Universidad de Oviedo

INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Algoritmos de Visión Artificial con Matlab

Sesión 5. Calibración, info 3D, OpenCV

Ignacio Alvarez García

Rafael C. González de los Reyes

Indice

- Estructura del curso
- Calibración de cámaras
- Obtención de información 3D
- Programación C/C++ con OpenCV
- Las nuevas técnicas: Inteligencia Artificial
- Conclusiones

Indice

- **Estructura del curso**
- Calibración de cámaras
- Obtención de información 3D
- Programación C/C++ con OpenCV
- Las nuevas técnicas: Inteligencia Artificial
- Conclusiones

Estructura del curso

Sesión 1 (3h)

- Sesión 1.1
  - Introducción a la visión por computador.
  - Elementos de un sistema de visión por computador.
  - Etapas del procesamiento de imágenes.
  - Formatos de almacenamiento de imágenes en memoria y disco.
  - Funcionalidades básicas de Matlab para la manipulación de imágenes.
- Sesión 1.2
  - Preprocesamiento de imágenes: introducción.
  - Mejora de contraste.
  - Zoom e interpolación.
  - Filtrado de ruidos.
  - Ejemplos con Matlab.
- Sesión 1.3
  - Resaltado de bordes.
  - Binarización y segmentación.
  - Operaciones con imágenes binarizadas.
  - Ejemplos con Matlab.

Sesión 2 (2h)

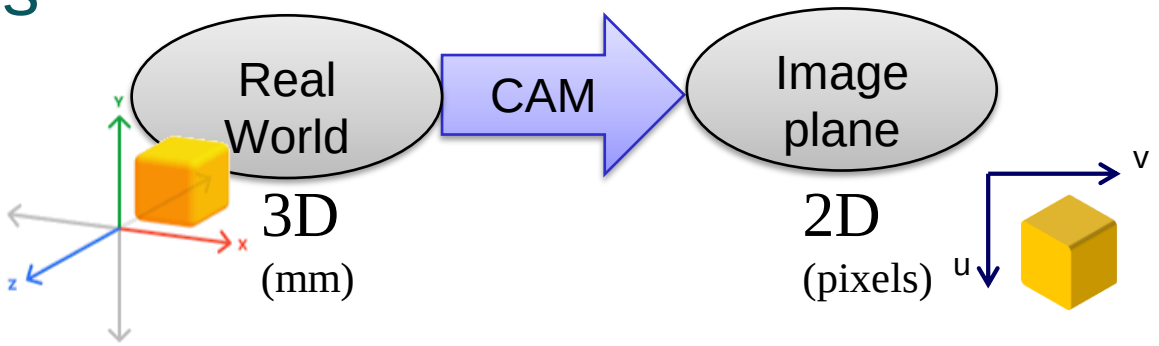
- Sesión 2.1
  - Búsqueda y ajuste de rectas.
  - Obtención de regiones.
  - Descriptores de regiones.
  - Uso de los descriptores.
  - Ejemplos con Matlab.
- Sesión 2.2
  - Calibración de cámaras.
  - Obtención de información 3D
- Sesión 2.3
  - Programación C/C++ con OpenCV
  - Inteligencia Artificial
  - Conclusiones

Indice

- Estructura del curso
- **Calibración de cámaras**
- Obtención de información 3D
- Programación C/C++ con OpenCV
- Las nuevas técnicas: Inteligencia Artificial
- Conclusiones

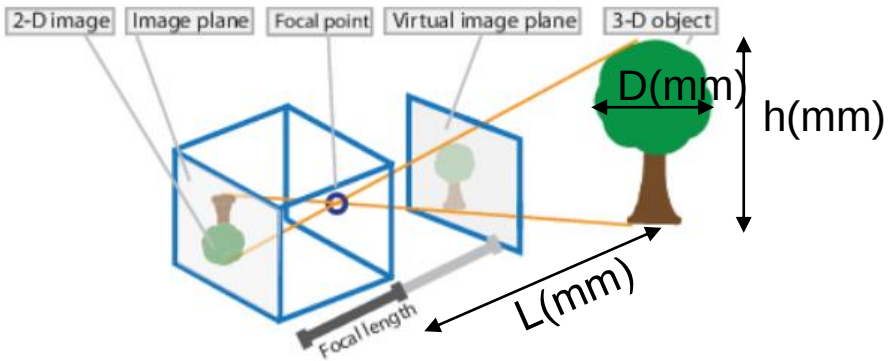
Calibración de cámaras

- Calibrar la cámara es encontrar el modelo y sus parámetros

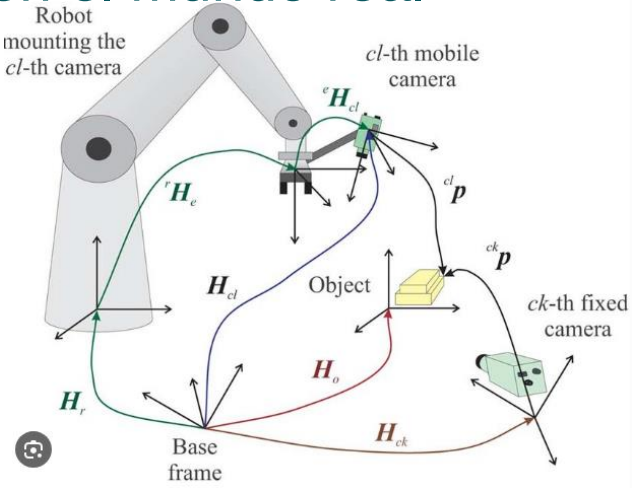


- ¿Para qué?

Permite reconstruir el mundo real en sus unidades

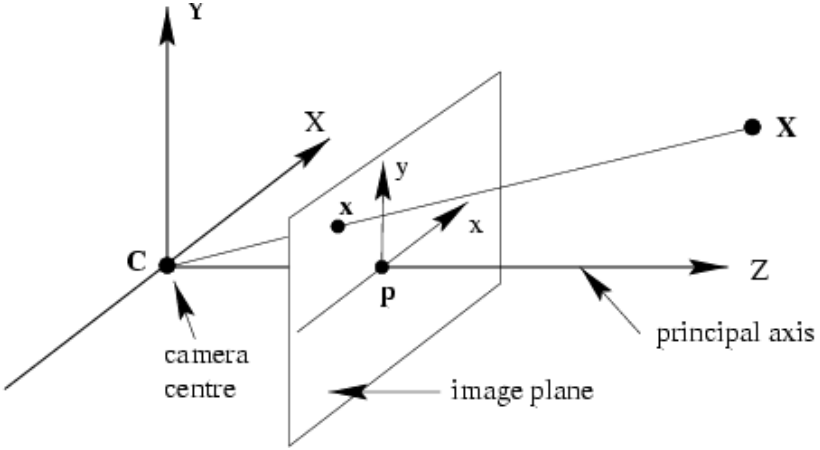


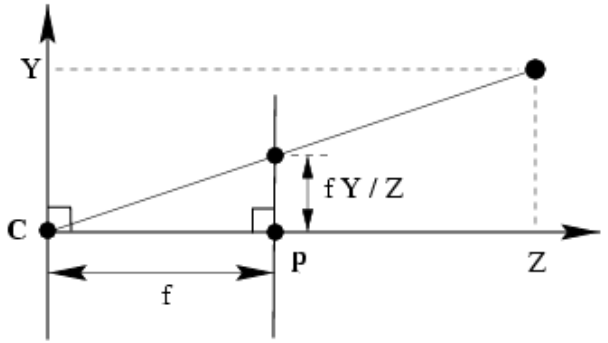
Permite obtener coordenadas en el mundo real



Modelo de cámara

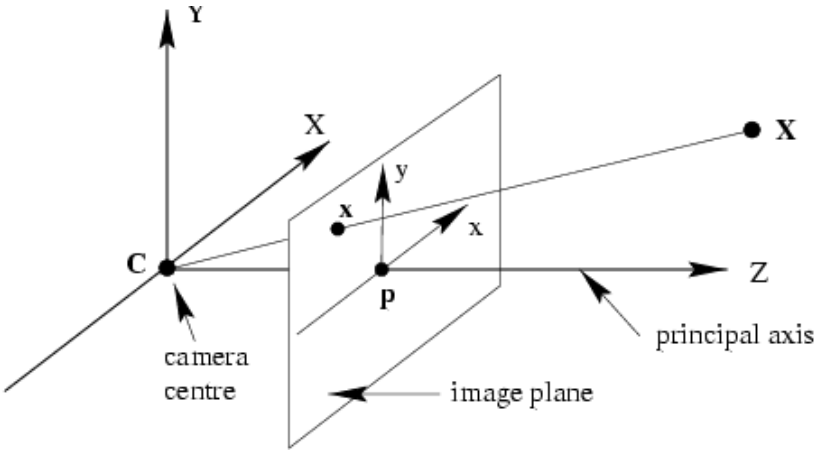
- Modelo de la cámara pinhole:

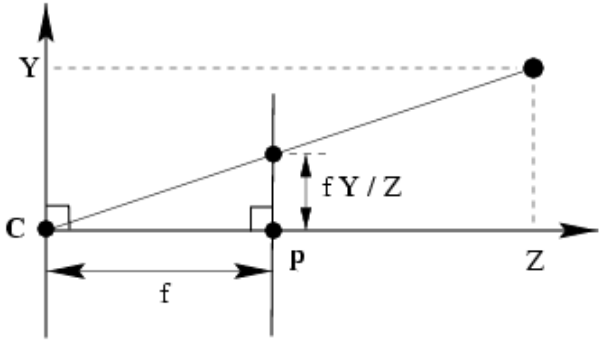




- Los sistemas de coordenadas
  - Nos basaremos en el modelo pinhole
  - Colocamos el centro óptico (***O***) en el origen
  - Colocamos el plano imagen (**Π'**) delante de ***O***

Modelo de cámara





- Ecuaciones de la proyección
  - Calculamos la intersección con Π' de un rayo procedente de ***P*** = (x,y,z) y que pasa por ***O***
  - Por semejanza de triángulos: $(x, y, z) \rightarrow (f\frac{x}{z}, f\frac{y}{z}, f)$
  - Obtenemos la proyección eliminando la última coordenada:

$$(x, y, z) \rightarrow (f\frac{x}{z}, f\frac{y}{z})$$

Coordenadas homogéneas

- Esta transformación es no lineal debido a la división por z.
- Truco: añadir una coordenada más

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Coordenadas homogéneas de la imagen

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Coordenadas homogéneas de la escena

- Conversión desde las coordenadas homogéneas

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Matriz de proyección perspectiva

- La proyección puede escribirse como un producto matricial en coordenadas homogéneas

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} \Rightarrow \left(f\frac{x}{z}, f\frac{y}{z} \right)$$

División por la tercera coordenada

- Es necesario añadir más transformaciones

$$\begin{pmatrix} \text{Punto 2D (3x1)} \end{pmatrix} = \begin{pmatrix} \text{Mat. de trans. de cámara a pixels (3x3)} \end{pmatrix} \begin{pmatrix} \text{Matriz de Proy. perspectiva (3x4)} \end{pmatrix} \begin{pmatrix} \text{Matriz de transf. de la escena a coord. cámara (4x4)} \end{pmatrix} \begin{pmatrix} \text{Punto 3D (4x1)} \end{pmatrix}$$

De la cámara a la imagen

- Existe una traslación y un cambio de escala (metros a pixels)

$$x_{image} = k_x x_{camera} + o_x$$
$$y_{image} = k_y y_{camera} + o_y$$

$$\begin{bmatrix} x_{image} \\ y_{image} \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & o_x \\ 0 & k_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{camera} \\ y_{camera} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} U_{image} \\ V_{image} \\ S \end{bmatrix} = \begin{bmatrix} k_x & 0 & o_x \\ 0 & k_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_{camera} \\ V_{camera} \\ S \end{bmatrix}$$

$U_{camera} \rightarrow -fX$

$V_{camera} \rightarrow -fY$

$S \rightarrow Z$

De la escena a la cámara

- Entre el sistema de referencia de la escena y el sistema de la cámara, existe una rotación y una traslación:

$$\mathbf{X}_{camera} = R\mathbf{X}_{world} + T$$

- En coordenadas homogéneas:

$$\begin{bmatrix} X_{camera} \\ Y_{camera} \\ Z_{camera} \\ 1 \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \\ 1 \end{bmatrix}$$

