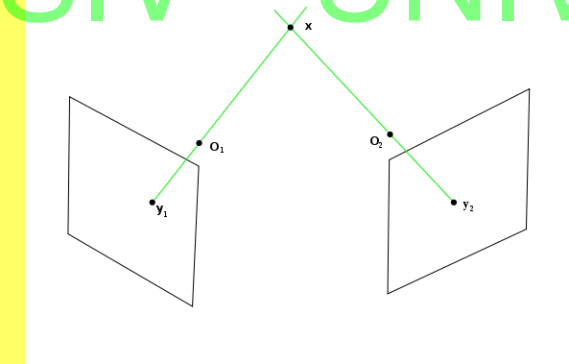
- Solución: buscar la imagen del punto 3D en al menos dos vistas relacionadas diferentes de la escena.

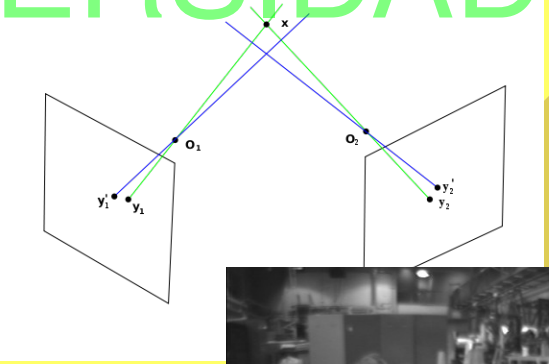


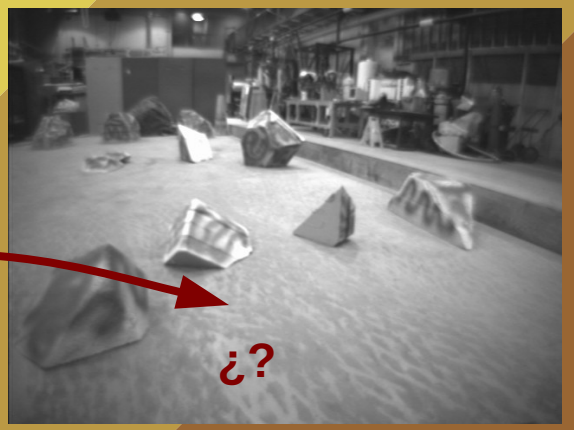
$$\begin{cases} r_1 = C_1 + \lambda \vec{v}_1 \\ r_2 = C_2 + \lambda \vec{v}_2 \end{cases}$$

Reconstrucción 3D

- En la práctica:
  - Necesitamos aproximar debido al ruido.
  - ¿Dónde está la correspondencia de un punto en la otra imagen?





