3D Vision

Marc Van Droogenbroeck

INTELSIG, Montefiore Institute, University of Liège, Belgium

Outline

Outline

Lecture

- Instructor: Marc Van Droogenbroeck
- Assistant: Philippe Latour
- Course Web page:
  `http://www2.ulg.ac.be/telecom/students/vision.html`
- Slides:
  `http://orbi.ulg.ac.be`

Evaluation

Motivation

- be capable to model a vision problem
- know the bases for object recognition

Examination and evaluation

- projects (including programming) in groups of 2 persons: written reports
- oral examination (related to theory) in January

Outline

Applications of computer vision

- Scene interpretation
- Detection and recognition of objects (faces, persons, etc).
- Motion analysis
- Counting operations
- etc

Outline

Image representation and fundamentals

- Elements of visual perception
  - Colors: representation and colorspaces
  - Transparency
- Data structure for images
- Resolution
- Examples of industrial applications:
  - Segmentation
  - Optical character recognition

Human visual system, light and colors I

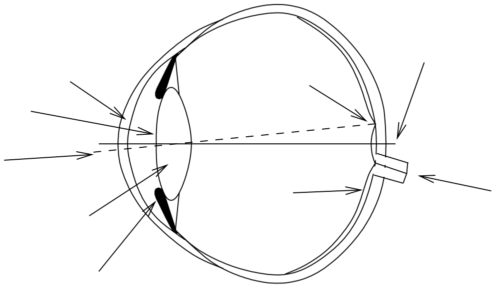


Figure : Lateral view of the eye globe (*rods* and *cones* are receptors located on the retina).

Human visual system, light and colors II

color	wavelength interval $\lambda\,[m]$	frequency interval $f\,[Hz]$
violet	$\sim$ 450–400 $[nm]$	$\sim$ 670–750 $[THz]$
blue	$\sim$ 490–450 $[nm]$	$\sim$ 610–670 $[THz]$
green	$\sim$ 560–490 $[nm]$	$\sim$ 540–610 $[THz]$
yellow	$\sim$ 590–560 $[nm]$	$\sim$ 510–540 $[THz]$
orange	$\sim$ 635–590 $[nm]$	$\sim$ 480–510 $[THz]$
red	$\sim$ 700–635 $[nm]$	$\sim$ 430–480 $[THz]$

Figure : Visible colors (remember that $\lambda = \frac{3\times10^8}{f}$).

Human visual system, light and colors III

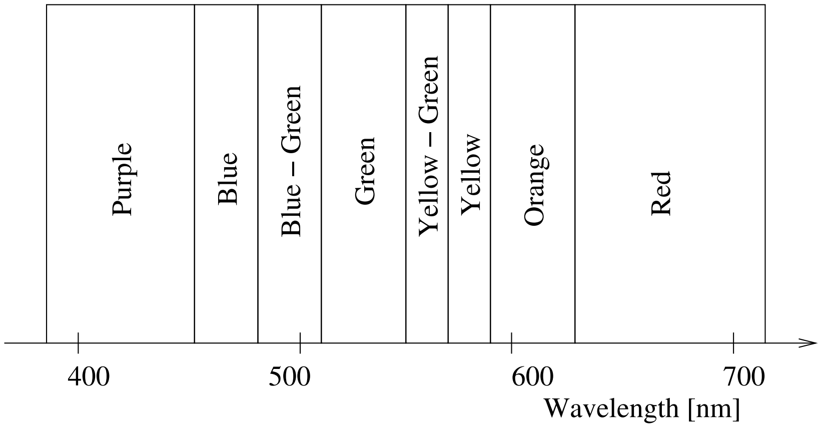


Figure : Colors on the visible spectrum.

Frequency representation of colors

$$\int_{\lambda} L(\lambda)\, d\lambda \qquad (1)$$

Impossible from a practical perspective because this would require *one sensor for each wavelength.*

Solution: use colorspaces

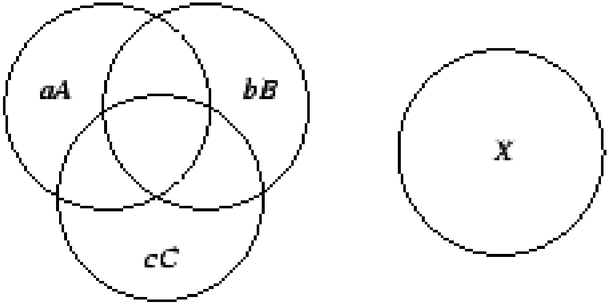


Figure : Equalization experiment of colors. The aim is to mix A, B, and C to get as close as possible to X.

Frequency representation of colors

$$\int_{\lambda} L(\lambda)\, d\lambda \tag{1}$$

Impossible from a practical perspective because this would require *one sensor for each wavelength.*

Solution: use colorspaces

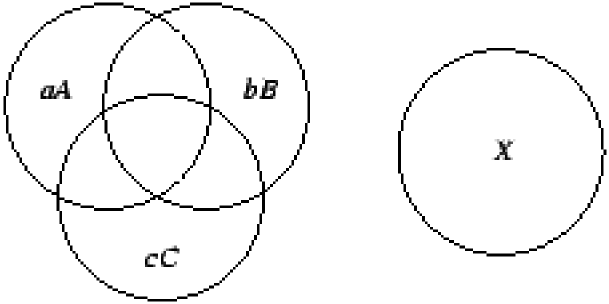


Figure : Equalization experiment of colors. The aim is to mix A, B, and C to get as close as possible to X.

The RGB additive colorspace

Three fundamental colors: red R $(700\,[nm])$, green G $(546,1\,[nm])$ and blue B $(435,8\,[nm])$,

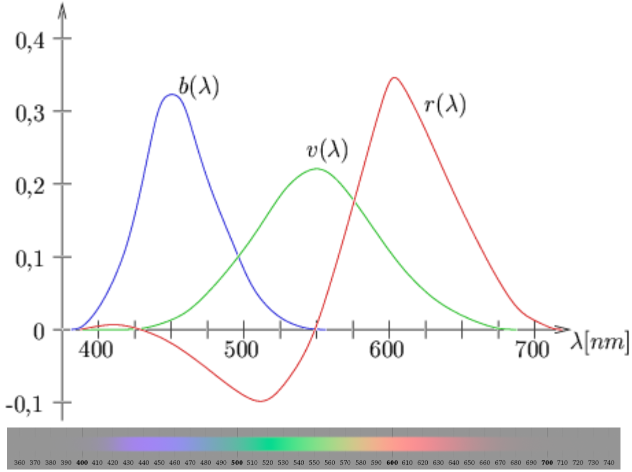
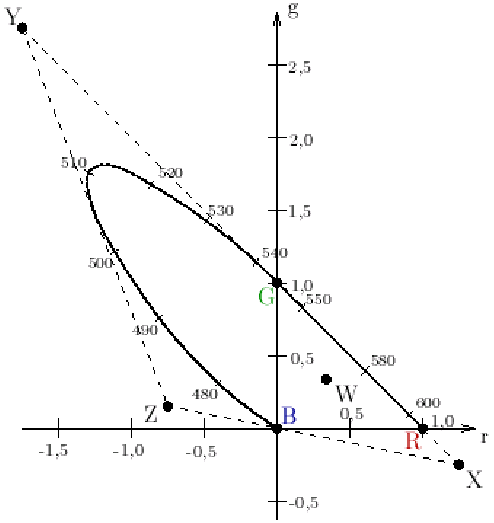


Figure : Equalization curves obtained by mixing the three fundamental colors to simulate a given color (wavelength).

CIE chromatic diagram for RGB



Notion of intensity

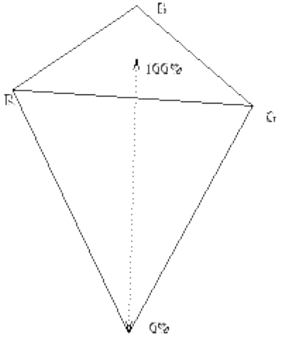


Figure : Pyramid derived from an RGB color representation.

Towards other colorspaces: the XYZ colorspace I

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2,769 & 1,7518 & 1,13 \\ 1 & 4,5907 & 0,0601 \\ 0 & 0,0565 & 5,5943 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2}$$

$$\begin{aligned} x &= \frac{X}{X+Y+Z} && (3) \\ y &= \frac{Y}{X+Y+Z} && (4) \\ z &= \frac{Z}{X+Y+Z} && (5) \\ x+y+z &= 1 && (6) \end{aligned}$$

Towards other colorspaces: the XYZ colorspace II

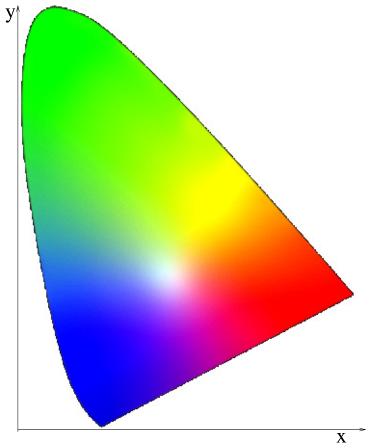
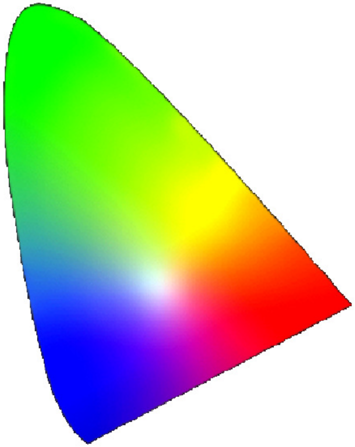


Figure : Approximative chromatic colorspace defined by two chrominance variables x and y.