Resumiendo

coordenadas homogeneas de imagen

parámetros intrínsecos de la cámara

parámetros extrínsecos de la cámara

coordenadas homogéneas de la escena

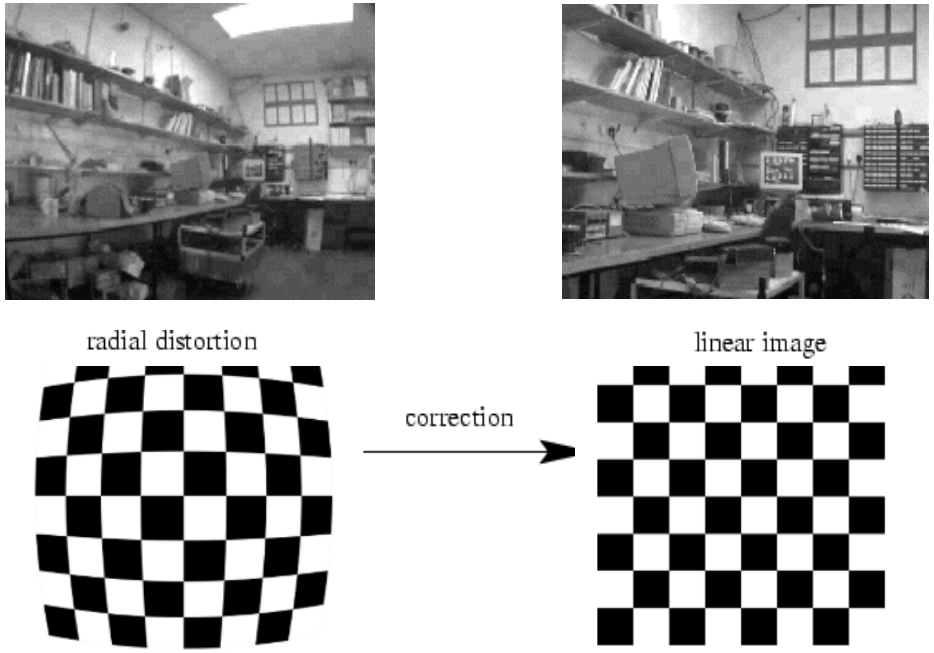
$$\begin{bmatrix} U_{image} \\ V_{image} \\ S \end{bmatrix} = \begin{bmatrix} -fk_x & 0 & o_x & 0 \\ 0 & -fk_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \\ 1 \end{bmatrix}$$

$$x_{image} - o_x = -f_x \frac{r_{1,1}X_{world} + r_{1,2}Y_{world} + r_{1,3}Z_{world} + t_x}{r_{3,1}X_{world} + r_{3,2}Y_{world} + r_{3,3}Z_{world} + t_z} \qquad (A)$$

$$y_{image} - o_y = -f_y \frac{r_{2,1}X_{world} + r_{2,2}Y_{world} + r_{2,3}Z_{world} + T_y}{r_{3,1}X_{world} + r_{3,2}Y_{world} + r_{3,3}Z_{world} + T_z} \qquad (B)$$

Otros fenómenos

- Skew (pixels no rectangulares)
- Distorsión radial



Projectar $(\hat{x}, \hat{y}, \hat{z})$
a coordenadas de
imagen “normalizadas”

$$x'_n = \hat{x}/\hat{z}$$
$$y'_n = \hat{y}/\hat{z}$$

Aplicar la distorsión radial

$$r^2 = {x'_n}^2 + {y'_n}^2$$
$$x'_d = x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$
$$y'_d = y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4)$$

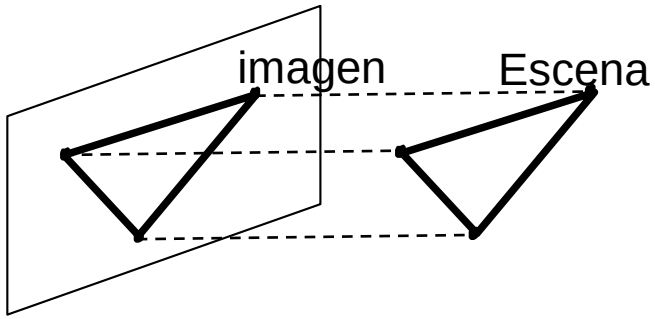
Aplicar la proyección y el paso
a coordendas de imagen

$$x' = f x'_d + x_c$$
$$y' = f y'_d + y_c$$

Otros tipos de proyecciones

- Poryección ortográfica o paralela
  - El centro de proyección está en el infinito

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

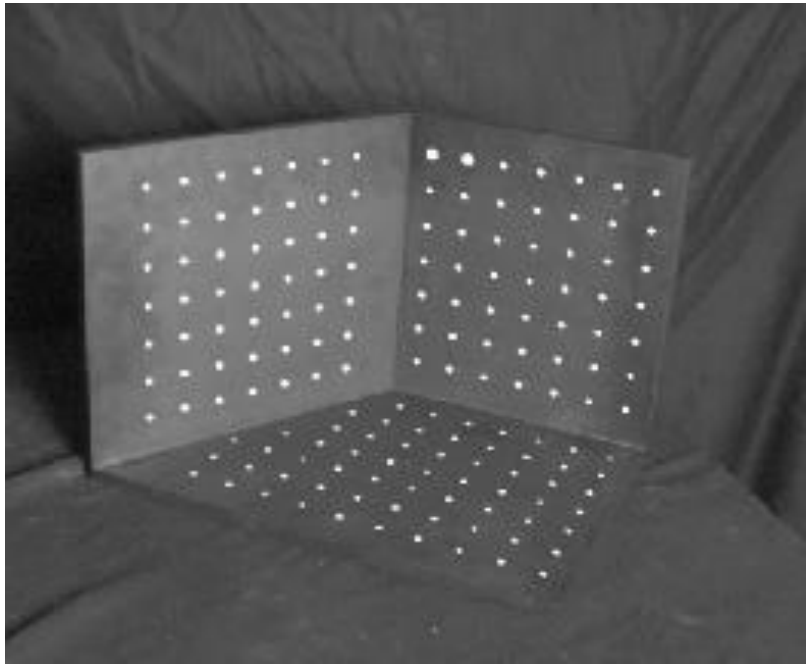


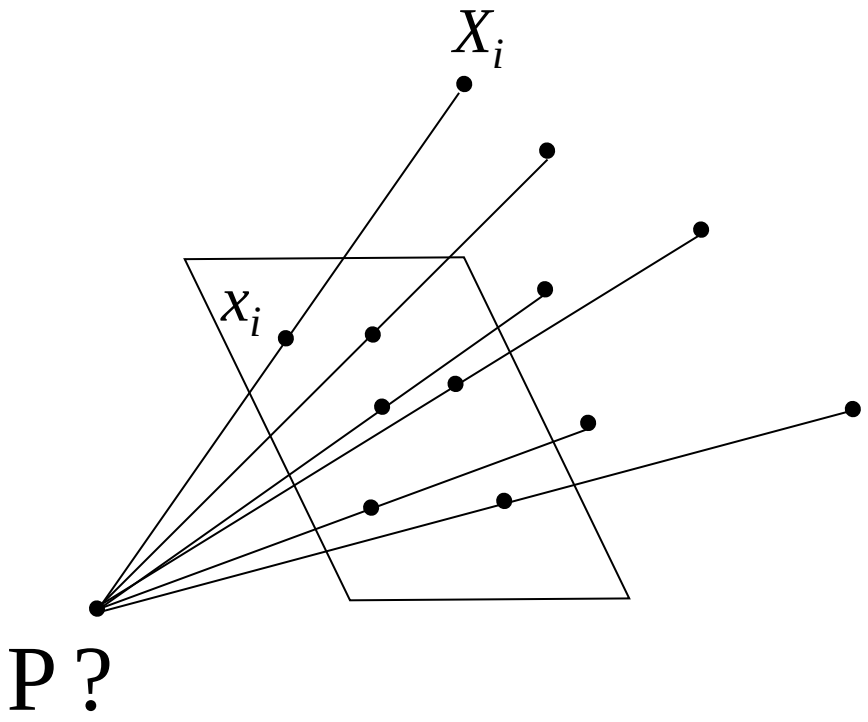
- Proyección ortográfica escalada
  - También denominada perspectiva débil

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1/d \end{bmatrix} \Rightarrow (dx, dy)$$

Calibración de la cámara

- Estimar los parámetros de la cámara dados n puntos con coordenadas 3D conocidas ($\mathbf{X}_i$) y sus proyecciones en la imagen también conocidas (x_i)





Calibración de la cámara

- El módelo de la cámara es una ecuación de la forma

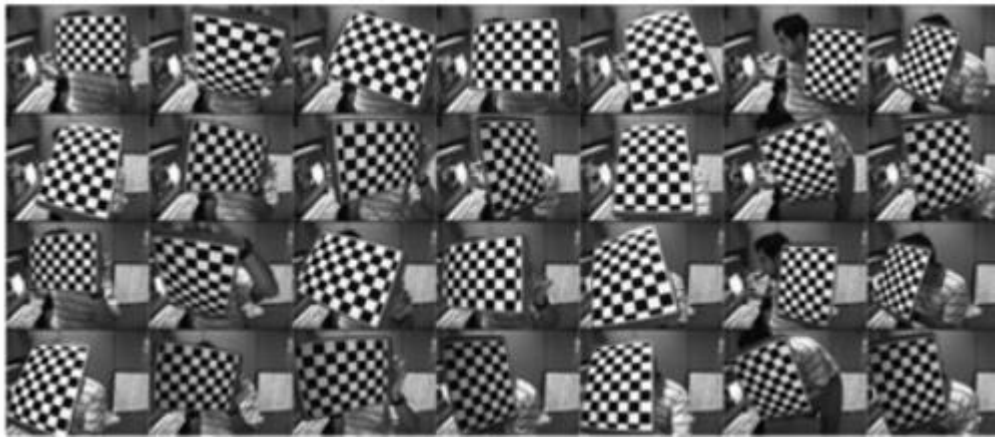
$$\begin{bmatrix} x_i \\ y_i \\ s \end{bmatrix} = A \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad donde \quad A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix}$$

- Lo que nos proporciona ecuaciones de la forma:

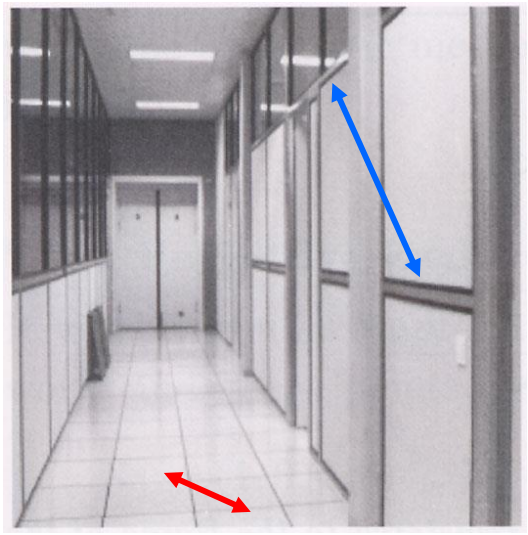
$$a_{11}Xi + a_{12}Yi + a_{13}Zi + a_{14} - x_i(a_{31}Xi + a_{32}Yi + a_{33}Zi + a_{34}) = 0$$
$$a_{21}Xi + a_{22}Yi + a_{23}Zi + a_{24} - y_i(a_{31}Xi + a_{32}Yi + a_{33}Zi + a_{34}) = 0$$

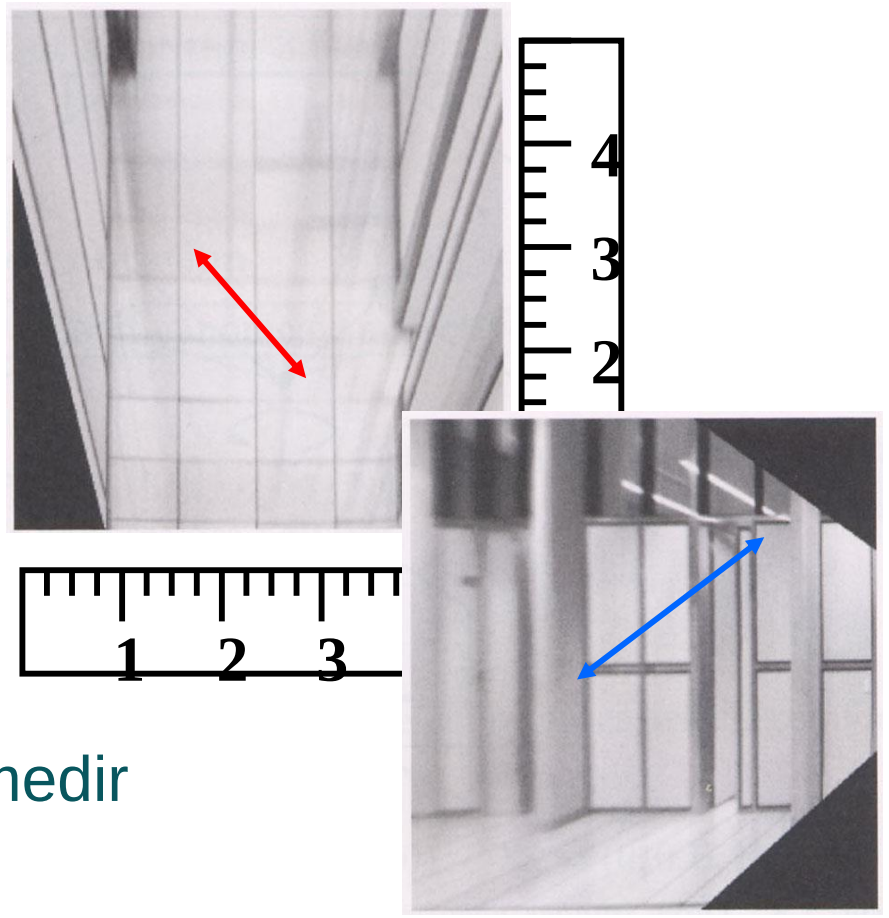
Calibración de la cámara

- Calibrar la cámara implica encontrar los coeficientes de la matriz A.
  - Se requieren al menos 6 puntos, conocidos tanto en coordenadas globales como en coordenadas de imagen (12 ecuaciones). (Más, si se desea calibrar la distorsión)
  - Se trata de un sistema homogéneo, hay infinitas soluciones.
  - Típicamente se toman más de 6 puntos, sistema sobredeterminado que se resuelve por mínimos cuadrados.
  - La forma más habitual es utilizar un chessboard: los puntos son fáciles de localizar de forma semi-automática, y además la restricción del plano hace que no sea necesario introducir su posición espacial