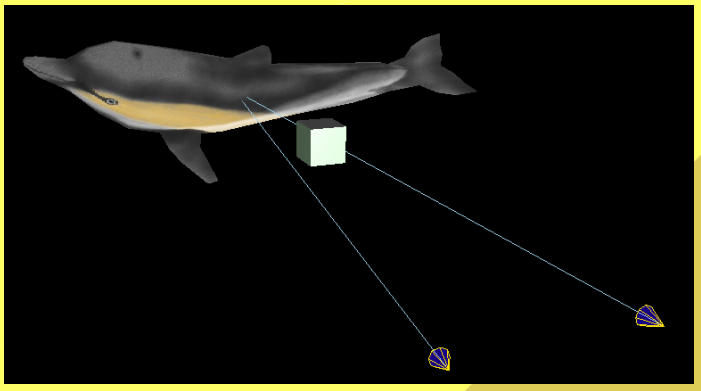


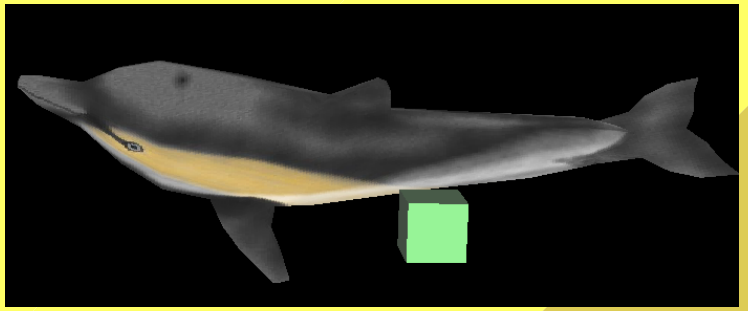
Visión Estéreo

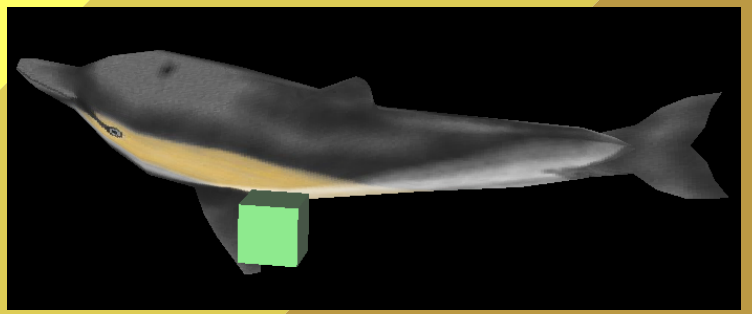
- Contenidos:
  - Concepto.
  - Geometría epipolar.
  - Rectificación.
  - Correspondencia estéro.

Visión estéreo

- Concepto:
  - *"refiere a la percepción de la profundidad y la estructura 3D obtenida en base a la información visual obtenida por dos ojos ..." (wikipedia).*
  - La distancia entre los dos ojos ("base line") es normalmente pequeña comparada con la profundidad, así que las dos imágenes no serán muy diferentes.