Luminance

Luminance: $Y = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B$



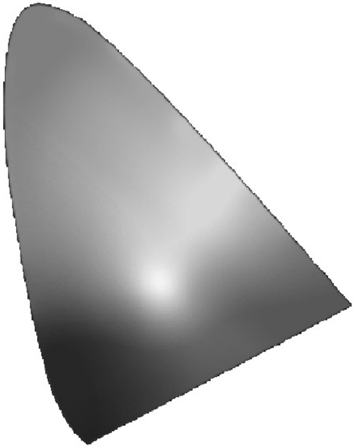


Figure : *xy* chromatic diagram and maximal luminance for each color.

Luminance + two chrominances

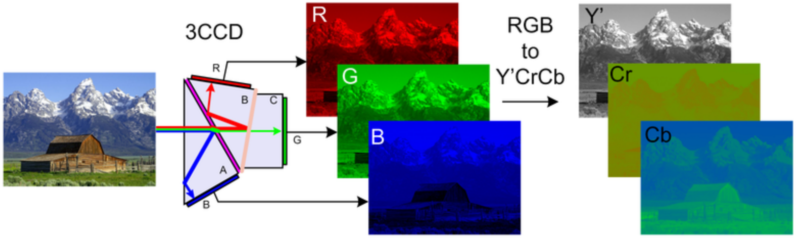


Figure : Acquisition of a $Y\ C_b\ C_R$ signal [Wikipedia]

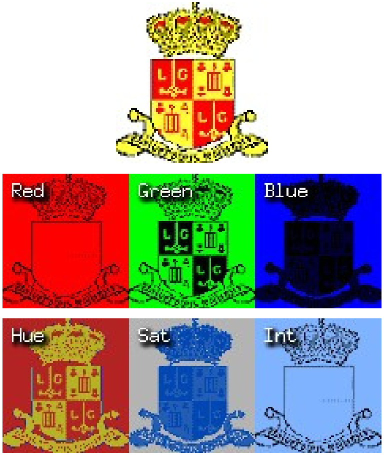
There are variations, such as the $Y\ U\ V$ colorspace, mainly developed for compression:

1. information concentration in the Y channel $\Rightarrow$ better compression.
2. better decorrelation between channels).

The HSI colorspace

Colorspace that has a better physical meaning:

- hue
- saturation
- intensity



Other colorspaces

- a subtractive colorspace: Cyan, Magenta, and Yellow (CMY)
- Luminance + chrominances (YIQ, YUV or YC_bC_r)

In practice,

Hexadecimal				R	G	B
00	00	00	⬛	0	0	0
00	00	FF	🟦	0	0	255
00	FF	00	🟩	0	255	0
00	FF	FF	🟦	0	255	255
FF	00	00	🟥	255	0	0
FF	00	FF	🟪	255	0	255
FF	FF	00	🟨	255	255	0
FF	FF	FF	⬜	255	255	255

Table : Definition of color values and conversion table between an hexadecimal and an 8-bits representation of colors.

Bayer filter I

A Bayer filter mosaic is a color filter array for arranging RGB color filters on a square grid of photo sensors.

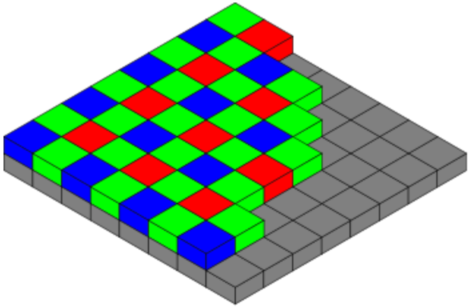


Figure : The Bayer arrangement of color filters on the pixel array of an image sensor. [Wikipedia]

Bayer filter II

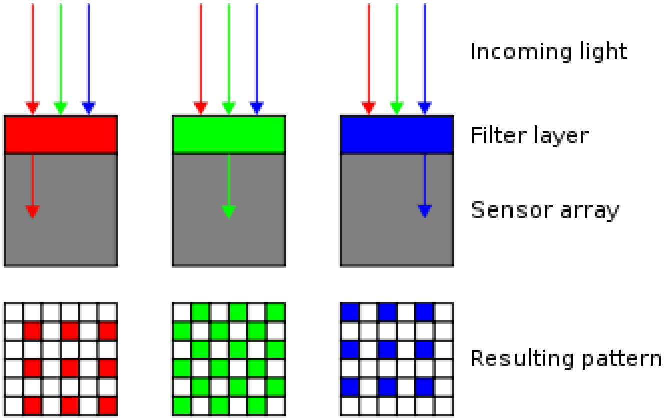


Figure : Profile/cross-section of sensor. [Wikipedia]

Bayer filter: practical considerations

- Most mono-sensor cameras use the Bayer pattern, except for professional 3CCD cameras (three sensor planes + prism to divide the incoming light)
- The filter pattern is 50% green, 25% red and 25% blue. Why?
- We only have one value per pixel. Other values are re-built by interpolation, but they might not even exist... !
- For compression or processing,
  - 1 sensor plane $\Rightarrow$ normally only one byte to process. Possible if the processing is very close to the sensor. Otherwise, there is no information about the real observed values.
  - 3 sensor planes $\Rightarrow$ 3 planes to process or to compress. Expected compression rate: $3\times$ lower than for a single sensor.
- It might be wiser, for processing, to have a black-and-white (or a monochromatic, such as red) camera, instead of a color camera.

What defines the color of an object?

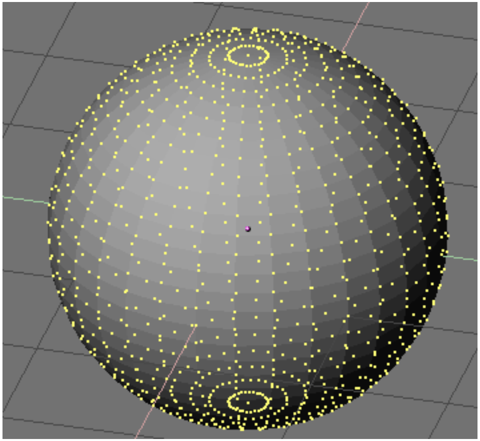


Figure : A synthetic 3D object. Shadows and surfaces with varying reflective coefficients model a 3D object.

Visual effects

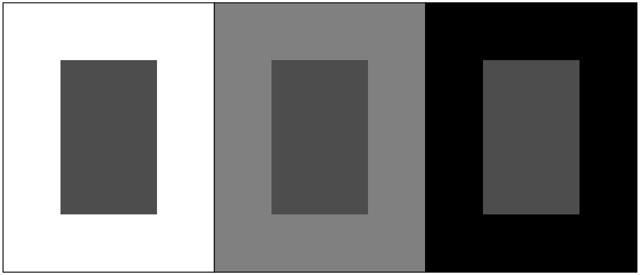


Figure : Illustration of a masking visual effect.

Transparency bits

Let

- $i(x, y)$ be the value of the image at location (x, y)
- $t(x, y)$ be the transparency (defined by 1 to 8 bits)
- $o(x, y)$ be the output value, after applying transparency

Applying transparency consists to calculate: $o(x, y) = \frac{t(x,y)}{255} i(x, y)$

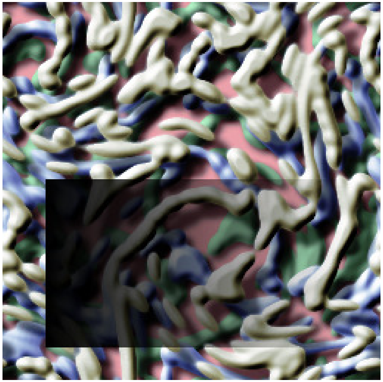


Figure : Transparency bits have been applied inside a rectangle.

Sampling grid and frame organization

- Each sample located on a grid is named a *pixel* (which stands for *picture element*).
- There are two common sampling grids and they induce certain types of connectivity.

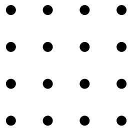
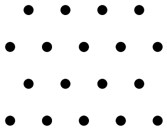
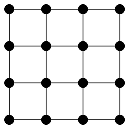
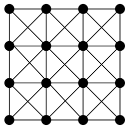
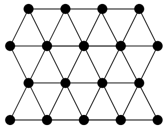
Square grid		Hexagonal grid
		
4-connectivity	8-connectivity	6-connectivity
		

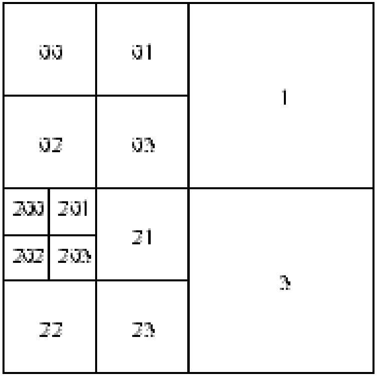
Table : Types of grid and associated connectivities.