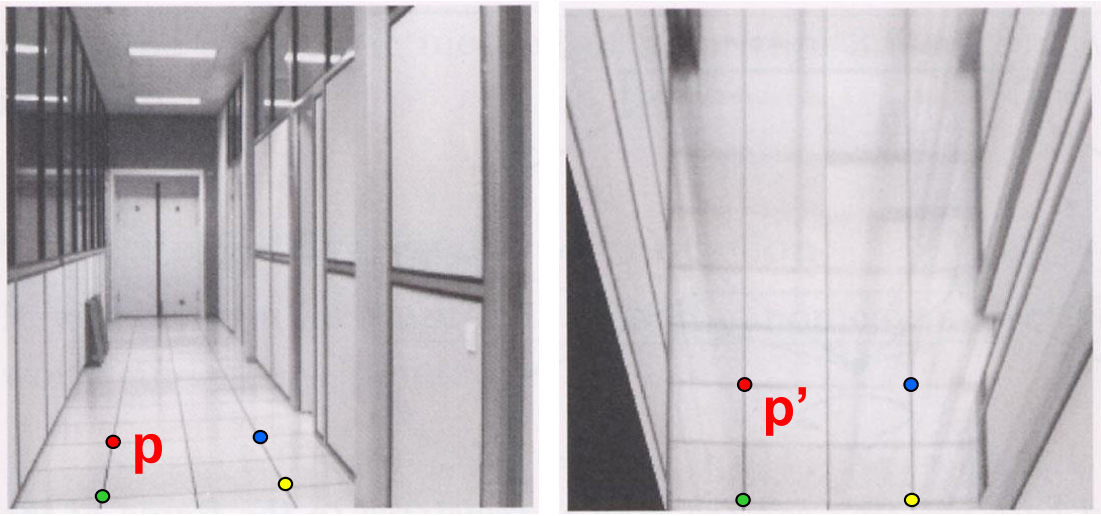
Medidas en planos





Solución: Rectificar y medir

Rectificación de la imagen



- Para rectificar (unwrap) una imagen
  - Resolver la transformación homográfica H dados p y p'
  - Implica resolver ecuaciones de la forma: $wp' = Hp$
    - Lineal en las incógnitas: w y coeficientes de H
    - H está definida salvo el factor de escala
    - Cuántos puntos se necesitan para obtener H?

Rectificación de la imagen

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \longrightarrow \begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Rectificación de la imagen

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

- Se resuelve mediante mínimos cuadrados:

$$minimizar\ (\|Ah - 0\|^2)$$

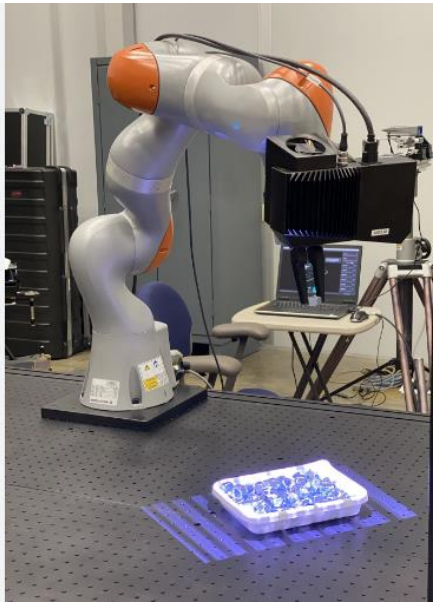
  - Como h está definida hasta un valor de escala, se puede resolver para que sea un vector unitario.
  - Solución: $\hat{h}$ = autovector de A^TA correspondiente al menor valor propio
  - Funciona con 4 o más puntos.

Indice

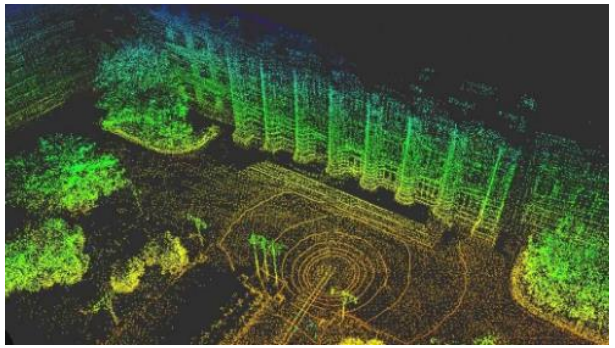
- Estructura del curso
- Calibración de cámaras
- **Obtención de información 3D**
- Programación C/C++ con OpenCV
- Las nuevas técnicas: Inteligencia Artificial
- Conclusiones

Aplicaciones de la información 3D

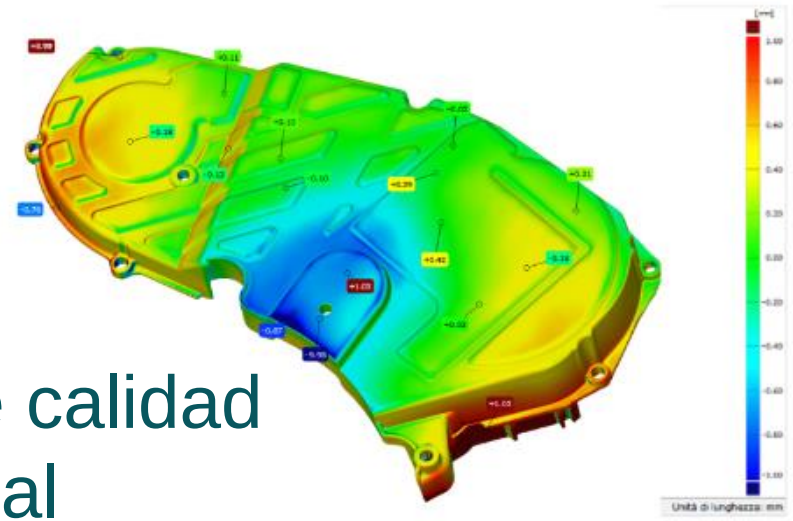
- Bin picking



- Navegación autónoma

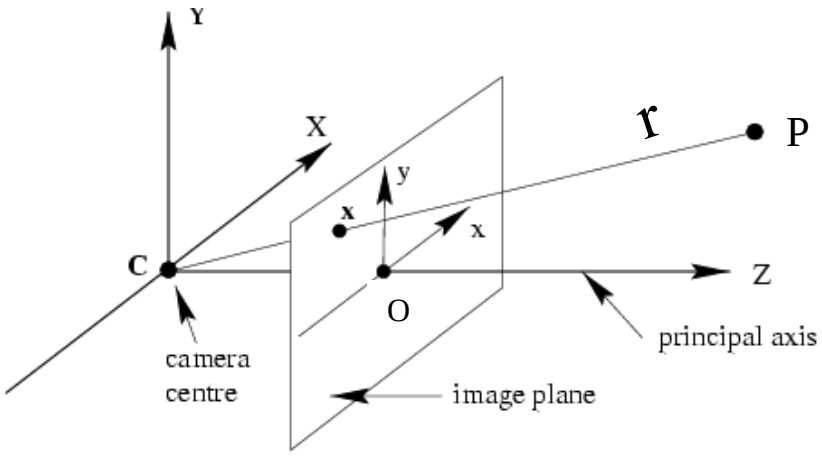


- Control de calidad dimensional



Obtener información 3D

- La cámara pierde información respecto al mundo real (3D → 2D)

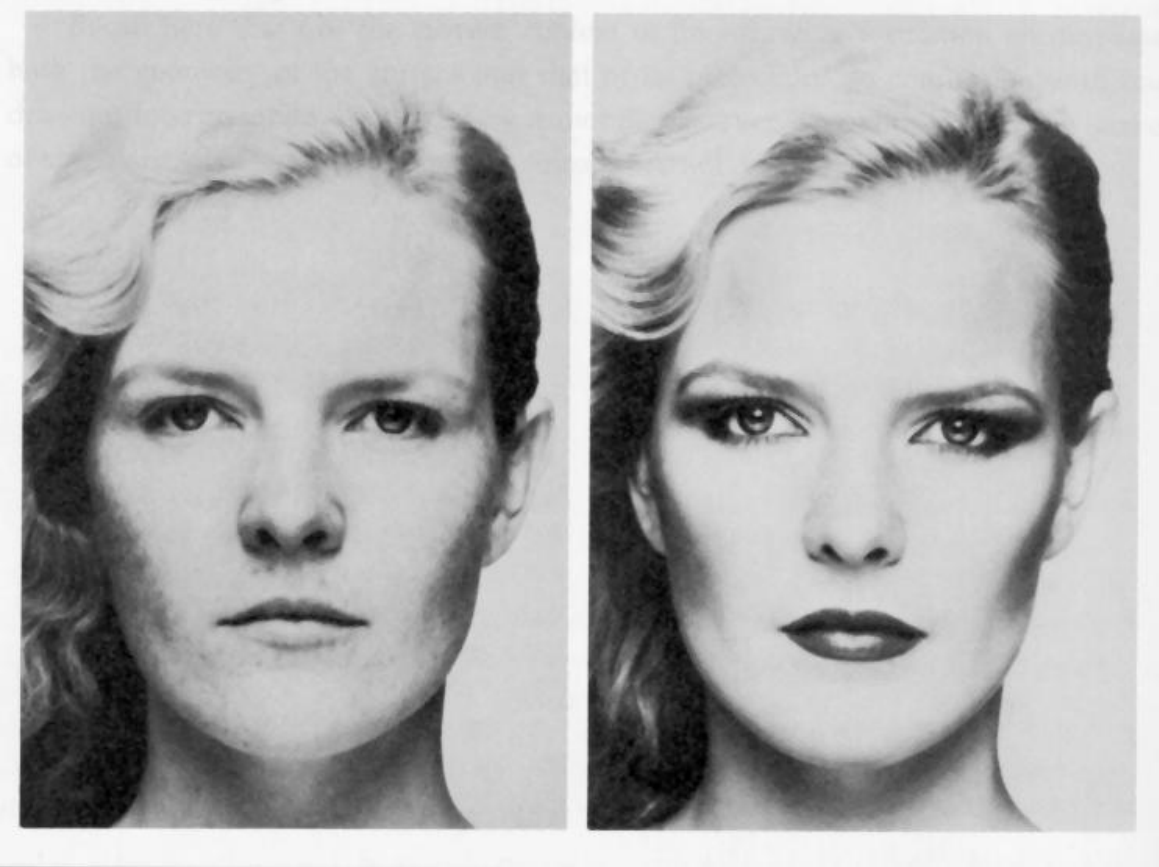


El punto del mundo real P que ha generado el punto en la imagen (x) puede haber sido cualquiera de la recta r

Es necesaria información adicional para resolver cuál de los puntos de la recta r es el punto origen P

Información 3D en una imagen

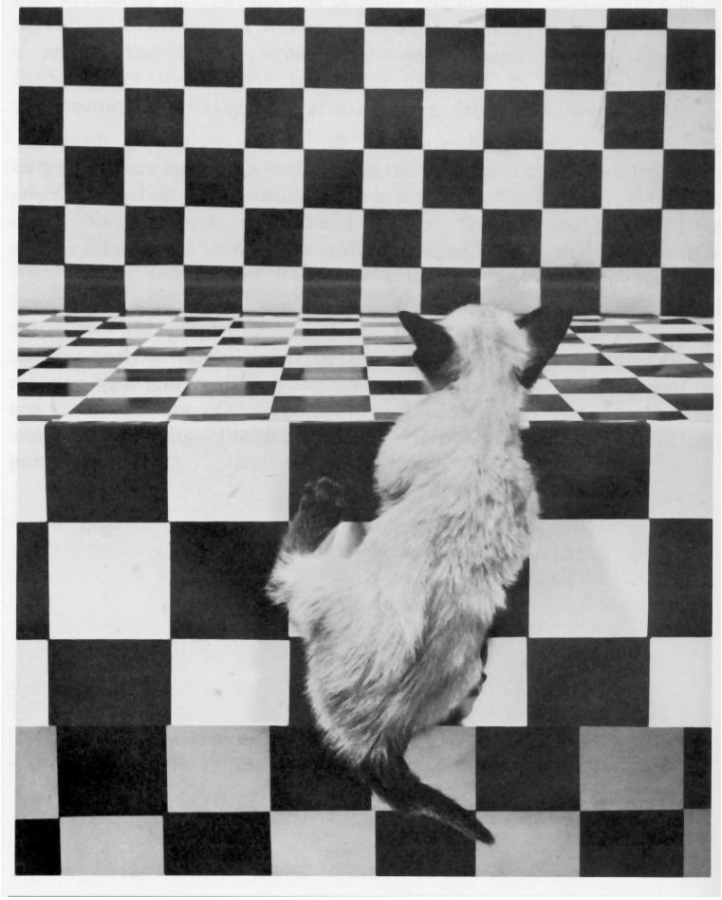
- Shading (sombreado)