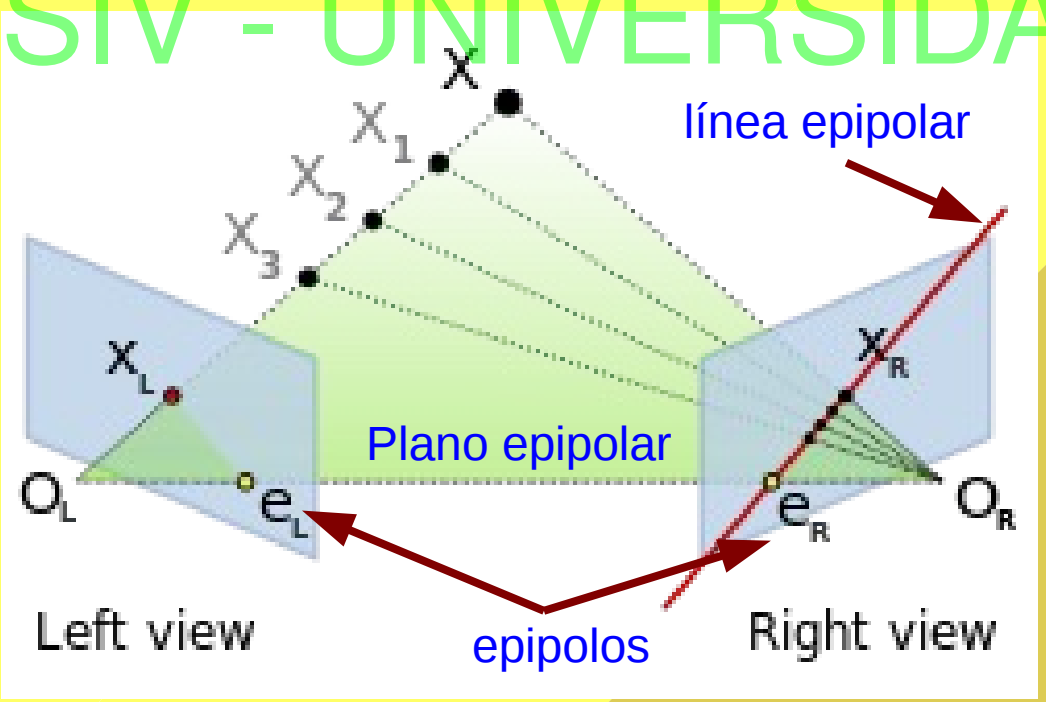


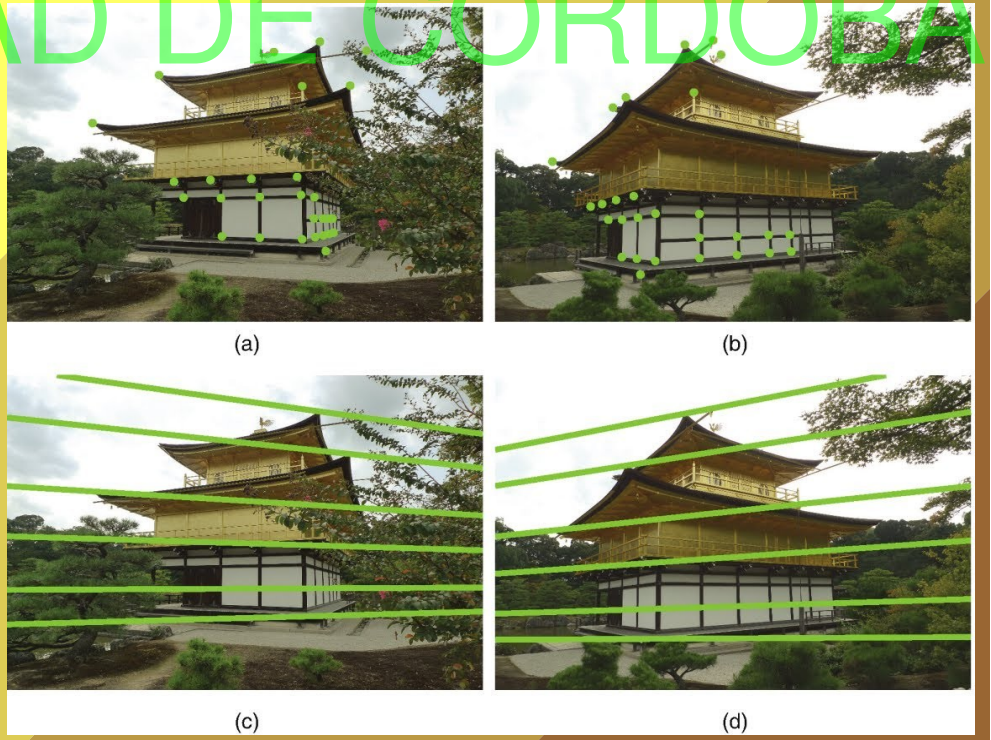




Visión estéreo

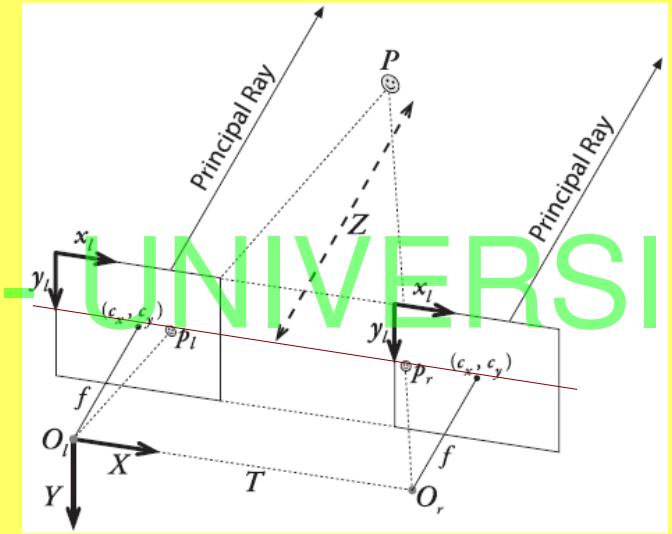
- Geometría epipolar
  - Todo punto 3D visto está restringido en un *plano epipolar*.
  - Dada una proyección de un punto en una imagen, su correspondiente en la otra debe estar en la *línea epipolar*.
  - Así reducimos la búsqueda $O(N^2)$ a $O(N)$.