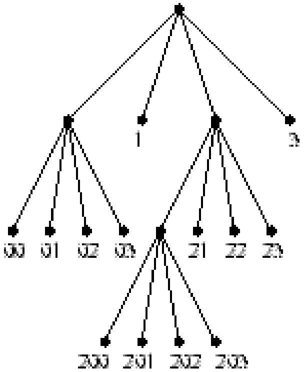
Typology of images and videos

- 2D. This types refers to a “classic” image and is usually expressed as a 2D array of values. It might represent the luminance, a color, depth, etc.
- 3D. 3D images are obtained with devices that produce 3D images (that is with x, y, z coordinates). Medical imaging devices produce this type of images.
- 2D+t. t refers to time. Therefore, $2D + t$ denotes a video composed over successive 2D images, indexed by t.
- 3D+t. $3D + \mathrm{t}$ images are in fact animated $3D$ images. A typical example is that of animated 3D graphical objects, like that produced by simulations.

Data structure for dealing with images

- Typical data structure for representing images: matrices (or 2D tables), vectors, trees, lists, piles, ...
- A few data structures have been adapted or particularized for image processing, like the *quadtree*.





The bitplanes of an image

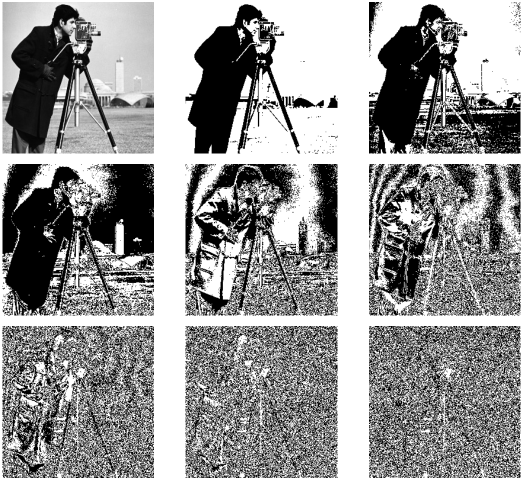
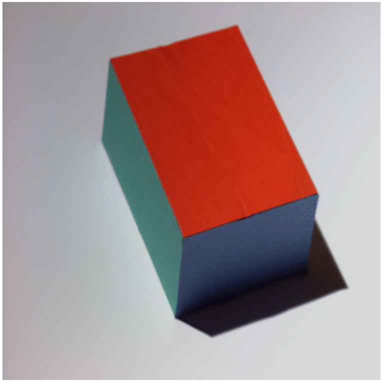
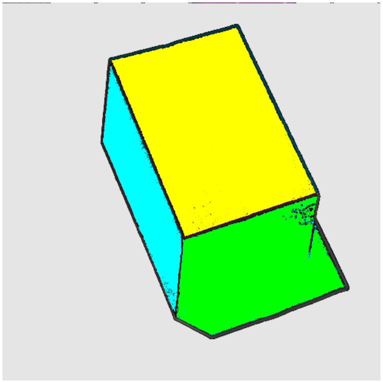


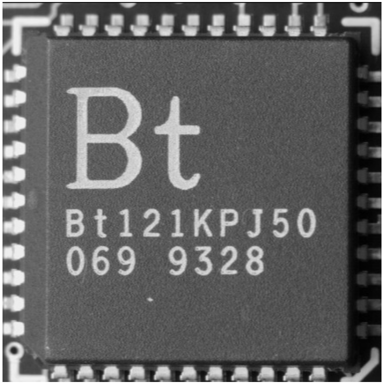
Table : An original image and its 8 bitplanes starting with the Most Significant Bitplane (MSB).

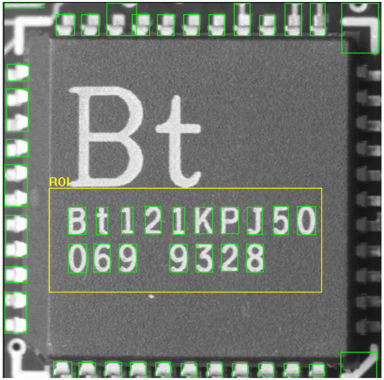
Image segmentation





Character recognition





Several successive stages:

- Selection of a Region of Interest (ROI). Processing is limited to that area.
- Detection of edges (contours).
- Identification and classification of characters.

Outline

Building a mathematical model for cameras I

A camera projects a 3D world onto a 2D image.
$\Rightarrow$ this is the concept of *projection*.

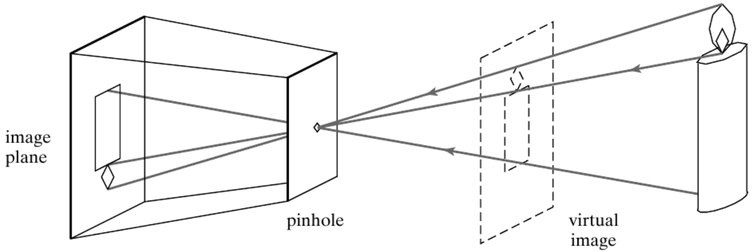
Specificities:

- when we apply a *transformation*, some characteristics are preserved. For example, when you translate an object, its dimensions and angles are preserved.
- translations and rotations (which can be expressed as the product of a matrix on the $(X, Y, Z))^T$ real world coordinates) preserves distances, angles, etc. These leads to so-called Euclidean transformations.
- But what is preserved in general?
    - distances? angles? parallelism? *alignment of pixel ($\equiv$ lines)*?

Building a mathematical model for cameras II

The real problem if that we have some ambiguities when we project an 3D object to a 2D plane.

One of the simplest model (and most common): *pin-hole camera* (central projection)

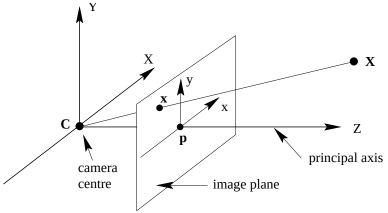


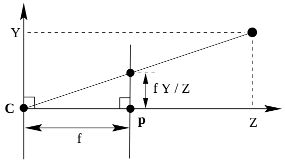
What is preserved?

- distances? angles? "parallelism"? Not really.
- *alignment of pixel* ($\equiv$ lines)? Yes

Pinhole camera model

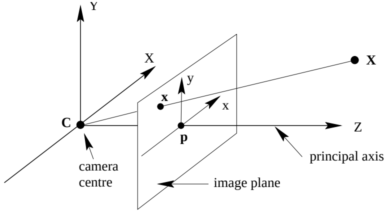
- All the light rays convergent to a unique point (*camera center*) before arriving on the sensor plane.

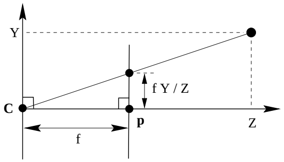




- Vocabulary terms:
  - *camera center*: center of the central projection.
  - *image plane*: plane of the sensor.
  - *principal axis*: line passing through the camera center and orthogonal to the image plane.
  - *principal* point: intersection between the principal axis and the image plane.
  - f is called the *focal length*.

Mathematical model of the pinhole camera I





If $\mathbf{X} = (X, Y, Z)^T$ is a point in space and $\mathbf{x} = (x, y)^T$ is its projection on the image plane, then similar triangles gives

$$\frac{x}{f} = \frac{X}{Z} \quad \text{and} \quad \frac{y}{f} = \frac{Y}{Z} \tag{7}$$

Mathematical model of the pinhole camera II

In a *convenient* matrix form:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{8}$$

where λ is equal to the depth Z. Remember that we don't know the depth Z from the observation in the place image (depth ambiguity).

Homogeneous coordinates I

Idea: new representation $\Rightarrow$ *homogeneous coordinates*

$$(x, y) \equiv (x, y, 1) \tag{9}$$

By definition, we assume that

$$(x, y) \equiv (\lambda x, \lambda y, \lambda) \tag{10}$$

so that all pixels with varying λ are equivalent.
But $(x, y, 0)$ is not equivalent to (x, y). It is a special point.

- Homogeneous coordinates are defined as a vector ended by 1. For example: $(x, y, 1)^T$. This yields for any dimension.
- A point of $\mathbb{R}^n$ is represented by a vector of size $n+1$. Example:

$$\mathbf{x} = (x_1, x_2, x_3)^T \in \mathbb{R}^3 \mapsto (\frac{x_1}{x_3}, \frac{x_2}{x_3})^T \in \mathbb{R}^2$$

Homogeneous coordinates II

- A line in a plane is represented by the following equation $ax + by + c = 0$. With homogeneous coordinates, this becomes:
$$\mathbf{l} = (a,\ b,\ c)^T$$
- A point belongs to a line if and only if $\mathbf{x}^T\mathbf{l} = 0$.
- Intersection between two lines: $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$ where $\times$ is the product between vectors.