Merle Norman Cosmetics, Los Angeles

Información 3D en una imagen

- Shading (sombreado)
- Textura



***The Visual Cliff*, by William Vandivert, 1960**

Información 3D en una imagen

- Shading (sombreado)
- Textura
- Enfoque

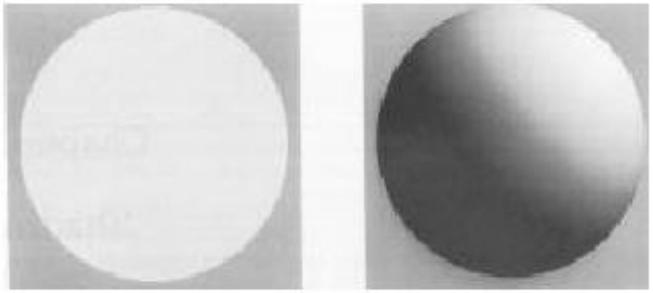
From *The Art of Photography*, Canon

Información 3D en una imagen

- Shading (sombreado)
- Textura
- Enfoque
- Movimiento

Técnicas de control dimensional on-line

Técnicas basadas en información de luz y sombra



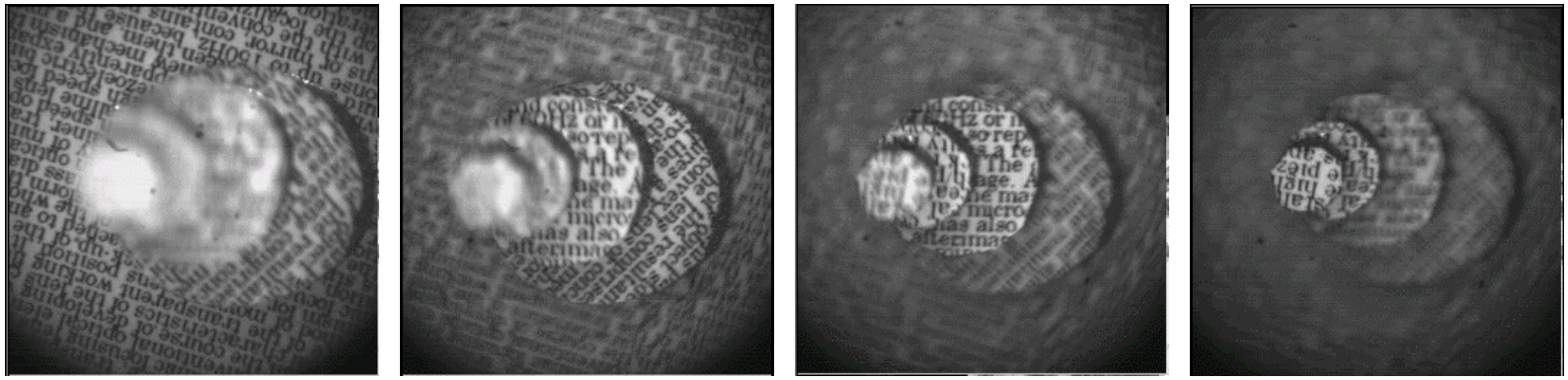
Shape from shading: uso directo de la información de sombras

Photometric stereo: iluminación de una escena desde varios puntos diferentes

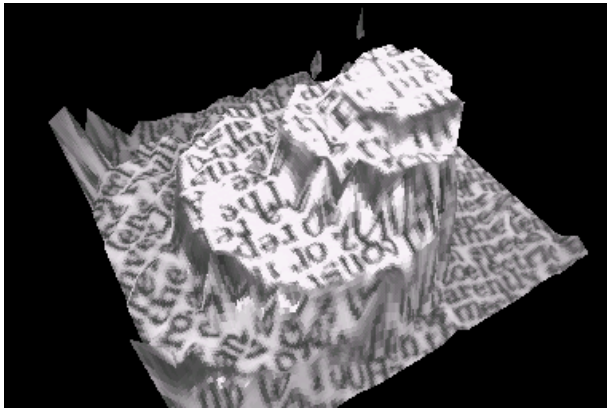
Técnicas de control dimensional on-line

Enfoque dinámico

- Buscar la distancia óptima de enfoque de cada punto de la escena

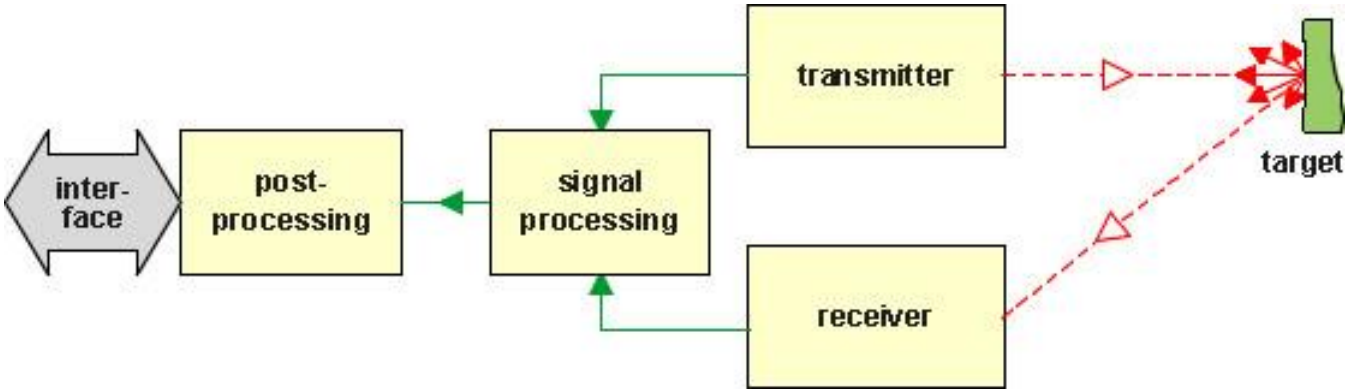


- Reconstrucción 3D



Técnicas de control dimensional on-line

- **Técnicas de tiempo de vuelo:**
  - Medida del tiempo en que una determinada forma de energía tarda en regresar rebotado de un objeto.

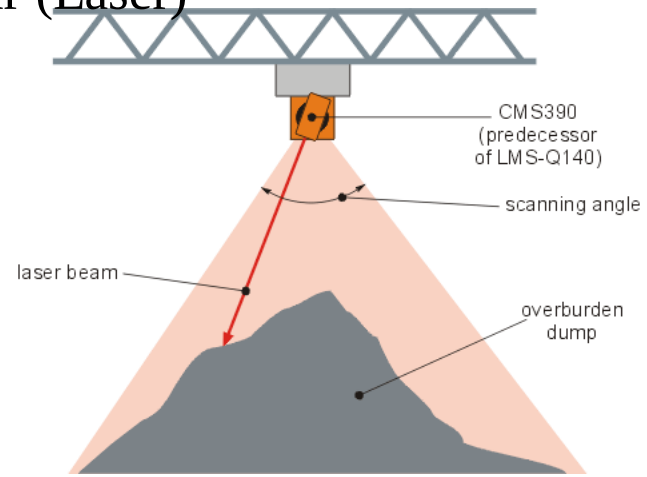


  - Energía: luz láser, microondas, ultrasonidos, etc.
  - Formato: pulso, onda.

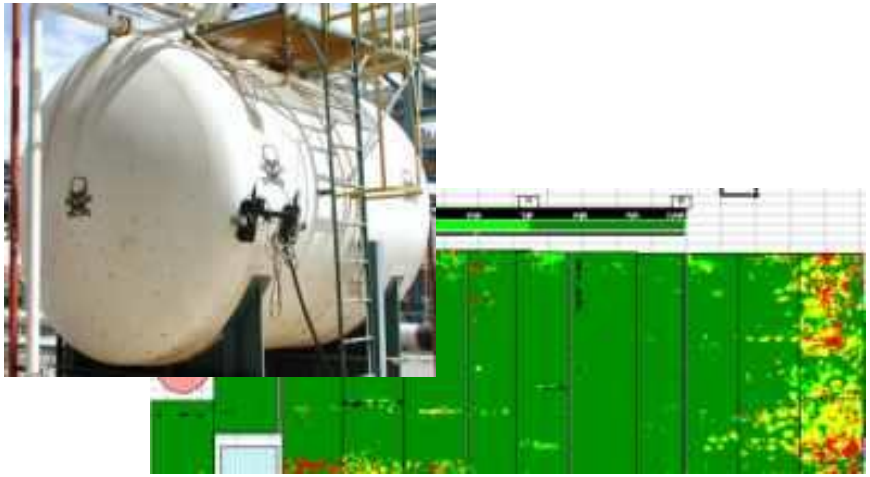
Técnicas de control dimensional on-line

Ejemplos de técnicas de tiempo de vuelo

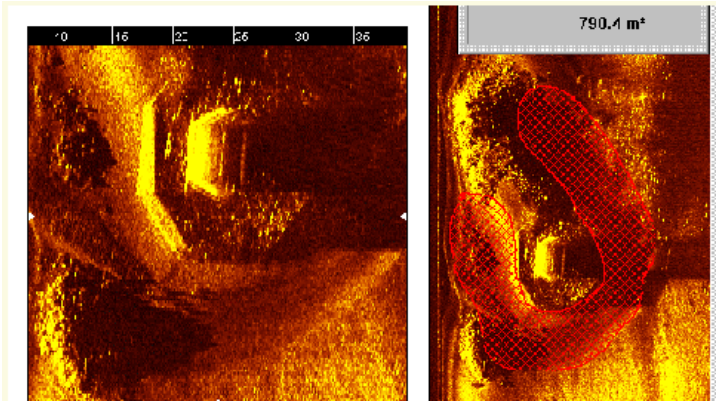
- Lidar (Láser)