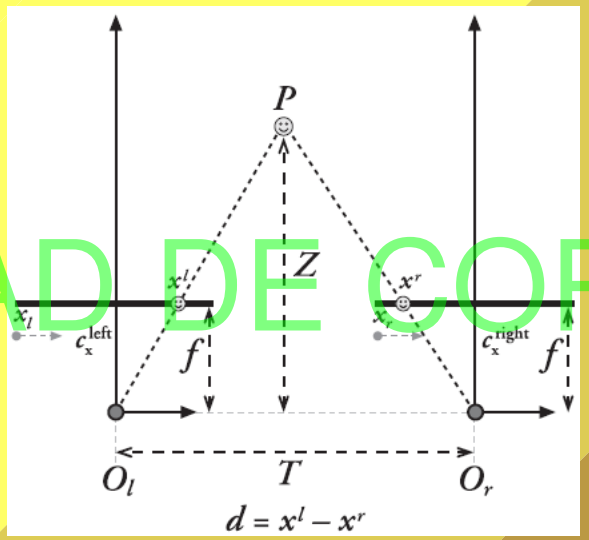


Visión estéreo

- Geometría epipolar: la situación ideal.

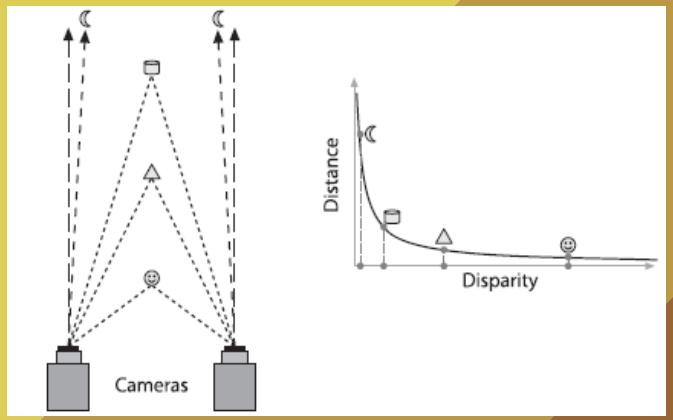


(tomada de "learning opencv", Bradski et al. O'Reilly)



(tomada de "learning opencv", Bradski et al. O'Reilly)

$$\frac{T-(x^l-x^r)}{Z-f}=\frac{T-d}{Z-f}=\frac{T}{Z}\rightarrow Z=f\frac{T}{d}$$