Camera matrix

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathtt{P}\mathbf{X}$$

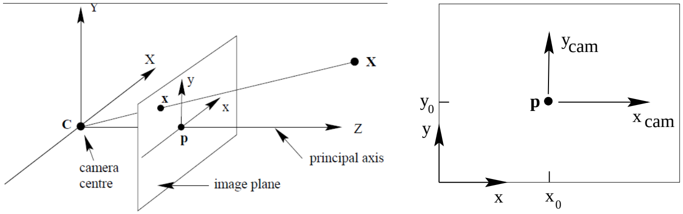
- $\mathtt{P}$ is *camera (projection) matrix*
- For convenience, $\mathtt{P}$ is decomposed as follows:

$$\mathtt{P} = \mathtt{K}\left[\mathtt{I}_{3\times 3} | \mathbf{0}_{3\times 1}\right]$$

- $\mathtt{K} = \mathrm{diag}(f,\ f,\ 1)$ is the calibration matrix.

Generalization of the pinhole camera model I

[1] The central point is not always located at (0, 0) in the image plane.



- If $(x_0, y_0)^T$ are the coordinates of the principal point, the projection becomes:

$$(X,\ Y,\ Z)^T \mapsto (f\frac{X}{Z} + x_0,\ f\frac{Y}{Z} + y_0)^T$$

Generalization of the pinhole camera model II

- The matrix form is:

$$\lambda \begin{bmatrix} x + Zx_0 \\ y + Zy_0 \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & x_0 & 0 \\ 0 & f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- The matrix $\mathtt{K}$ is thus:

$$\mathtt{K} = \begin{bmatrix} f & & x_0 \\ & f & y_0 \\ & & 1 \end{bmatrix}$$

[2] Non rectangular light-elements on the sensor $\rightarrow$ *skew*.
Non *squared elements* modify the aspect ratio $\rightarrow \alpha_x$ and α_y.
Therefore

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

Intrinsic parameters

In a refined camera model,

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

This matrix has 5 parameters. These parameters are called *intrinsic parameters*.
They characterize a camera and should be estimated for each camera separately. But once they are known, there is no need to estimated them again!

Calibration

Definition

A camera is said to be *calibrated* if

$$\mathtt{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

is known.

In general:

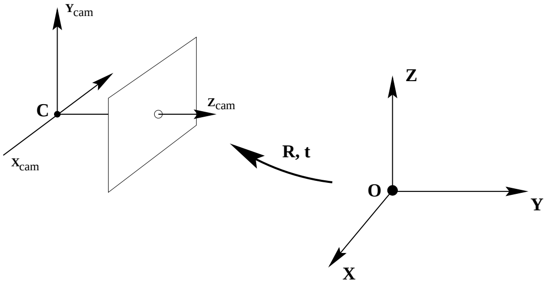
- $\alpha_x \simeq \alpha_y$
- $s \simeq 0$
- x_0 and y_0 close to 0 pixel (typically a few pixels maximum).

But we have a problem: **we don't know where the center of camera is located**... So there is no way to measure $(X, Y, Z, 1)^T$.

Extrinsic parameters I

Points in the 3D world need to be expressed in a system coordinate different from that of the camera (which is not known).

- Both coordinate systems are related by a rotation and a translation:



Extrinsic parameters II

For a translation:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} X_0 - t_1 \\ Y_0 - t_2 \\ Z_0 - t_3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -t_1 \\ 0 & 1 & 0 & -t_2 \\ 0 & 0 & 1 & -t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \quad (11)$$

$$= \begin{bmatrix} \underline{\mathtt{I}} & -t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \quad (12)$$

More generally (translation + rotation):

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \underline{\mathtt{R}}^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \underline{\mathtt{I}} & -t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \quad (13)$$

Extrinsic parameters III

In conclusion:

$$\mathbf{X_c} = \begin{bmatrix} \underline{\mathtt{R}}^T & -\underline{\mathtt{R}}^T t \\ 0 & 1 \end{bmatrix} \mathbf{X_0} \tag{14}$$

where **R** is a rotation matrix and t a translation vector. This is not a projection, but an Euclidean transform between coordinate systems.
By substitution:

$$\lambda \mathbf{x} = \mathtt{K}\mathtt{R}^T [\mathtt{I}| - t] \mathbf{X_0} = \mathtt{P}\mathbf{X} \tag{15}$$

where $\mathtt{P} = \mathtt{K}\mathtt{R}^T [\mathtt{I}| - t]$.

Conclusions on $\mathtt{P}$

Definition

A camera represented with a camera matrix of the form $\mathtt{P} = \mathtt{K}\mathtt{R}^T[\mathtt{I}| - t]$ is called **normalized camera**.

$\mathtt{P}$ is a 3×4 matrix, which is subject to scale ambiguity. Therefore, $\mathtt{P}$ has 11 degrees of freedom (unknown parameters):

- 5 intrinsic parameters (*related to camera itself*; the manufacturer can computed them and give them to the user who buys the camera).
- 6 extrinsic parameters (*related to* the choice of an external *coordinate system*).

Calibration procedure I

How do we find the parameters of $\mathtt{P}$?

- We need correspondences between some 3D positions on their location on the projected 2D image $\mathbf{X}_i \leftrightarrow \mathbf{x}_i = (x_i, y_i, z_i)$.
- As a result

$$\mathbf{x}_i = \mathtt{P}\mathbf{X}_i \Rightarrow \mathbf{x}_i \times \mathtt{P}\mathbf{X}_i = 0$$

Let $\mathbf{P}^j$ be the vector with the jth line of $\mathtt{P}$,

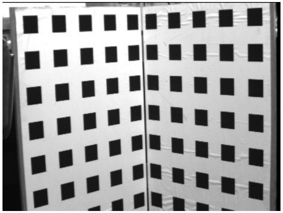
$$\begin{bmatrix} \mathbf{0}^T & -z_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ z_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

- Two of these equations are linearly independent. But how many correspondences do we need to solve this linear systems?

At least 6

Calibration procedure II

- In practice, we have an over-determined system of linear equations $\mathtt{A}\mathbf{p} = \mathbf{0}$ if there is no noise. But due to noise, we use an optimization procedure, such as the minimum square optimization
- Use of a precise and well-known 3D calibration pattern



Decomposition of camera matrix

How can we determine $\mathtt{K}$, $\mathtt{R}$, and the camera center $\mathbf{C}$?

- If $\mathbf{p}_i$ is the ith column of $\mathtt{P}$ and $\mathbf{C} = (X,\ Y,\ Z,\ T)^T$, then:

$$\begin{aligned} X &= \det([\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4]) & Y &= \det([\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4]) \\ Z &= \det([\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4]) & T &= \det([\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3]) \end{aligned}$$

- We can then compute a matrix $\mathtt{M}$ such that $\mathtt{P} = [\mathtt{M}| - \mathtt{M}\mathbf{C}]$
- $\mathtt{K}$ and $\mathtt{R}$ are found by a RQ matrix decomposition of $\mathtt{M}$

Outline

Enjeux de la géométrie 3D

L'analyse 3D comporte plusieurs aspects importants:

- reconstruction 3D de la scène
  - reconstruction dense (pour tous les points de l'image, on détermine la profondeur)
  - reconstruction non dense
- détection de points caractéristiques, propres à la structure des objets
- la présence ou non de mouvement de la caméra ou des éléments de la scène

Ces deux derniers aspects sont intimement liés!

Méthodes d'acquisition 3D

Il y a plusieurs moyens d'obtenir la perspective d'une scène par caméras:

- méthodes directes:
  - caméra profondeur (on parle de caméra 3D)
  - vision stéréoscopique (au moyen de deux caméras) ou vision multi-caméras
- méthodes indirectes (plus ou moins tombées dans l'oubli avec l'apparition des caméras profondeur):
  - "depth from motion"
  - "depth from focus"
  - "depth from shadow"
  - ...

Depth cameras

There are two *acquisition* technologies for depth-cameras, also called range- or 3D-cameras:

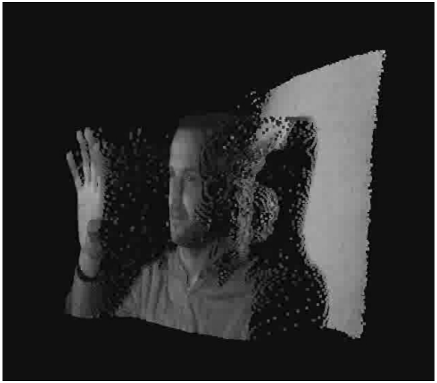
- measurements of the deformations of a pattern sent on the scene (*structured light*).

  - first generation of the Kinects

- measurements by *time-of-flight* (ToF). Time to travel forth and back between the source led (camera) and the sensor (camera).

  - Mesa Imaging, PMD cameras
  - second generation of Kinects

Illustration of a depth map acquired with a range camera



Systèmes à caméras multiples

Trois concepts s'entremêlent:

- calibration
- mouvement éventuel de la caméra
- la structure 3D de la scène

Theorem

Soit une série d'images d'une caméra en mouvement non calibrée pour laquelle on est parvenu à établir la correspondance entre certains points, il n'est possible de reconstruire la scène qu'à une transformée projective près.