- Ultrasonido

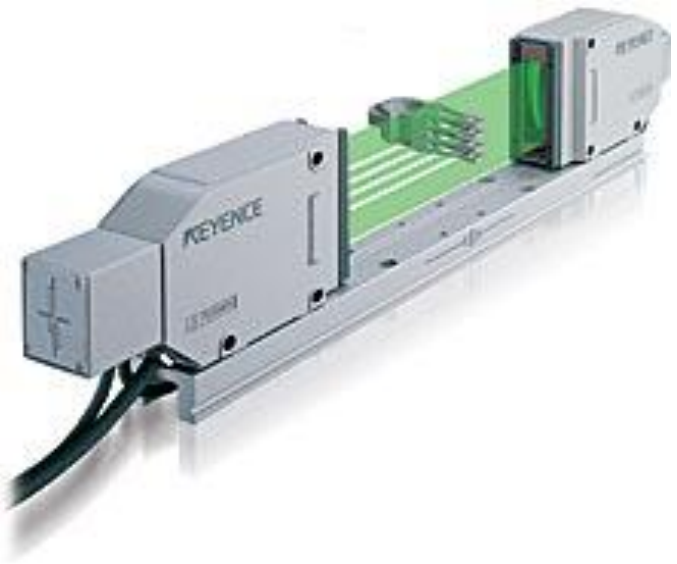


- Sonar

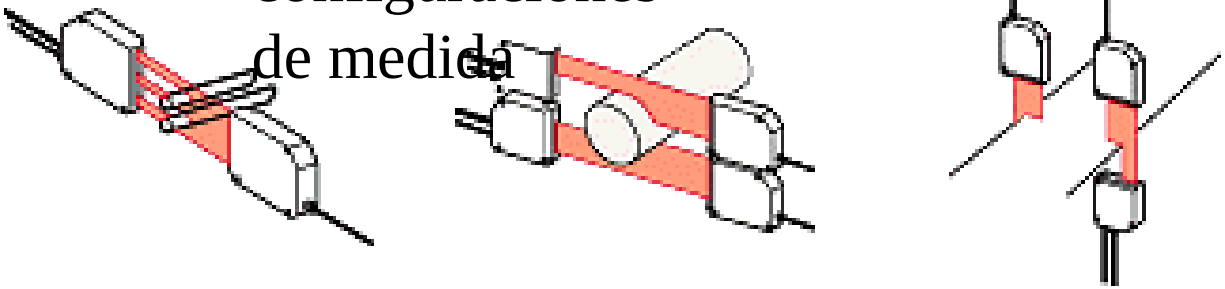


Técnicas de control dimensional on-line

Ténicas de barrera

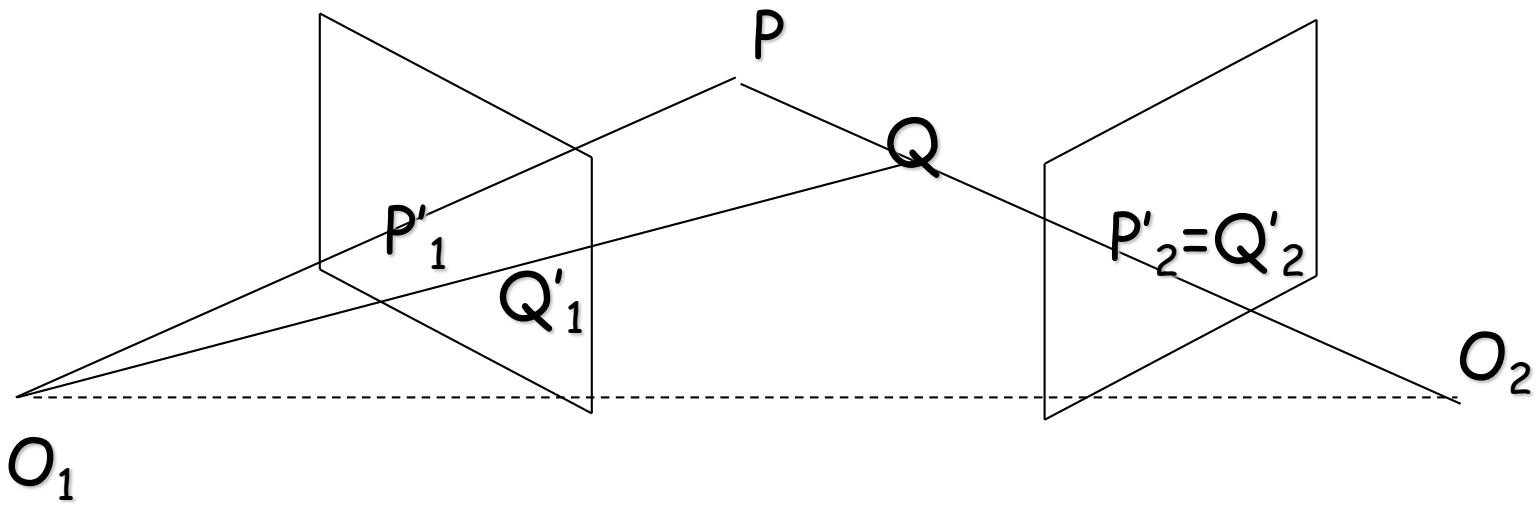


Sensores de barrera en varias configuraciones de medida



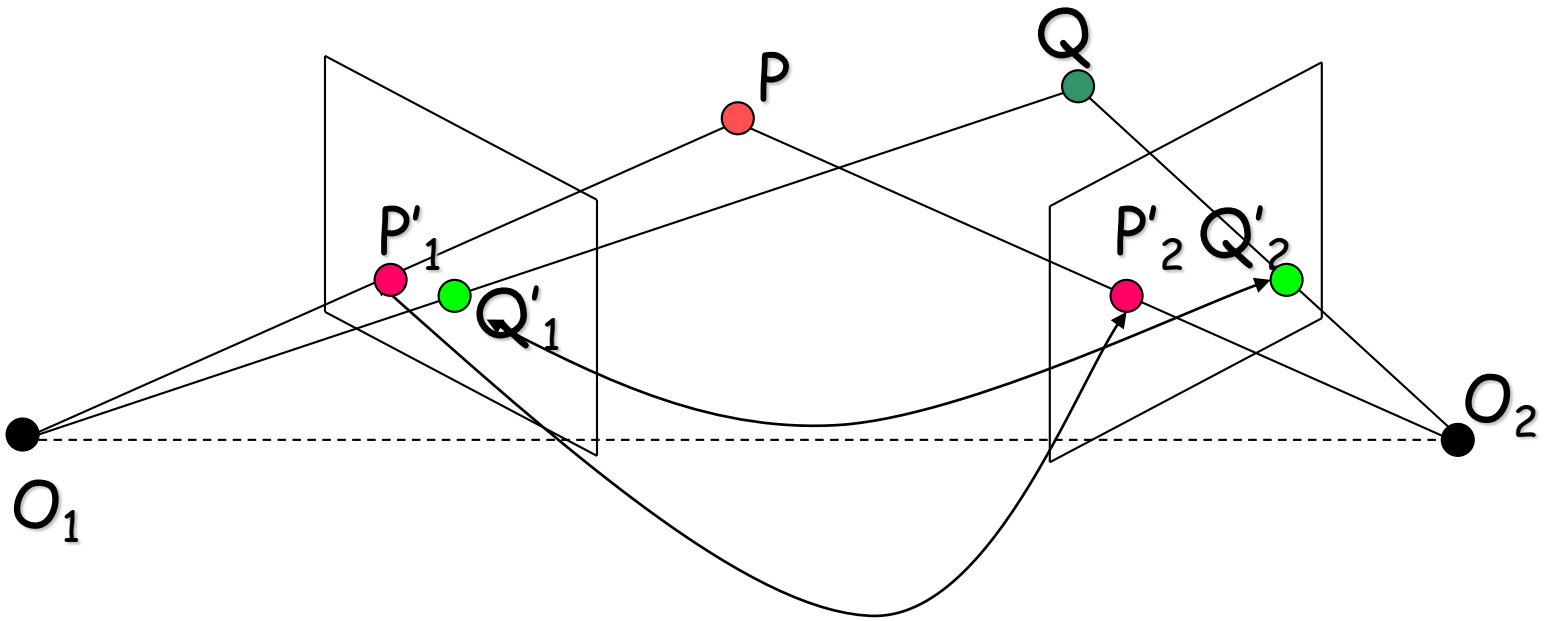
Estéreo

La profundidad pueden calcularse con dos imágenes y triangulación



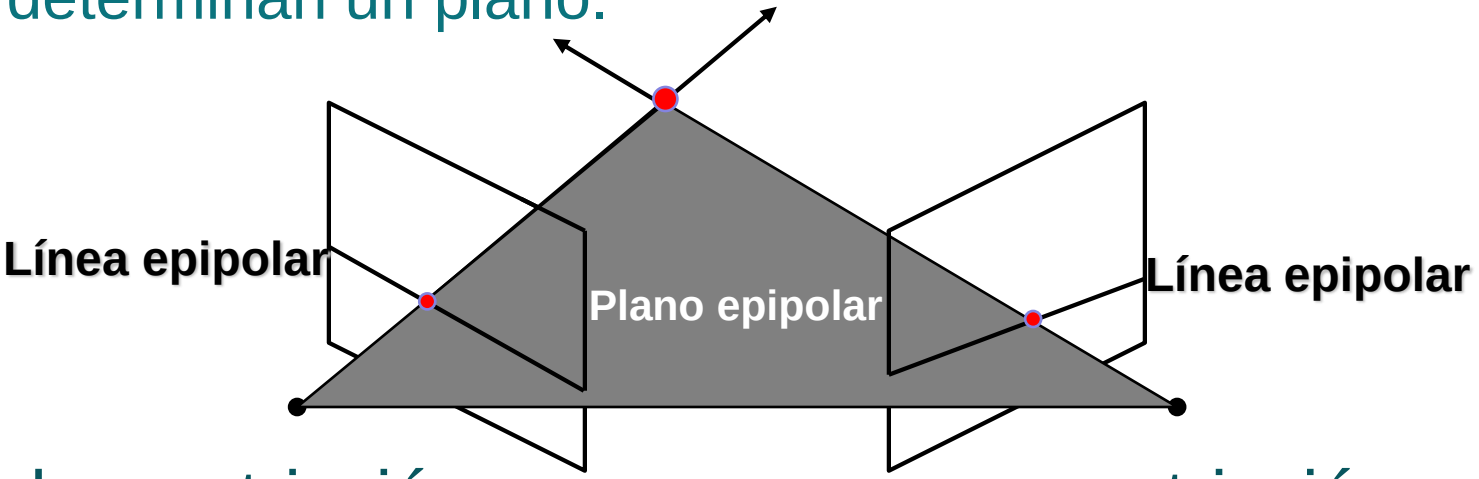
Estéreo

- Para ello es necesario establecer correspondencias:
  - Identificar las proyecciones de un punto de la escena en cada imagen



Estéreo

- Determinación de correspondencias
  - Las líneas proyectivas de un punto de la escena determinan un plano.

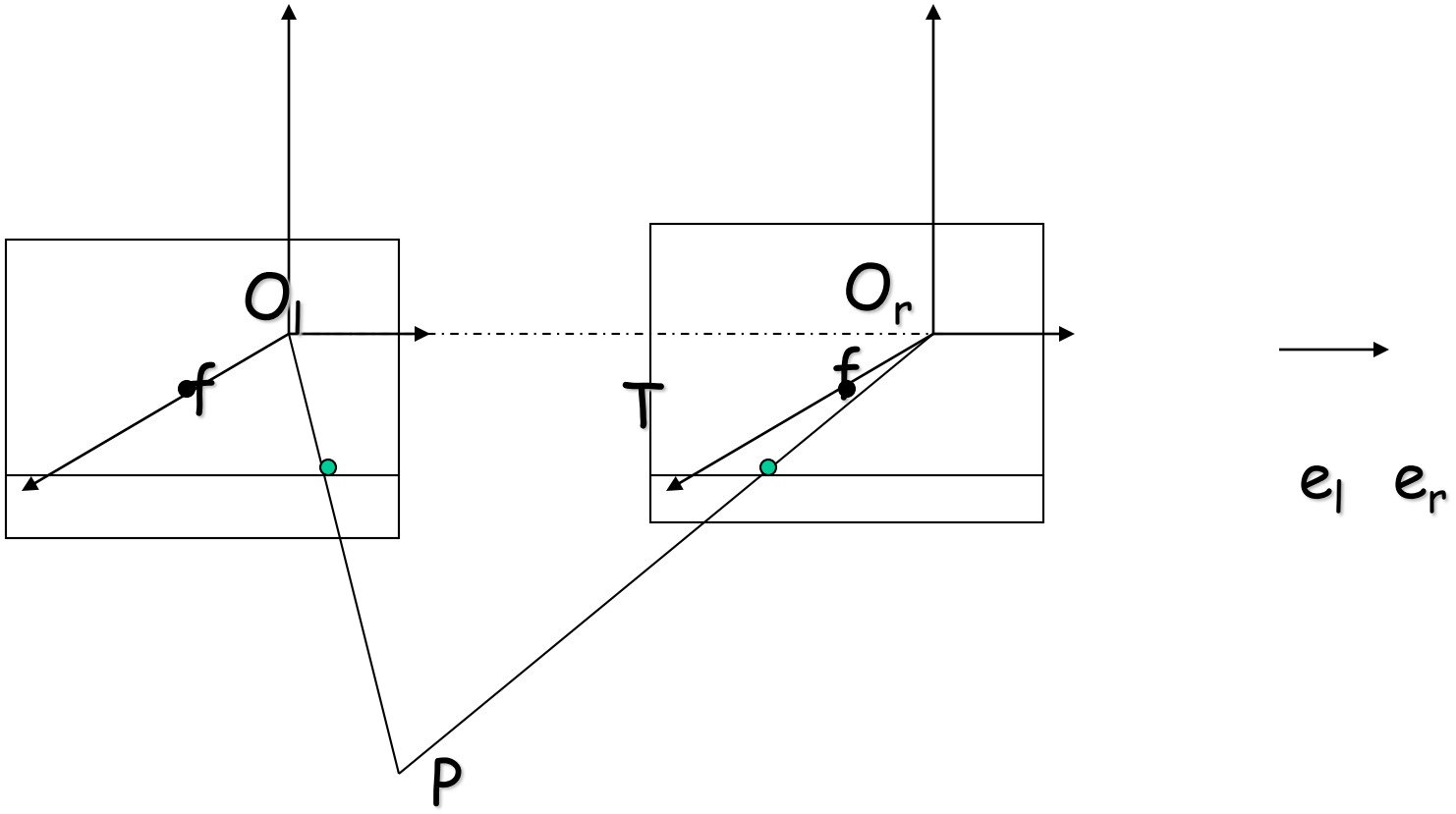


- Dicha restricción se conoce como restricción epipolar
  - Reduce el problema de la búsqueda de correspondencias a un problema de búsqueda 1D a lo largo de líneas epipolares conjugadas.

Estéreo

- Los planos imagen de ambas cámaras deben ser paralelos.
- Los puntos focales deben estar a la misma altura.
- La distancia focal debe ser la misma.
- En ese caso, las líneas epipolares coinciden con las líneas horizontales de scan.