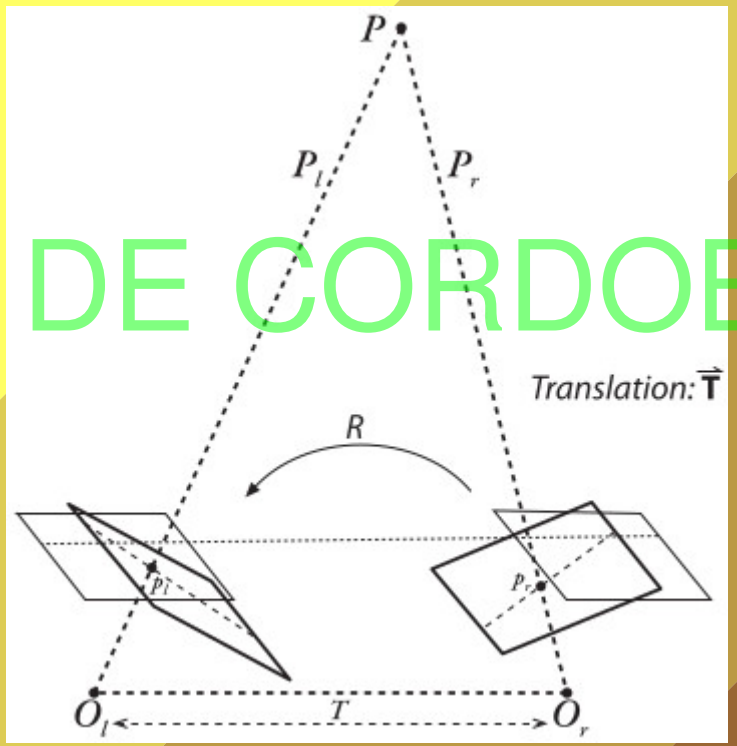
(tomada de "learning opencv", Bradski et al. O'Reilly)

Visión estéreo

- Rectificación.

  Aplicar proceso matemático para conseguir la situación ideal:

  1. Rotar ambas cámaras para que miren perpendicularmente a la línea que une los dos centros proyectivos c_0 y c_1.
  2. Rotar los ejes ópticos de tal forma que los ejes horizontales se se alineen.
  3. Si es necesario, escalar la imagen más pequeña tal que ambas imágenes tengan la misma resolución (y por lo tanto con correspondencia línea-a-línea).



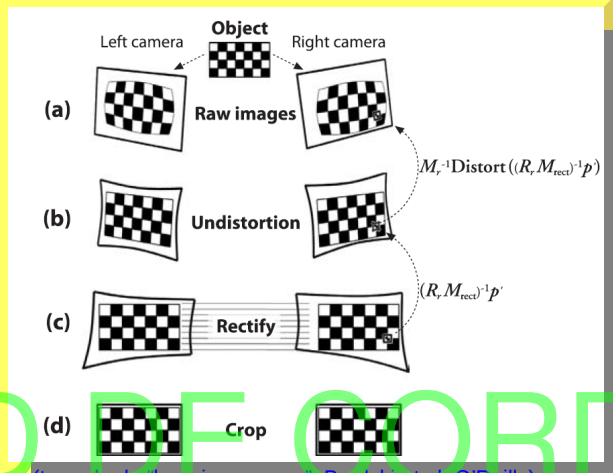
(tomada de “learning opencv”, Bradski et al. O’Reilly)

Visión estéreo

- Rectificación: OpenCV.

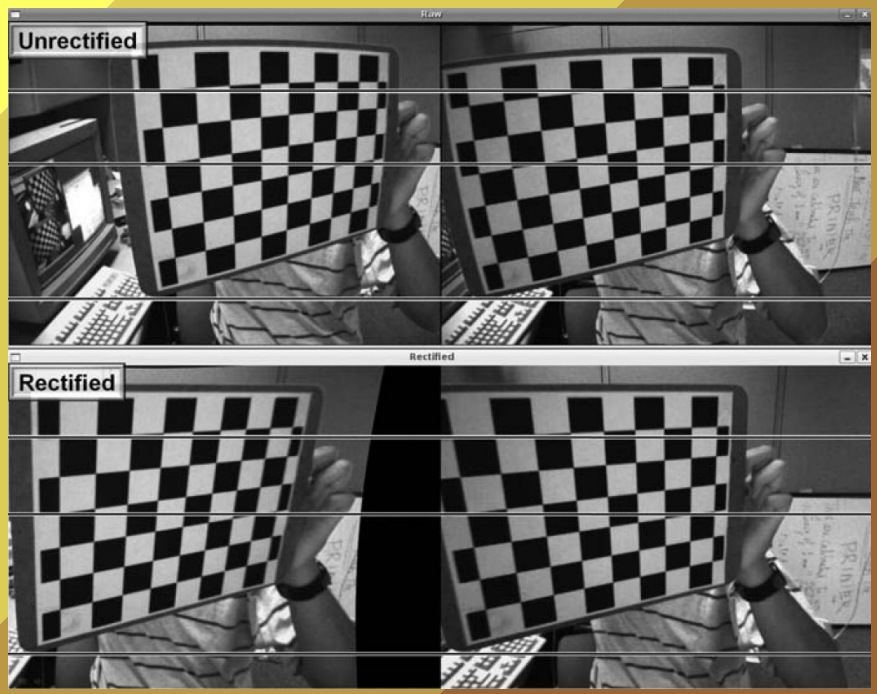
  1. Usando un patrón de calibración, tomar una secuencia de imágenes con ambas cámaras a la vez.
  Usar **cv::stereoCalibrate()** para obtener los parámetros intrínsecos de cada cámara y la pose [R|t] de la cámara derecha respecto a la izquierda (referencia).