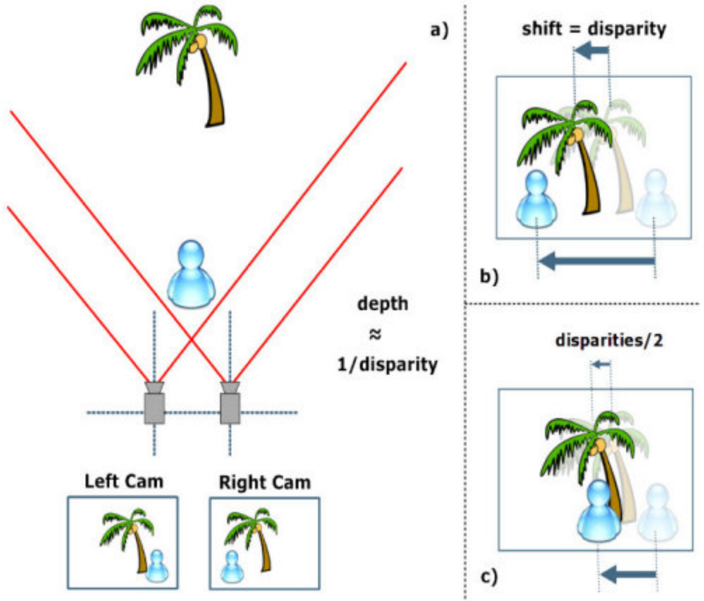
Par la suite, on s'intéresse exclusivement à la mise en correpondance de points entre différentes vues de caméras fixes.

Démarche pour la reconstruction dans un système stéréoscopique

De manière générale, on cherche à établir des correspondances entre des points correspondants dans les deux vues. Plusieurs scénarios sont possibles, dont les suivants:

1. *les caméras sont alignées* (mécaniquement ou après transformation) $\rightarrow$ calcul de la *carte de disparité*.
2. *les caméras sont disposées arbitrairement* $\rightarrow$ deux points correspondants sont liés entre eux par la *matrice fondamentale*. Cette matrice établit une contrainte; elle ne suffit pas à résoudre le problème de la reconstruction (et il faut calibrer les caméras, etc). Les contraintes sont développées dans le cadre de la *géométrie épipolaire*.
3. *tous les points à reconstruire sont situés dans un même plan* $\rightarrow$ cette contrainte permet de trouver une matrice qui projette chaque point d'une vue sur le point correspondant dans l'autre vue. On parle d'*homographie*.

Stéréoscopie: principes de la disparité (caméras alignées)



Calculer la disparité

Pour deux caméras qui ne diffèrent que par leur position horizontale, la différence horizontale entre deux points correspondant donne une information de profondeur.

Attention : les caméras et leur alignement ne sont pas parfaits!

Carte de disparité

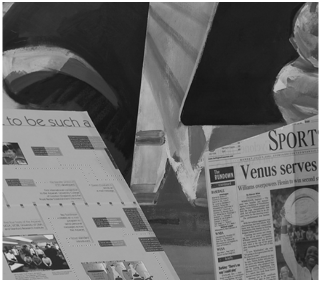
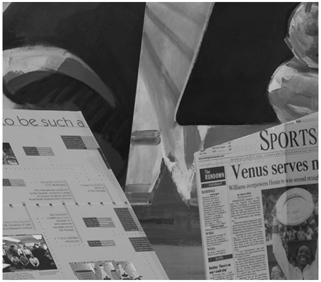
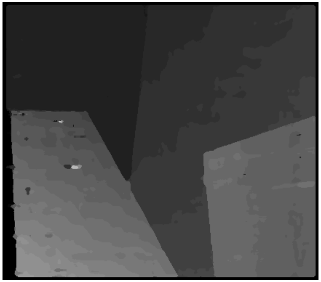
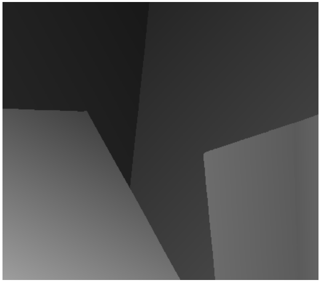
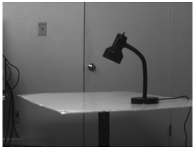
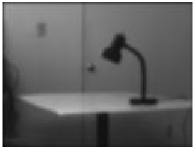
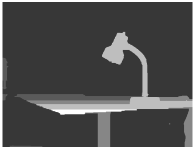
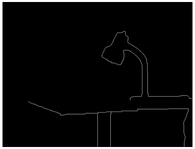





Illustration de la disparité et difficultés


vue gauche


vue droite


disparité


contours

Le calcul de la disparité est délicat:

- à proximité des bords (diffraction)
- dans les zones sans texture

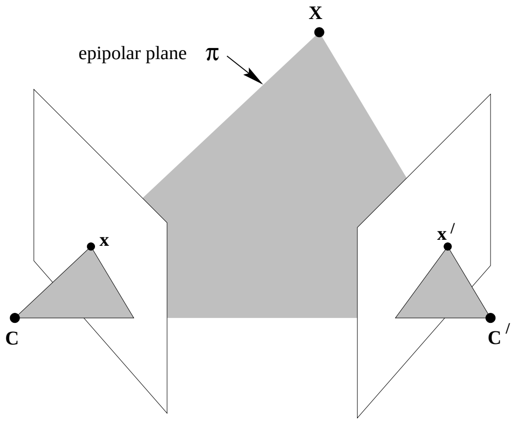
Discussion

Avantages (+) ou inconvénients (-) dune reconstruction ($\equiv$ calcul de la profondeur) par calcul de disparité

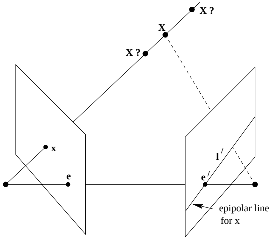
- (+) estimer la profondeur sur base de la disparité est simple (calcul de correspondance le long d'une ligne)
- (-) il faut aligner les caméras,
  - soit physiquement,
  - soit par software. Dans ce cas, il faut ré-échantillonner une image. D'où problème de précision.

On a besoin d'une alternative pour la reconstruction quand les caméras sont disposés arbitrairement.

Caméras quelconques: géométrie épipolaire et matrice fondamentale I



Caméras quelconques: géométrie épipolaire et matrice fondamentale II



Vocabulaire:

- *Epipole*: point d'intersection de la ligne joignant les centres focaux des caméras avec le plan de l'image
- *Ligne de base*: ligne joignant les épipoles
- *Plan épipolaire*: plan contenant la ligne de base
- *Ligne épipolaire*: intersection entre un plan épipolaire et le plan de l'image

Difficultés pour la reconstruction stéréoscopique

Objectif: mettre en correspondance les observations dans les deux images pour en déduire un maximum de points (X, Y, Z).

- Calibration:
  - a priori, chaque caméra a ses propres paramètres de calibration:
    - solution: détermination de la *matrice de calibration* (avec des *paramètres intrinsèques* et *extrinsèques*).
  - on dispose de deux caméras
    - solution: relation entre les projections dans les deux plans au moyen de la *matrice fondamentale* ou de la *matrice essentielle*.
- Disposition des caméras:
  - *fixe*: calibration possible en usine.
  - *changeante* ou *inconnue*: on pourrait isoler les paramètres intrinsèques, mais le reste nécessite une calibration.
- Structure des objets 3D: elle peut être quelconque ou comporter des ambiguïtés.
  - solution “réaliste”: faire des hypothèses a priori sur l’objet.

Calcul menant à la matrice fondamentale

Soit un point physique $\mathbf{X}$, dans chacune des vues:

$$\begin{aligned} \lambda_1 \mathbf{x}_1 &= \mathtt{P}_1 \mathbf{X} = [\mathtt{A}_1 | \, b_1] \mathbf{X} && (16) \\ \lambda_2 \mathbf{x}_2 &= \mathtt{P}_2 \mathbf{X} = [\mathtt{A}_2 | \, b_2] \mathbf{X} && (17) \end{aligned}$$

Si on utilise la première caméra pour résoudre en X, Y, Z:

$$\lambda_1 \mathbf{x}_1 = \mathtt{P}_1 \mathbf{X} = [\mathtt{A}_1 | \, b_1] \mathbf{X} = \mathtt{A}_1 \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + b_1 \qquad (18)$$

implique que

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathtt{A}_1^{-1} (\lambda_1 \mathbf{x}_1 - b_1) \qquad (19)$$

et en insérant ceci dans l'équation de la seconde caméra:

$$\lambda_2 \mathbf{x}_2 = \mathtt{A}_2 \mathtt{A}_1^{-1} (\lambda_1 \mathbf{x}_1 - b_1) + b_2 = \lambda_1 \mathtt{A}_{12} \mathbf{x}_1 + (-\mathtt{A}_{12} b_1 - b_2) \qquad (20)$$

Donc, $\mathbf{x}_1$ et $\mathbf{x}_2$ sont linéairement dépendants.

Matrice fondamentale

Contrainte épipolaire

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}^T \mathtt{F} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

- La **matrice fondamentale** $\mathtt{F}$ est une matrice 3×3 qui lie des points correspondants dans des images stéréo. Autrement dit, on a un lien entre les projections dans les deux plans caméra.
- $\mathtt{F}\mathbf{x}$ donne la ligne épipolaire sur laquelle se trouve le point $\mathbf{x}'$ dans l'autre image
- $\mathbf{x}'\mathtt{F}\mathbf{x} = 0$ pour toutes paires de points correspondants $\mathbf{x} \leftrightarrow \mathbf{x}'$.