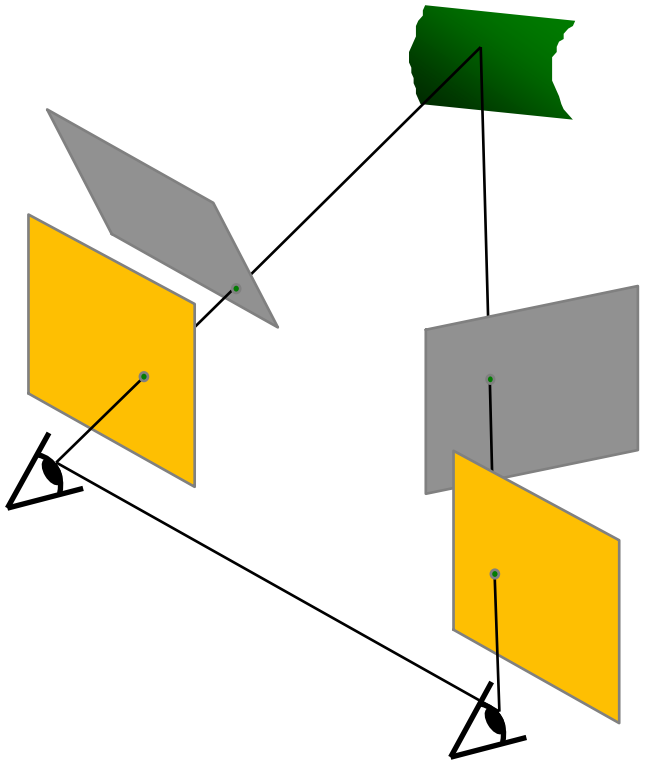
Estéreo



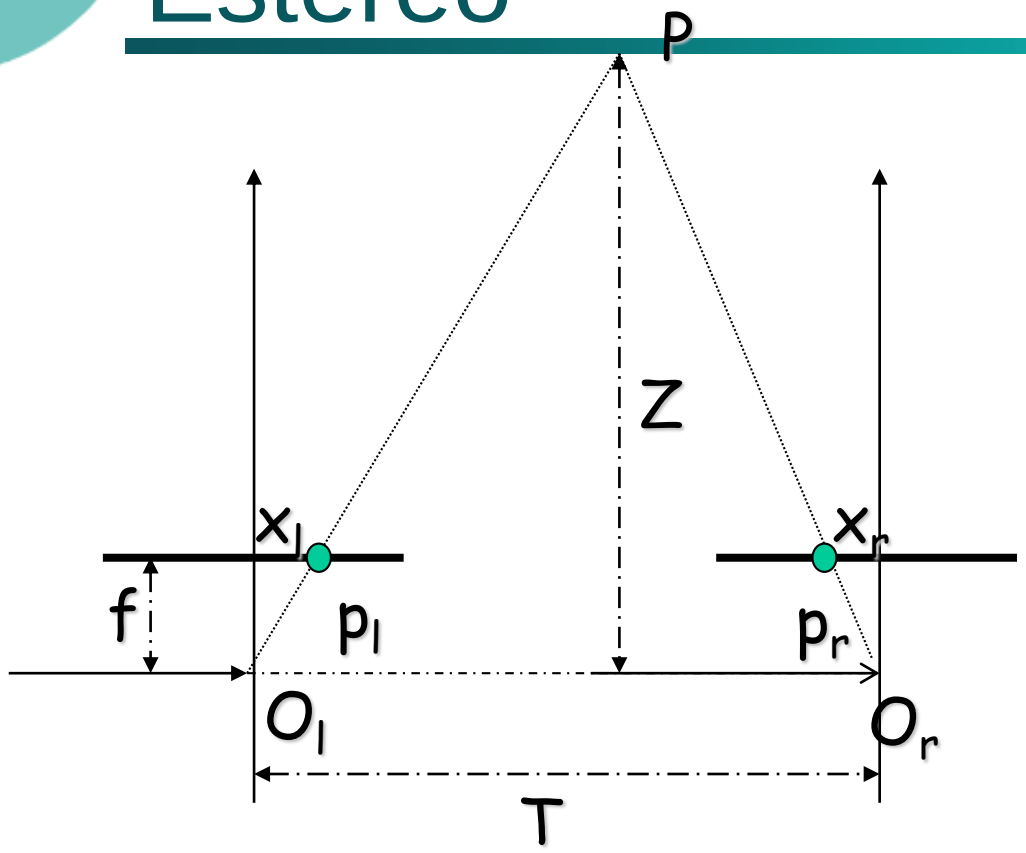
Los epipolos están en el infinito.
Las líneas epipolares son paralelas a la linea base

Estéreo

- Mediante la reproyección de la imagen es posible pasar del caso general al caso de ejes ópticos paralelos.
- La reproyección se logra mediante una transformación homográfica.



Estéreo



$$\frac{T + x_r - x_l}{Z - f} = \frac{T}{Z}$$

$$Z = f\frac{T}{x_l - x_r}$$

disparidad: $d = x_l - x_r$

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

Conocido Z, podemos calcular X e Y.
T es la línea base (distancia entre los centros ópticos)
d mide la diferencia de posición entre puntos correspondientes

Estéreo

- Emparejamiento de puntos
  - Se supone que el brillo del pixel es invariante entre ambas imágenes
  - Presenta el problema de las oclusiones
  - Es un problema complejo
  - Existen numerosos enfoques
    - Programación dinámica
    - Funciones de suavizado
    - Correlación
    - Uso de más de dos imágenes (trinocular, N-ocular)
    - Algoritmos basados en grafos

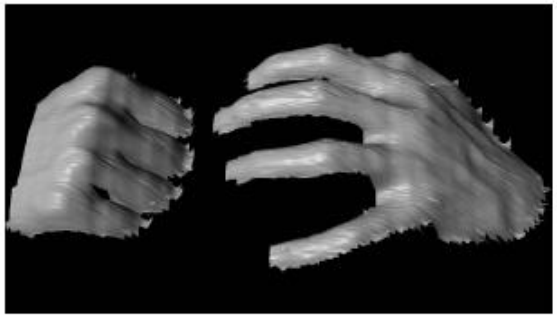
Estéreo

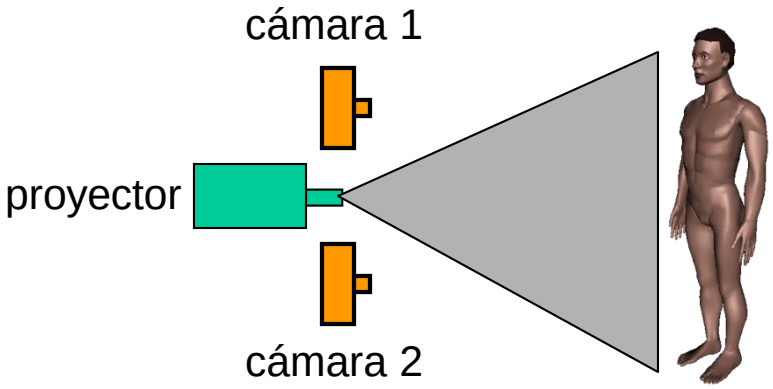
- Pasos
  - Calibrar las cámaras
  - Rectificar las imágenes
  - Calcular la disparidad
  - Estimar la profundidad
- Causas de error
  - Errores en la calibración de las cámaras
  - Mala resolución de la imagen
  - Oclusiones
  - Violaciones de la constancia del brillo
  - Grandes desplazamientos
  - Regiones poco contrastadas

Estéreo

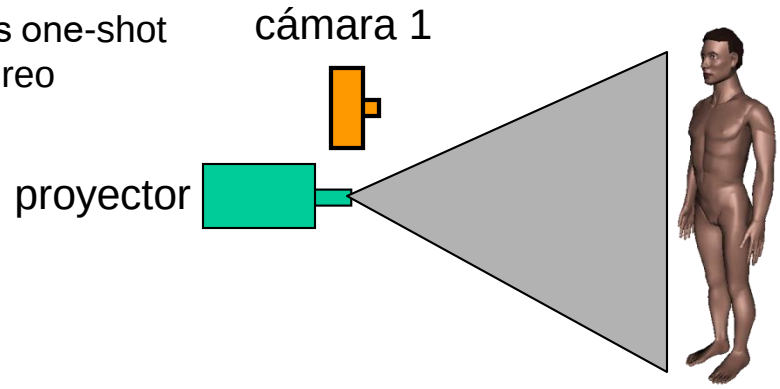
Luz estructurada





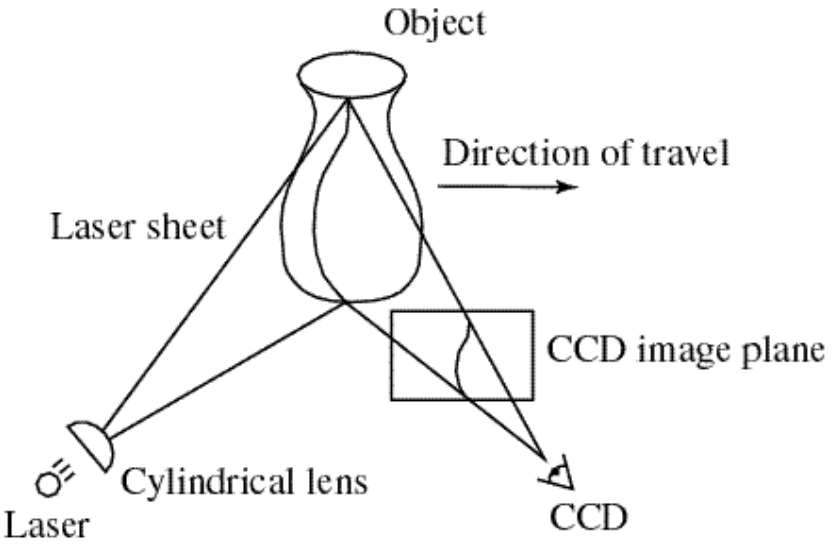


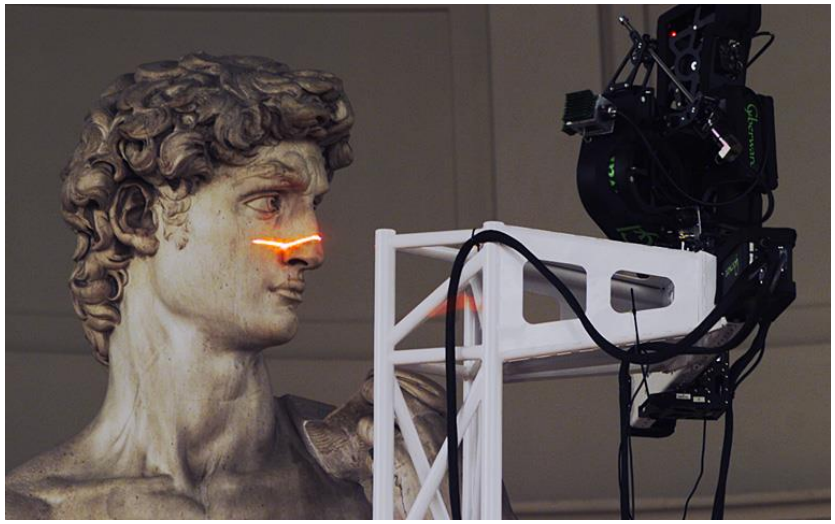
Li Zhang's one-shot stereo



- Se proyectan patrones de luz "estructurada" sobre el objeto.
  - Simplifica el problema de la correspondencia

Escaneado láser



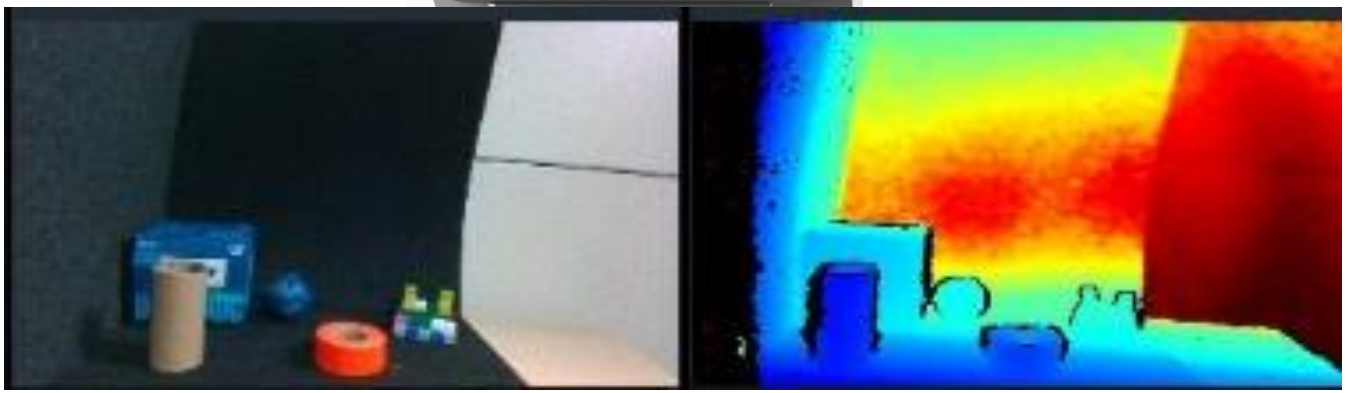


Digital Michelangelo Project

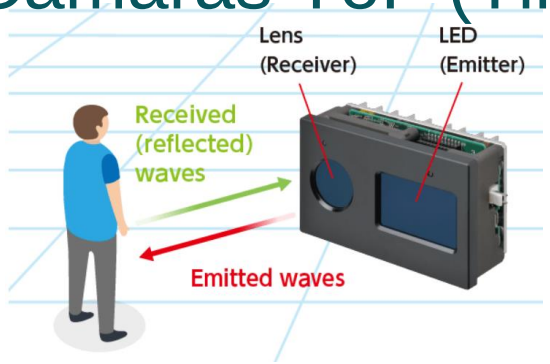
- Se recurre a la triangulación óptica
  - Se proyecta una línea láser sobre el objeto.
  - Se hace un barrido sobre la superficie el objeto
  - Se pueden lograr reconstrucciones muy precisas