  2. Capturadas dos imágenes,
  usar **stereoRetify()** para obtener las versiones rectificadas.



(tomada de "learning opencv", Bradski et al. O'Reilly)

Pasos para crear un sistema estéreo



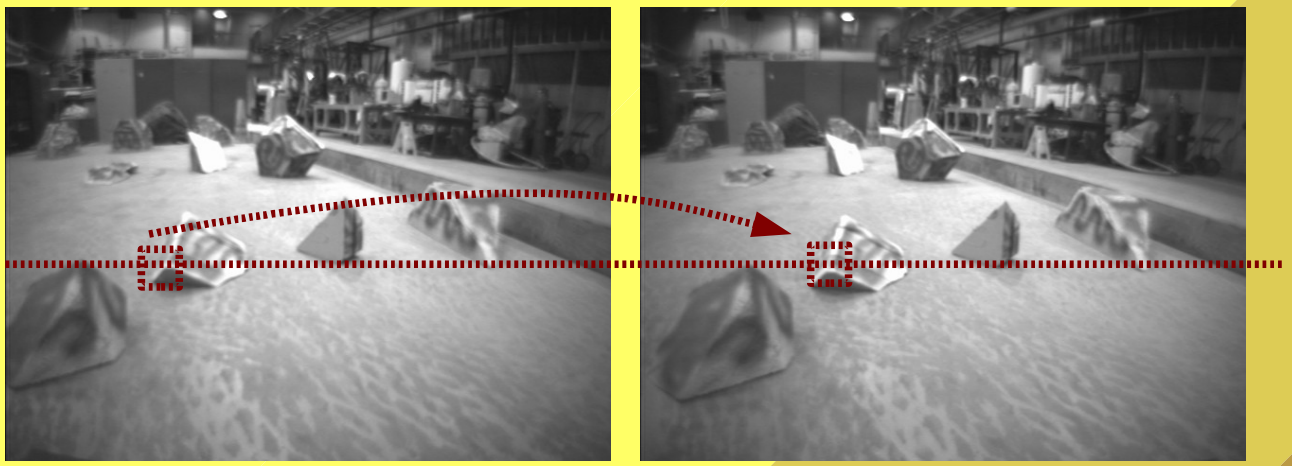
(tomada de "learning opencv", Bradski et al. O'Reilly)

Visión estéreo

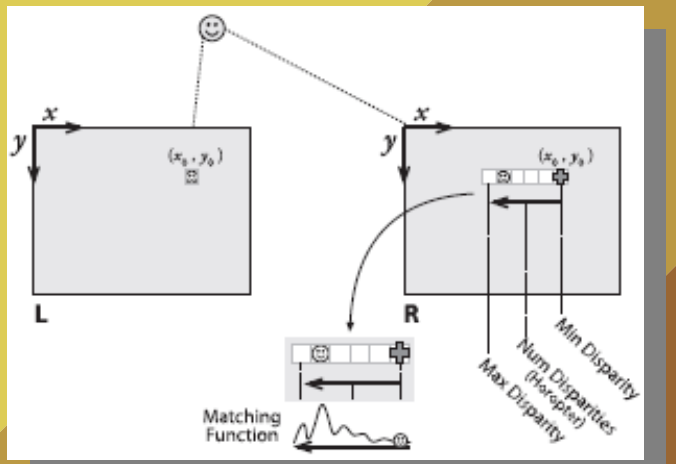
- Correspondencia Estéreo.