Caractéristiques:

- 7 degrés de liberté (matrice de rang 2)
- déterminant nul
- elle est définie à un facteur près

Lien entre la matrice fondamentale et les matrices de calibration des caméras I

On peut obtenir la matrice fondamentale en repartant des deux matrices des deux caméras. Supposons que:

1. l'on isole les paramètres *intrinsèques* et *extrinsèques* de chaque caméra et
2. le système de coordonnées absolu soit placé sur une des deux caméras

$$\mathtt{P} = \mathtt{K}\,[\mathtt{I}|0] \quad \text{et} \quad \mathtt{P}' = \mathtt{K}'\,[\mathtt{R}|t] \tag{21}$$

La matrice fondamentale est alors obtenue par [sans démonstration]

$$\mathtt{F} = \mathtt{K}'^{-T}[t]_\times \mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-T}\mathtt{R}[\mathtt{R}^T t]_\times \mathtt{K}^{-1} \tag{22}$$

où

$$[a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

Lien entre la matrice fondamentale et les matrices de calibration des caméras II

Si les paramètres intrinsèques sont connus, on peut normaliser les coordonnées pour éliminer les paramètres intrinsèques (par des opérations du type de $\hat{u} = \mathtt{K}^{-1} u$ et $\hat{u}' = \mathtt{K}'^{-1} u'$) et se ramener à

$$\mathtt{P} = [\mathtt{I}|0] \quad \text{et} \quad \mathtt{P}' = [\mathtt{R}|t] \tag{23}$$

La matrice fondamentale se simplifie et produit la matrice essentielle

$$\mathtt{E} = [t]_\times \mathtt{R} = \mathtt{R}[\mathtt{R}^T t]_\times \tag{24}$$

L'avantage est que seuls les paramètres extrinsèques interviennent.

Calcul de la matrice fondamentale. Même genre de problématique que pour l'étalonnage de la caméra!

- On trouve un certain nombre de correspondances $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ (minimum 7)
- On construit un système linéaire d'équation à partir des équations $\mathbf{x}'_i \mathtt{F} \mathbf{x}_i = 0$
- On résoud le système pour obtenir $\mathtt{F}$

Il existe de nombreux algorithmes un peu plus compliqués mais beaucoup plus robustes et précis.

La matrice fondamentale établit une contrainte entre les deux vues. Mais elle ne suffit pas pour résoudre le problème de la reconstruction!

Reconstruction de la scène 3D I

1. Trouver des correspondances entre les deux images
2. Calculer la matrice fondamentale
3. Calculer les caméras à partir de la matrice fondamentale :

$$\mathtt{P} = [\mathtt{I}|\mathbf{O}] \qquad \mathtt{P}' = \left[[\mathbf{e}']_\times \mathtt{F}|\mathbf{e}'\right]$$

où

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

et

$$\mathtt{F}^T \mathbf{e}' = \mathbf{0}$$

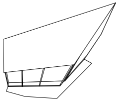
Reconstruction de la scène 3D II

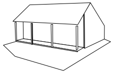
4. Pour chaque correspondance $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, trianguler le point $\mathbf{X}_i$:

$$\mathtt{P}\mathbf{X}_i \times \mathbf{x}_i = 0 \quad \text{et } \mathtt{P}'\mathbf{X}_i \times \mathbf{x}'_i = 0$$

   On obtient 4 équations pour 3 inconnues.

Attention! Cette reconstruction n'est bonne qu'à une transformation projective près.





Références proposées pour la géométrie 3D

- R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.
- Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models.* Springer, 2006.

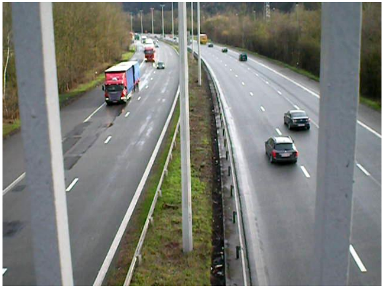
Outline

Segmentation by background subtraction I

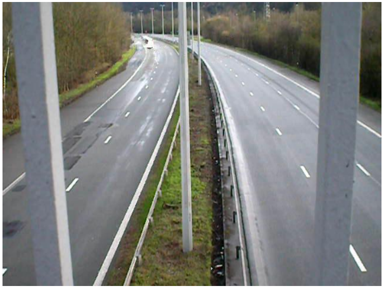
- Objective:
  - separate the foreground (pixels “in motion”) from the background (“static” pixels) OR
  - separate the foreground (pixels “close to the camera”) from the background (pixels in the back of the “scene”) OR
  - both !
- Steps:

[Initialization]	build a *reference frame* or a *model* for the background.
[Subtraction]	*compare* the current frame to the reference frame or model, and “*subtract*” the frame to get a binary image indicating pixels who have changed.
[Updating]	*update* the reference frame or model.

Segmentation by background subtraction II



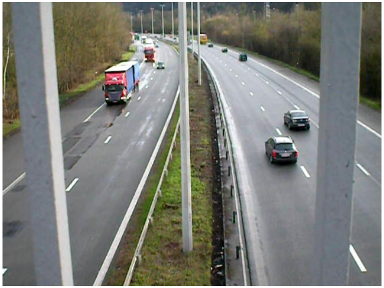
One frame in the sequence



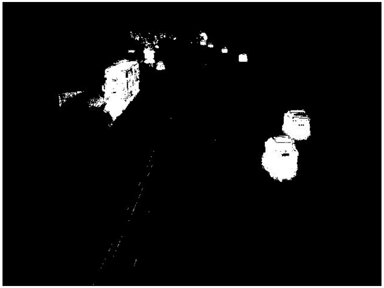
Built reference frame

Figure : Building a reference frame or a model.

Segmentation by background subtraction III



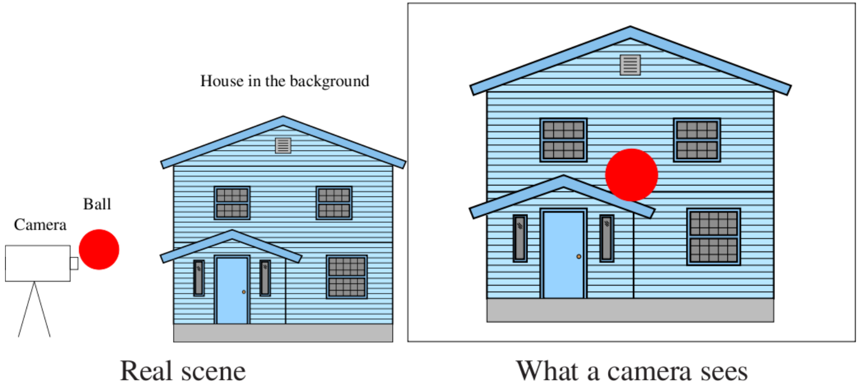
Original image



Features detected by ViBe

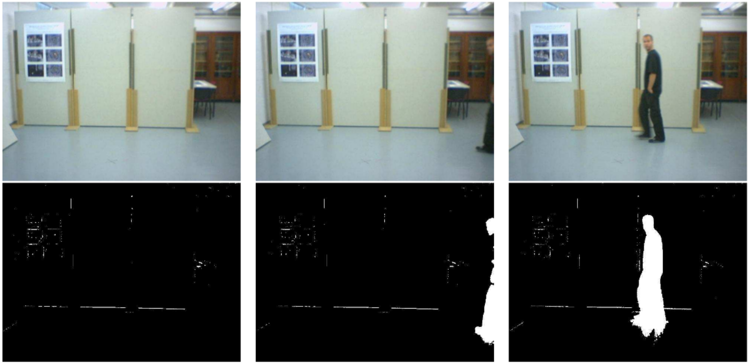
Figure : Segmentation by background subtraction.

Principles



- Major assumption: fixed camera

Example



- Any application with moving objects
- Video-surveillance

Elementary method

Naive approach (static background)

Foreground is detected, pixel by pixel, as the difference between the current frame and a static reference image (background):

$$|I_t - B| > \text{threshold} \tag{25}$$

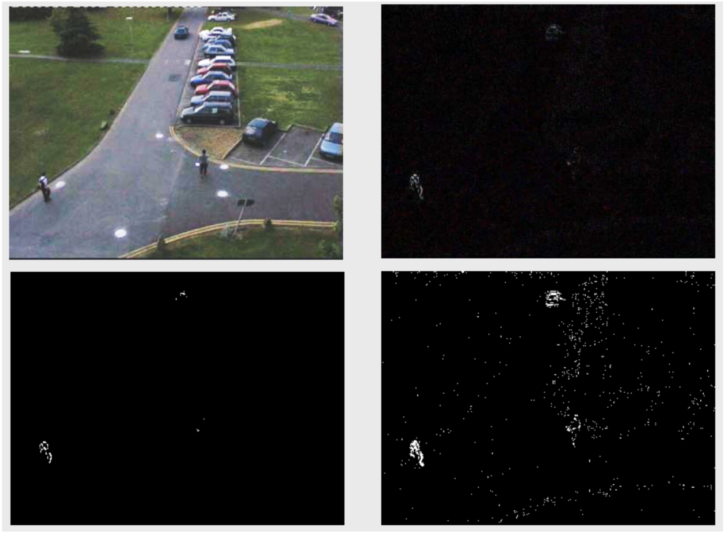
where

- I_t is the current pixel value (at time t),
- B is the reference background value for that pixel.

Problems:

- How do we choose the reference image?
- What's the best threshold?