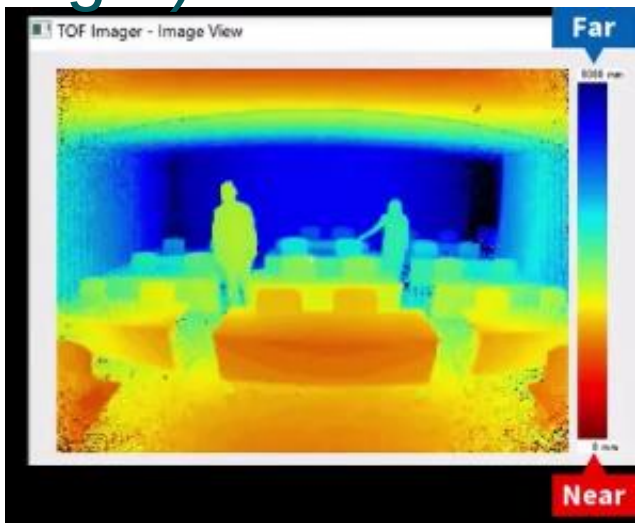
Dispositivos de imagen 3D

- Cámaras estéreo



- Cámaras ToF (Time Of Flight)





Dispositivos de imagen 3D

- Laser scanner

- Photogrammetry

Indice

- Estructura del curso
- Calibración de cámaras
- Obtención de información 3D
- **Programación C/C++ con OpenCV**
- Las nuevas técnicas: Inteligencia Artificial
- Conclusiones

Programación con OpenCV

- Con Matlab (u otros similares) desarrollamos la investigación
- En general, queremos implementar la solución con C/C++, Python, etc. para no necesitar Matlab (o similares) en el equipo destino
- Son necesarias librerías para las funcionalidades más habituales → OpenCV

  https://opencv.org/releases/

  (Descargar y descomprimir en directorio local ***my_path***)

  (Tutorial de uso:

https://docs.opencv.org/4.8.0/df/d65/tutorial_table_of_content_introduction.html)

Programación con OpenCV

- Instalación para Qt Creator-mingw64 (Windows): hay que compilar, porque los binarios de la versión descargable son para compilar con Visual Studio
  - En Qt Creator, abrir archivo CMakeLists.txt situado en C:***my_path***\opencv\sources
  - Seleccionar kit para compilación
  - Ir a Projects → Build → Seleccionar opciones:
    - Build directory: **C:*my_path*\opencv\build-4.8.0**
    - Build type: **Release**
    - QML debugging and profiling: **Disable**
    - Current configuration: **OPENCV_ENABLE_ALLOCATOR_STATS=OFF**
      BUILD_JAVA=OFF
      BUILD_TESTS=OFF
      BUILD_PERF_TESTS=OFF
  - Ejecutar Run cMake
  - Ejecutar build

Programación con OpenCV

- Desde Qt Creator
  - Modificar .pro:

```
OPENCV_INSTALL_PATH = C:/my_path/opencv
INCLUDEPATH += $${OPENCV_INSTALL_PATH}/build/include
LIBS += -L$${OPENCV_INSTALL_PATH}/build-4.8.0/bin
LIBS += -lopencv_core480  -lopencv_imgcodecs480
LIBS += -lopencv_imgproc480 -lopencv_video480 -lopencv_videoio480
... etc. según funciones utilizadas ...
```

  - Incluir en .cpp ó .h:

```
#include <opencv2/opencv.hpp>
```

  - Usar objetos de tipo cv::Mat :

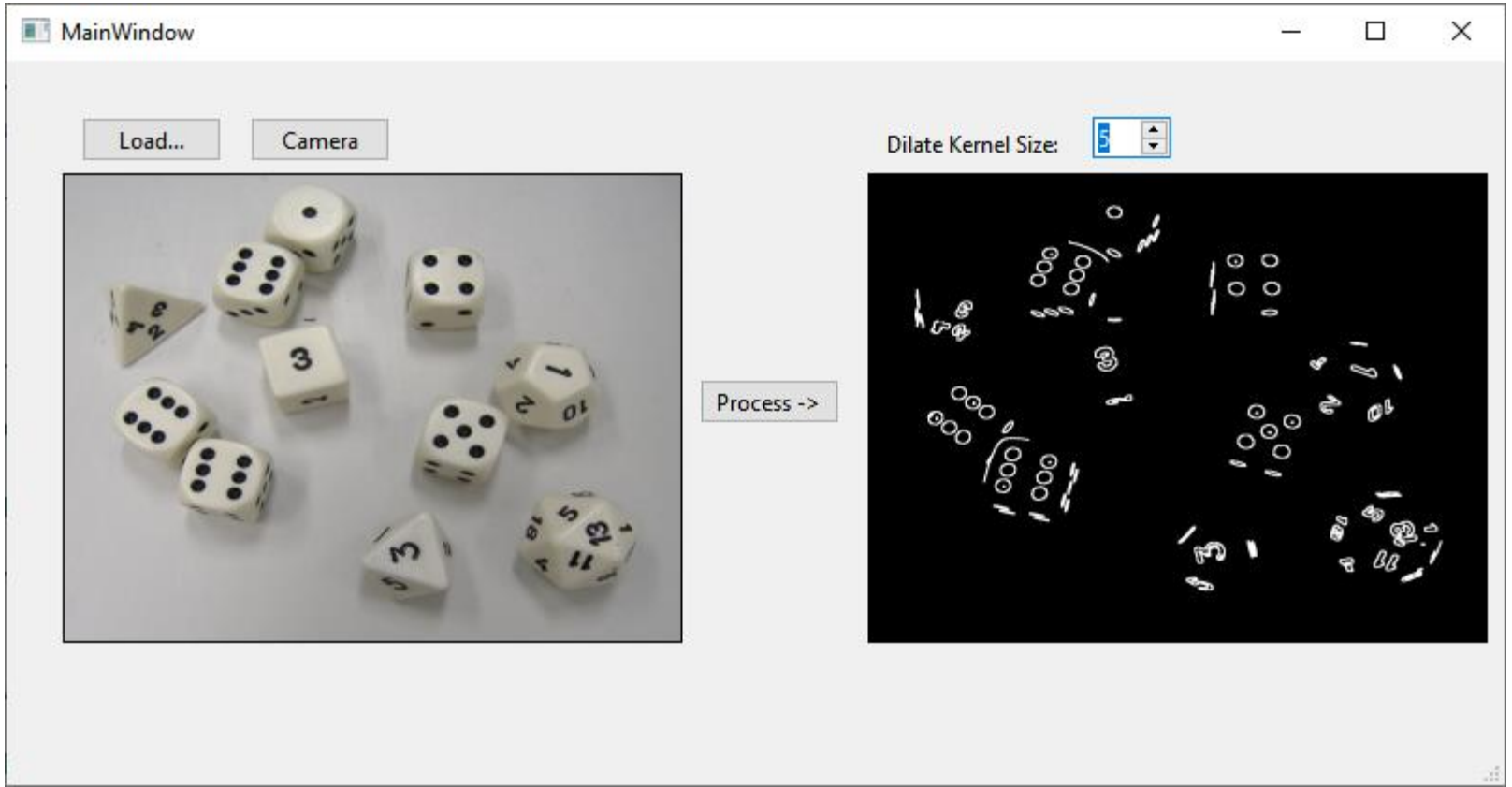
```
cv::Mat imgOrig , imgDest ;
```

  - Usar funcionalidades para operaciones:

```
imgOrig=cv::imread("archivo de imagen");
cv::cvtColor(imgOrig,imgDest,COLOR_BGR2GRAY);
…
```

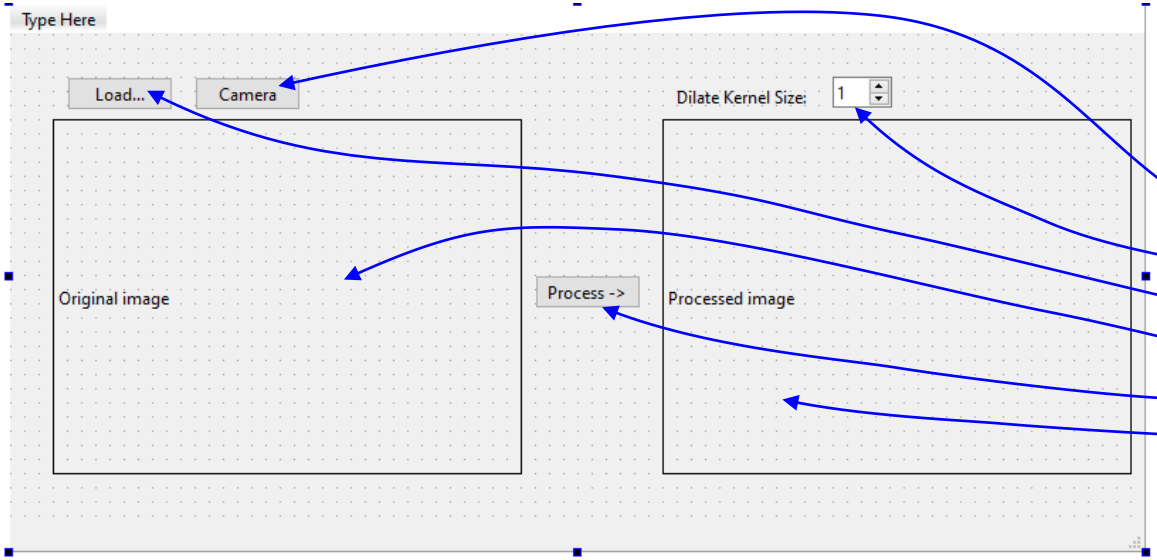
Programación con OpenCV

- Ejemplo:
  - Obtener imagen desde archivo o desde cámara, y ejecutar una búsqueda de bordes de Canny



Programación con OpenCV

- Desde Qt Creator
  - Crear proyecto tipo Widgets / qmake
  - Modificar .pro como se ha indicado anteriormente
  - Añadir elementos en interfaz (.ui):



Object	Class
MainWindow	QMainWindow
centralwidget	QWidget
label	QLabel
qCap_pushButton	QPushButton
qDilateSize_spinBox	QSpinBox
qLoad_pushButton	QPushButton
qOrigImg_label	QLabel
qProcess_pushButton	QPushButton
qResultImg_label	QLabel
menubar	QMenuBar
statusbar	QStatusBar

Programación con OpenCV

- Desde Qt Creator
  - Añadir slots para los botones y el spinbox
  - Editar MainWindow.h:

```
#include <opencv2/opencv.hpp>
#include <QLabel>
...
class MainWindow : public QMainWindow
{
    Q_OBJECT
private:
    cv::Mat imgOrigRGB,imgResultGray;
    cv::VideoCapture cap;
    void ShowImage(const cv::Mat& img,QLabel* dest);
public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_qLoad_pushButton_clicked();
    void on_qProcess_pushButton_clicked();
    void on_qCap_pushButton_clicked(bool checked);
    void on_qDilateSize_spinBox_valueChanged(int arg1);

private:
    Ui::MainWindow *ui;
};
```

Añadir includes necesarios

Añadir variables para imágenes, captura de vídeo
Añadir función para ver imagen en widget tipo label

Slots añadidos desde el ui

Programación con OpenCV

- Desde Qt Creator
  - Rellenar código en MainWindow.cpp

```
#include "MainWindow.h"
#include "ui_MainWindow.h"
#include <QFileDialog>

MainWindow::MainWindow(QWidget *parent) :  QMainWindow(parent), cap(0), ui(new Ui::MainWindow)
...

void MainWindow::ShowImage(const cv::Mat &img, QLabel *dest)
{
    QImage::Format format;
    if (img.depth()==CV_8U ) {
        if (img.channels()==1)
            format=QImage::Format_Grayscale8;
        else if (img.channels()==3)
            format=QImage::Format_BGR888;
    }
    else {
        return;
    }

    QImage qImg(img.data,img.cols,img.rows,img.step,format);

    dest->setPixmap(QPixmap::fromImage(qImg));
    dest->setScaledContents(true);
}
```

Dispositivo de captura de imagen por defecto

Convierte img (cv::Mat) en QImage y dibuja la misma en el QLabel

Descubre formato de imagen img (cv::Mat) y establece formato equivalente para crear imagen qImg (Qimage)