La búsqueda de pares se ha reducido a 1D.
Algoritmo ejemplo [Konolige97]:

- Paso 1: procesar la imagen para normalizar brillo y resaltar textura.
- Paso 2: Para cada línea buscar correspondencias usando SAD
- Paso 3: Filtrar para eliminar malas correspondencias.



(tomada de “learning opencv”, Bradski et al. O’Reilly)

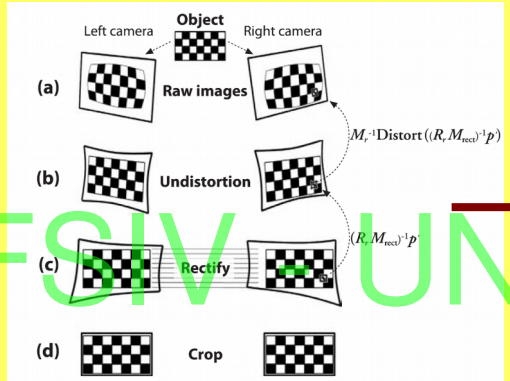


(tomada de “learning opencv”, Bradski et al. O’Reilly)

$$SAD(i,j,d)=\sum_{h=-1}^{1}\sum_{k=-1}^{1}\left|I_l(i+h,j+k)-I_r(i+h,j+k-d)\right|$$

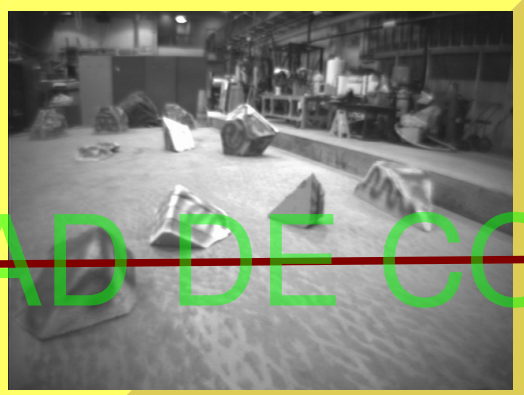
Visión estéreo

- Resumiendo.

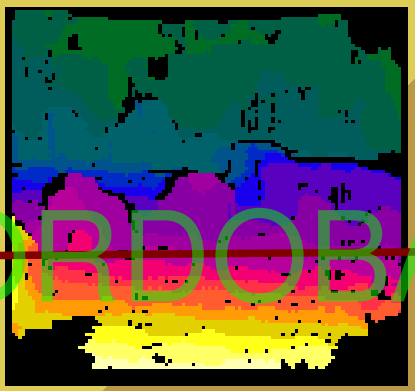


1. Calibración Stereo.
2. Rectificación

3. Correspondencia
StereoBinaryBM class.



4. Mapa de disparidad



(tomada de "learning opencv", Bradski et al. O'Reilly)

$$\frac{T-d}{Z-f}=\frac{T}{Z}\rightarrow Z=f\frac{T}{d}$$

5. Reconstrucción.
cv::reprojectImageTo3D()

Demo:
https://youtu.be/wYNafaF4h5U

Referencias

- Richard Szeliski, "*Computer Vision: Algorithms and Applications*", Springer, 2011.
- Adrian Kaehler and Gary Bradski, *"Learning OpenCV 3"*, O'Reilly, 2017.