Problem with the threshold



Classical processing chain

pre-processing of an input frame

$\Downarrow$

segmentation map after background processing

$\Downarrow$

post-processing of the segmentation map

Typical post-processing operations are:

- morphological filtering (cleaning): erosion, dilation, opening, area opening
- median filtering
- analysis of connected components
- shadow removal
- ...

Classical processing chain

pre-processing of an input frame

$\Downarrow$

segmentation map after background processing

$\Downarrow$

post-processing of the segmentation map

Typical post-processing operations are:

- morphological filtering (cleaning): erosion, dilation, opening, area opening
- median filtering
- analysis of connected components
- shadow removal
- ...

Choice for a reference image

Simple techniques

- One “good” image is chosen (usually a frame empty of foreground objects).
- Exponentially updated reference image

$$B_t = \alpha I_t + (1-\alpha) B_{t-1} \tag{26}$$

  Typical value for α: 0.05

- Median of the last N frames.

Important choice:

- *conservative update* (only when the pixel belongs to the background) or not (the pixel belongs to the foreground or the background).

Advanced techniques

The background is modelled as a probability density function to be estimated

- One gaussian distribution per pixel
- *Mixture of gaussians for each pixel*
- *Kernel based estimation of the probability* density function

One Gaussian per pixel

For each pixel, the probability density function of observed values is modeled by a **single** Gaussian.

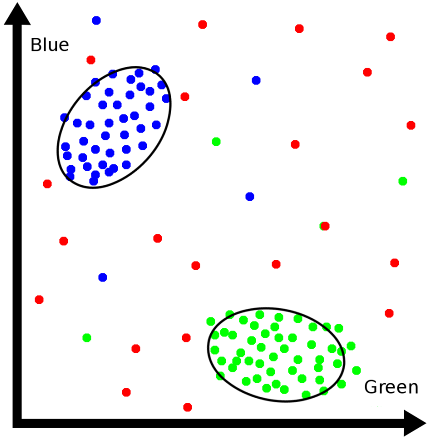
Once the model is built (here it means that we need to estimate the mean and variance), we evaluate the distance to the mean. If

$$|I_t - \mu| \leq \text{threshold} \times \sigma \tag{27}$$

then the pixel belongs to the background.

Mixture of Gaussians Model

Motivation: the probability density function of the background is multi-modal



[Stauffer, 1999, 2000] [Power, 2002] [Zivkovic, 2006]

Mixture of Gaussians Model

For each pixel:

$$P(X) = \sum_{i=1}^{N} \alpha_i N(\mu_i, \sigma_i) \tag{28}$$

Typical values for n: 3 or 5

Fundamental assumptions

- The background has a low variance.
- The background is more frequently visible than the foreground.

Kernel Density Estimation (KDE) methods

For each pixel:

$$P(X) = \sum_{i=1}^{N} \alpha_i K_\sigma(X - X_i) \tag{29}$$

where $\{X_i\}_{i=1,\ldots,N}$ are the N last values observed (samples) for that pixel, and $K_\sigma()$ is a kernel probability function centered at X_i.

Decision rule

The pixel belongs to the background if $P(X) > threshold$.

[Elgammal, 2000] [Zivkovic, 2006]

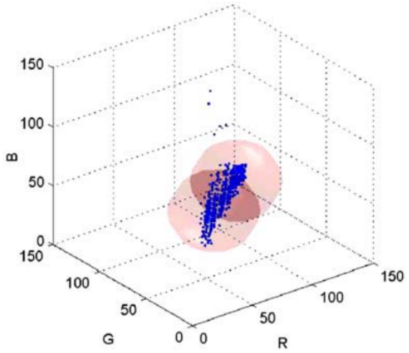
Parameters of KDE techniques

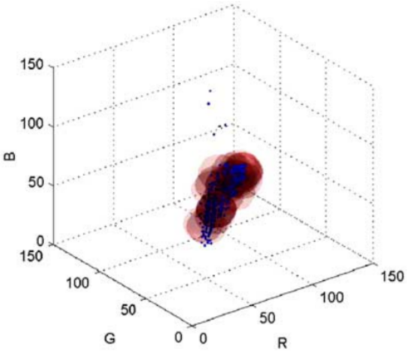
Typical values

$P(X) = \sum_{i=1}^{N} \alpha_i K_\sigma(X - X_i)$ with

- Number of samples: $N = 100$
- Weight: $\alpha_i = \alpha = \frac{1}{N}$
- Spreading factor: $\sigma = \text{Variance}(X_i)$
- Probability density function chosen to be gaussian: $K_\sigma(X - X_i) = N(X_i, \sigma^2)$

GMM techniques vs KDE techniques





Challenges

- Input related issues
  - lighting changes
    - slow (day/night cycles)
    - fast (light switch, clouds in the sky, ...)
  - unwanted motions
    - camera shaking (wind)
    - in the background (tree leaves, waving grass, water)
  - appearance changes of foreground objects (reflections), shadows, camouflage, ...
- Implementation related issues
  - Robustness
  - Real time

Outline

What's a border/contour/edge?

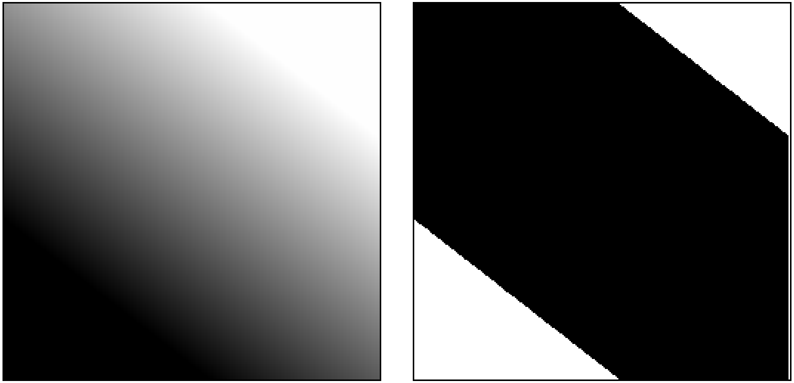


Figure : An image (diagonal ramp) and its contours (in black).

Can we locate edge points?

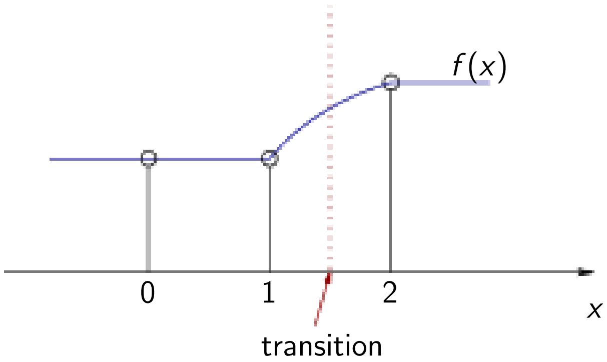


Figure : Problem encountered to locate an edge point.

Linear operators I

For derivate operators, we have to address two problems:

1. find the best approximate for the derivate
2. avoid an excessive amplification of the noise

These are two apparent contradictory requirements $\Rightarrow$ trade-offs

Linear operators II

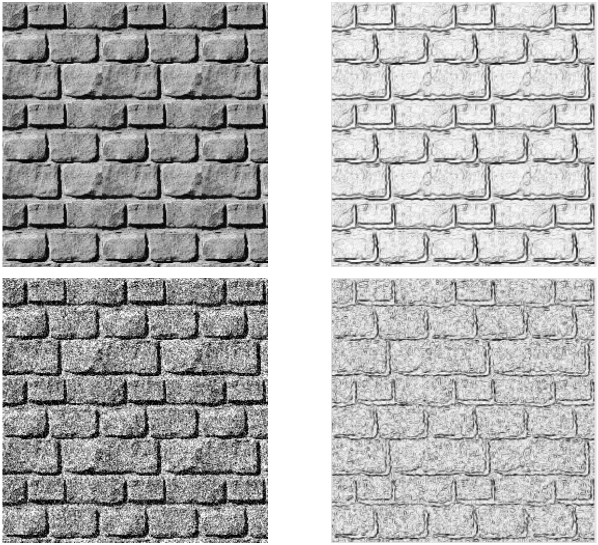


Figure : Images (left-hand side) and gradient images (right-hand side)

First derivate operator I

Let us consider the *partial* derivate of a function $f(x, y)$ with respect to x. Its Fourier transform is given by

$$\frac{\partial f}{\partial x}(x, y) \rightleftharpoons 2\pi j u \, \mathcal{F}(u, v) \tag{30}$$

In other words, deriving with respect to x consists of multiplying the Fourier transform of $f(x, y)$ by the following transfer function $\mathcal{H}_x(u, v) = 2\pi j u$, or of filtering $f(x, y)$ with the following impulse function:

$$h_x(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (2\pi j u) \, e^{2\pi j (xu + yv)} du dv \tag{31}$$

If we adopt a vectorial notation of the derivate, we define the *gradient* ∇f of image f by

$$\nabla f = \frac{\partial f}{\partial x} \overrightarrow{e_x} + \frac{\partial f}{\partial y} \overrightarrow{e_y} = (h_x \otimes f) \overrightarrow{e_x} + (h_y \otimes f) \overrightarrow{e_y} \tag{32}$$

First derivate operator II

Definition

[Gradient amplitude]

$$|\nabla f| = \sqrt{(h_x \otimes f)^2 + (h_y \otimes f)^2} \tag{33}$$

The amplitude of the gradient is sometimes approximated by

$$|\nabla f| \simeq |h_x \otimes f| + |h_y \otimes f| \tag{34}$$

which introduces a still acceptable error (in most cases) of 41%!