Crea QImage a partir de cv::Mat

Establece pixmap para el QLabel

Escala al tamaño del widget

Programación con OpenCV

```
void MainWindow::on_qLoad_pushButton_clicked() {
    QString file=QFileDialog::getOpenFileName(this,"Seleccione imagen","","");
    if (! file.isEmpty())     {
        imgOrigRGB=cv::imread(file.toLatin1().constData());
        ShowImage(imgOrigRGB,ui->qOrigImg_label);
    }
}

void MainWindow::on_qCap_pushButton_clicked(bool checked) {
    if (checked && cap.grab() )
    {
        cap.read(imgOrigRGB);
        ShowImage(imgOrigRGB,ui->qOrigImg_label);
    }
}

void MainWindow::on_qProcess_pushButton_clicked() {
    cv::Mat gray;
    cv::cvtColor(imgOrigRGB,gray,cv::COLOR_RGB2GRAY);
    cv::Canny(gray,imgResultGray,255,255/3);
    on_qDilateSize_spinBox_valueChanged(-1);
}

void MainWindow::on_qDilateSize_spinBox_valueChanged(int arg1) {
    cv::Mat cannyDilated;
    if (arg1<0)
        arg1=ui->qDilateSize_spinBox->value();
    cv::dilate(imgResultGray,cannyDilated,cv::Mat::ones(arg1,arg1,CV_8UC1));
    ShowImage(cannyDilated,ui->qResultImg_label);

}
```

Botón Load pulsado:
pide nombre archivo (el 4º argumento podría ser un filtro tipo "Images (.png *.jpg);;All files (*.*)"*
Si es válido, lee la imagen en la variable imgOrigRGB de la clase, y la muestra

Botón Camera pulsado:
captura un frame sobre la variable imgOrigRGB de la clase, y la muestra

Botón Process pulsado:
Convierte RGB a escala de gris, aplica filtro de canny, y llama a función para mostrar resultado según el valor del spinbox (los argumentos constantes de Canny podrían ser spinboxes para ajustar)

Spinbox Dilate cambiado (o llamada desde Process):
Dilata el resultado para una mejor visualización, y lo muestra

Programación con OpenCV

- Desde Python:
  - Instalar:

    ```
    pip3 install opencv-python
    ```

  - Importar en archivo .py:

    ```
    import cv2
    ```

  - Usar objetos de tipo cv2:

    ```
    imcap = cv2.VideoCapture(0)
    cv2.namedWindow("Camera")
    cv2.namedWindow("Result")
    ```

  - Usar funcionalidades para operaciones:

    ```
    success, img = imcap.read() # capture frame from video
    img=cv2.imread("filename.jpg") # obtain image from file
    cv2.imshow("Camera",img)
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgCanny = cv2.Canny(img,255,255/3)
    cv2.imshow("Result",imgCanny)
    ```

Programación con OpenCV

- Para ESP32
  - No hay versión oficial aún de OpenCV
  - Versión "Lite" (no oficial) de OpenCV e instrucciones para compilación: https://github.com/joachimBurket/esp32-opencv
  - ¡ OJO ! La limitada capacidad de computación del ESP32 hace que sólo se puedan usar en tiempo real las funcionalidades más básicas y sobre imágenes pequeñas.

Function name and arguments	BUILD_TYPE=Debug				BUILD_TYPE=Release			
	160x120	320x240	640x480	1024x768	160x120	320x240	640x480	1024x7
Edge detection								
Sobel	34	116	438	1129	14	50	187	497
Canny	80	256	886	ERR	32	108	375	ERR
Hough tranformations								
HoughLines	392	897	ERR	ERR	314	686	2121	ERR
HoughLines probabilistic	699	1652	ERR	ERR	603	1352	3765	ERR

Programación con OpenCV

- Algunos enlaces:
  - Curso completo OpenCV 4 para C++ de la Universidad Miguel Hernández:

    https://umh1782.umh.es/opencv/
  - Curso completo OpenCV 4 para Python de la Universidad Miguel Hernández:

    https://umh1782.umh.es/python/
  - Tutoriales variados:

    https://learnopencv.com/getting-started-with-opencv/

Indice

- Estructura del curso
- Calibración de cámaras
- Obtención de información 3D
- Programación C/C++ con OpenCV
- **Las nuevas técnicas: Inteligencia Artificial**
- Conclusiones

Usando Inteligencia Artificial

- Técnicas clásicas:
  - El desarrollador define la secuencia de operaciones
  - Hay que fijar los umbrales para cada operación
  - Los umbrales cambian con la escena, la iluminación, etc.
  - Difícil asentar algoritmos para condiciones cambiantes
- Inteligencia artificial:
  - El desarrollador define una serie de imágenes base y los resultados deseados (clasificación correcta)
  - Se entrena una red neuronal convolucional (CNN) para que minimice el error de clasificación
  - El entrenamiento es muy lento, requiere gran capacidad de computación y muchos datos
  - Una vez entrenada la CNN, la aplicación para nuevas imágenes es muy rápida y sencilla

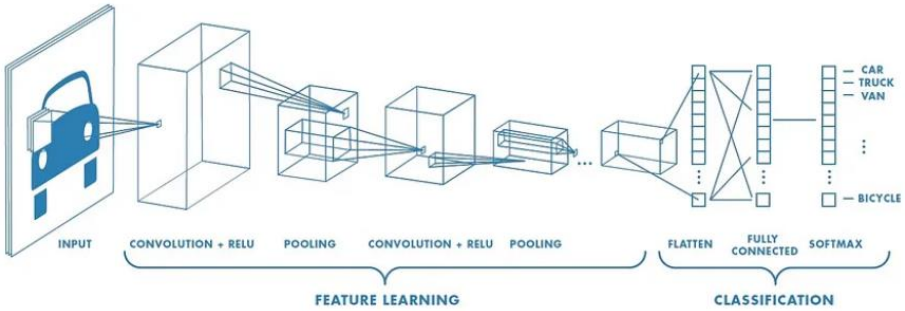
Usando Inteligencia Artificial

- Visión con I.A.:
  - Permite solucionar problemas inabordables con técnicas clásicas, al ser capaz de "reconocer" patrones muy complejos:
    - Reconocimiento de señales de tráfico, caras, objetos, defectos, tumores, …
    - Seguimiento de "targets"
    - Generación de imágenes a partir de características deseadas
    - …
  - Pero:
    - Los requerimientos para el entrenamiento no son abordables para pequeños desarrolladores: hay que basarse en redes pre-entrenadas y adaptar al caso particular
    - Las redes pre-entrenadas pueden tener "bias" hacia soluciones no deseadas
    - No generamos conocimiento humano: las interioridades de "cómo" y "por qué" permanecen ocultas

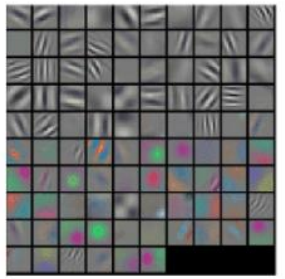
Usando Inteligencia Artificial

- Visión con I.A.: convolutional neural networks

https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

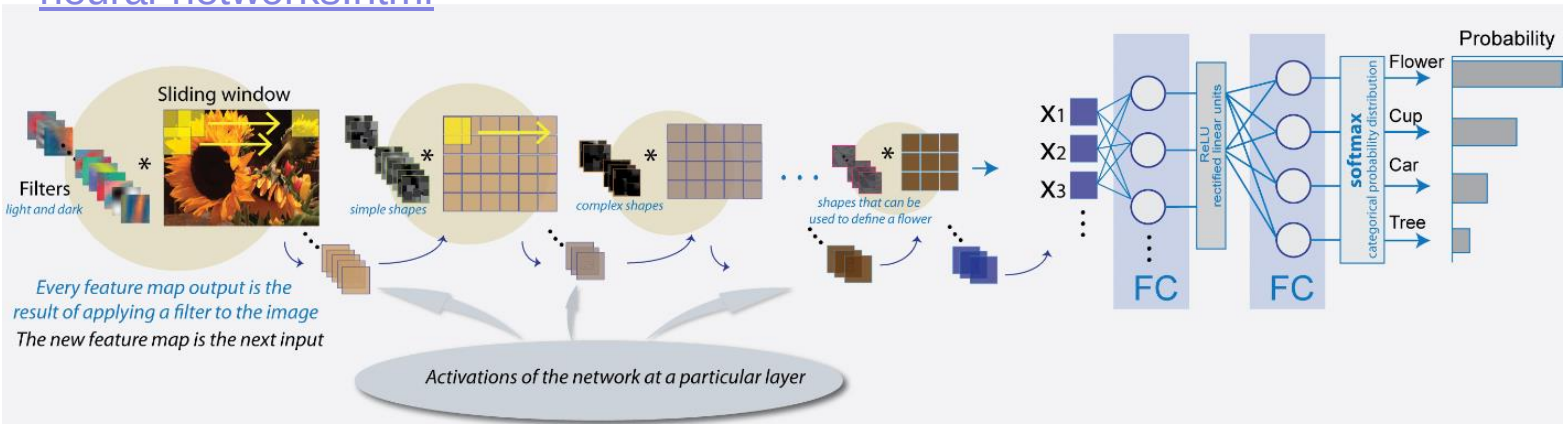


Alexnet 1st conv filters



https://lamaquinaoraculo.com/deep-learning/alexnet/

https://es.mathworks.com/help/deeplearning/ug/introduction-to-convolutional-neural-networks.html

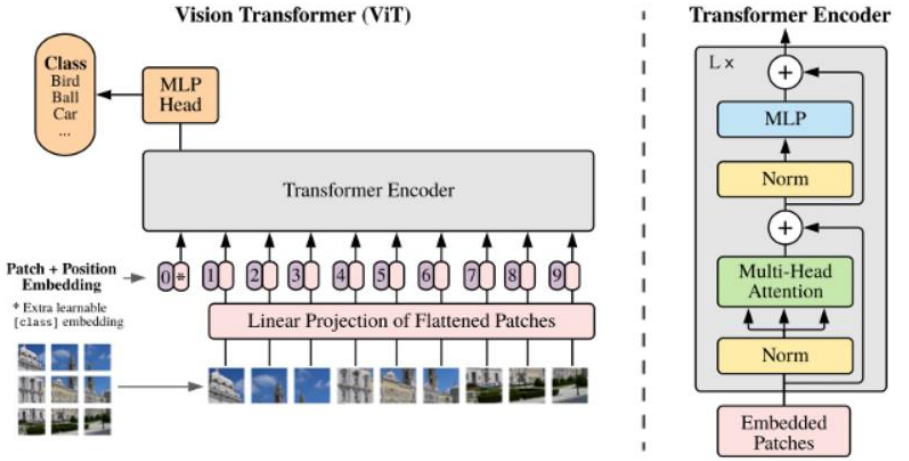


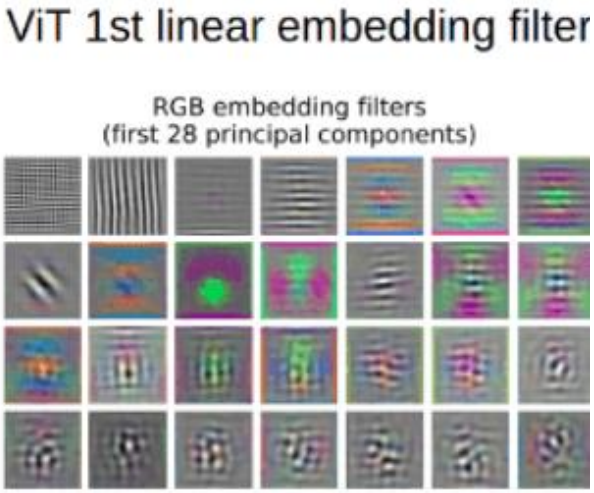
Usando Inteligencia Artificial

- Visión con I.A.: vision Transformers (ViT)

  https://theaisummer.com/vision-transformer/

  https://www.edge-ai-vision.com/2022/05/transformers-in-computer-vision/





Usando Inteligencia Artificial

- Visión con I.A. Ejemplo en Matlab:

  https://es.mathworks.com/help/vision/ug/image-category-classification-using-deep-learning.html

- Visión con I.A. Ejemplos en Python:

  https://asperbrothers.com/blog/image-recognition-in-python/

  https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/

  https://learnopencv.com/implementing-cnn-tensorflow-keras/

  https://medium.com/analytics-vidhya/image-processing-using-opencv-cnn-and-keras-backed-by-tensor-flow-c9adf22bb271

- Visión con I.A. en ESP32 (requiere ESP-IDF 5) :

  https://docs.espressif.com/projects/esp-dl/en/latest/esp32/introduction.html

Indice

- Estructura del curso
- Calibración de cámaras
- Obtención de información 3D
- Programación C/C++ con OpenCV
- Las nuevas técnicas: Inteligencia Artificial
- **Conclusiones**

Conclusiones

1) La Visión por Computador:
   - Herramienta muy útil para muchos dispositivos mecatrónicos
   - En continuo avance
   - Necesidades de procesamiento "exigentes"
2) Un buen comenzo: buenas imágenes
   - La etapa de captura de imagen es crucial: iluminación, lente, cámara, filtros, etc.
3) Dos aproximaciones:
   - Tratamiento clásico: pasos establecidos por el programador (preprocesamiento, realce, segmentación, extracción de características, clasificación). Dificultad de ajustes en condiciones cambiantes.
   - Machine learning (CNNs y ViT): resultados en casos complejos, necesidad de bancos de imágenes y redes pre-entrenadas,
4) Diseño e implantación:
   - Clásico: (Matlab , C/C++ o Python) (OpenCV)
   - CNNs: (Matlab , C/C++ o Python) (OpenCV, Caffe , SciKit, PyTorch, YOLO, …)
   - Equipos de visión compactos (programación visual por bloques): Cognex, Keyence, Sick, Omron, …