Definition

[Gradient orientation]

$$\varphi_{\nabla f} = tan^{-1}\left(\frac{h_y \otimes f}{h_x \otimes f}\right) \tag{35}$$

Second derivate operator

Definition

[Laplacian]

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = (h_{xx} \otimes f) + (h_{yy} \otimes f) \qquad (36)$$

As the first derivate, it can be shown that in the Fourier domain, the Laplacian consists to apply the following filter

$$\nabla^2 f \rightleftharpoons -4\pi^2(u^2 + v^2)\mathcal{F}(u, v) \qquad (37)$$

As can be seen, high frequencies tend to be amplified.

Sampling the gradient and residual error I

In order to derive practical expressions for the computation of a derivate, we adopt the following approach:

- develop some approximations and compute the resulting error,
- study the spectral behavior of these approximations, and
- discuss some practical approximations expressed in the terms of convolution masks.

Centered approximations?

An approximation of the first derivate is given by

$$f_a'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{38}$$

where h is the distance between two samples and index a denotes that it is an approximation. Please note that his approximation consists to filter $f(x)$ by the following convolution mask

$$\frac{1}{2h}\left[\begin{array}{ccc} -1 & 0 & 1 \end{array}\right] \tag{39}$$

Sampling the gradient and residual error II

For the second derivate, one possible approximation is

$$f_a''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{40}$$

Computation of the residual error. Let's consider the following Taylor extensions

$$\begin{aligned} f(x+h) &= f(x) + h\,f'(x) + \frac{h^2}{2} f''(x) + \ldots + \frac{h^n}{n!} f^{(n)}(x) + \ldots \quad (41) \\ f(x-h) &= f(x) - h\,f'(x) + \frac{h^2}{2} f''(x) + \ldots + (-1)^n \frac{h^n}{n!} f^{(n)}(x) + \ldots \quad (42) \end{aligned}$$

First derivate. By subtraction, member by member, these two equalities, one obtains

$$f(x+h) - f(x-h) = 2h\,f'(x) + \frac{2}{3!} h^3 f^{(3)}(x) + \ldots = 2h\,f'(x) + O(h^3) \tag{43}$$

Sampling the gradient and residual error III

After re-ordering,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (44)$$

Second derivatee. Like for the first derivate, we use the Taylor extension by add them this time (so we sum up (41) and (42)),

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + \frac{2}{4!} h^4 f^{(4)}(x) + \dots \quad (45)$$

As a result:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2) \quad (46)$$

The $f_a''(x)$ approximation is also of the second order in h.
Synthesis of expressions with a pre-defined error. Another approximation, of order $O(h^4)$, can be built. It corresponds to

$$f_a'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} \quad (47)$$

Spectral behavior of discrete gradient operators I

Consider the one-dimensional continuous function $f(x)$ and the following first derivate:

$$f'_a(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{48}$$

Its Fourier is given by

$$\frac{f(x+h) - f(x-h)}{2h} \rightleftharpoons \frac{e^{2\pi juh} - e^{-2\pi juh}}{2h} \mathcal{F}(u) \tag{49}$$

which can be rewritten as

$$\frac{f(x+h) - f(x-h)}{2h} \rightleftharpoons (2\pi ju) \frac{\sin(2\pi hu)}{2\pi hu} \mathcal{F}(u) \tag{50}$$

where the $(2\pi ju)$ factor corresponds to the ideal (continuous) expression of the first derivate.

Spectral behavior of discrete gradient operators II

Let us now consider the approximation of the second derivate

$$f''_a(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{51}$$

Its Fourier is given by

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \rightleftharpoons (-4\pi^2 u^2) \left(\frac{\sin(\pi h u)}{(\pi h u)} \right)^2 \mathcal{F}(u) \tag{52}$$

Spectral behavior of discrete gradient operators III

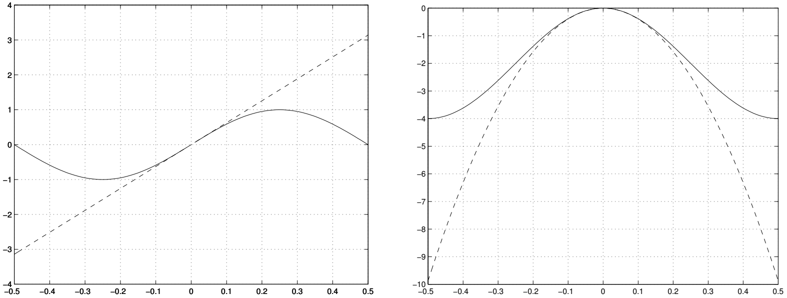


Figure : Spectral behavior of the derivate approximations (for $h = 1$). Left: first derivate, right: second derivate.

Practical expressions of gradient operators and convolution masks I

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \tag{53}$$

corresponds to the following non-centered approximation of the first derivate:

$$\frac{f(x+h,y)-f(x,y)}{h} \tag{54}$$

This "convolution mask" has an important drawback. Because it is not centered, the result is shifted by half a pixel. One usually prefers to use a centered (larger) convolution mask such as

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \tag{55}$$

Practical expressions of gradient operators and convolution masks II

In the y direction, this becomes

$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{56}$$

But then, it is also possible to use a diagonal derivate:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{57}$$

Practical expressions of gradient operators and convolution masks III

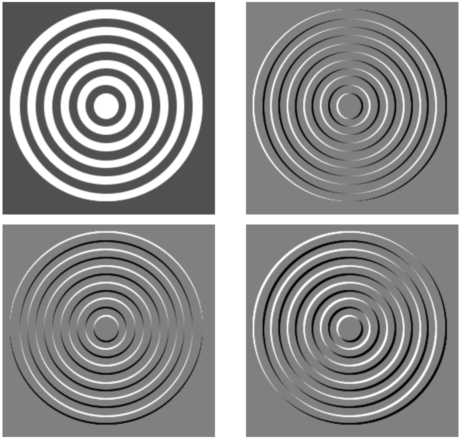


Figure : (a) original image, (b) after the application of a horizontal mask, (c) after the application of a vertical mask, and (d) mask oriented at 135^0.

Practical problems

The use of convolution masks has some drawbacks:

- **Border effects**. Solutions:
  - (i) put a *default* value *outside* the image;
  - (ii) *mirroring extension*: copy inside values starting from the border;
  - (iii) *periodization* of the image –pixels locate on the left are copied on the right of the image,
  - (iv) *copy* border values to fill an artificial added border.
- **The range (dynamic) of the possible values is modified.**
- **It might be needed to apply a normalization factor**.

Prewitt gradient filters

$$[h_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{58}$$

$$[h_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{59}$$

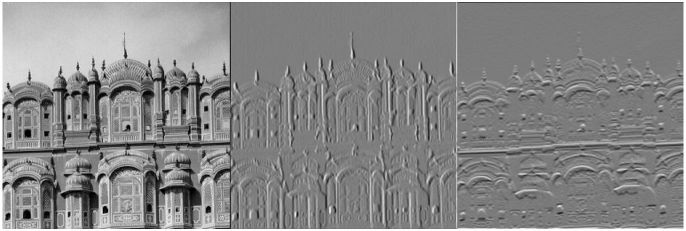


Figure : Original image, and images filtered with a horizontal and vertical Prewitt filter respectively.

Sobel gradient filters

$$[h_x] = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad (60)$$

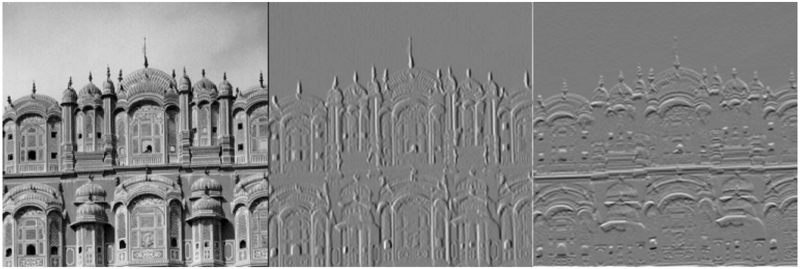


Figure : Original image, and images filtered with a horizontal and vertical Sobel filter respectively.

Second derivate: basic filter expressions

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

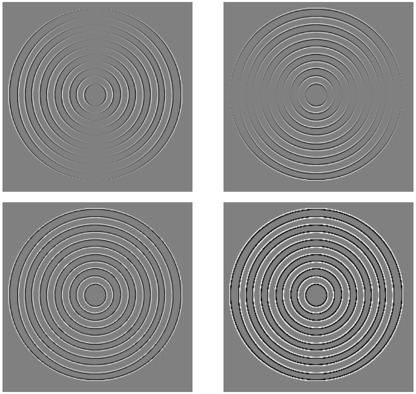


Figure : Results after filtering with the second derivate mask filters.

Détecteur de Harris I

Soit un pixel p dont les coordonnées sont (x, y). On calcule

$$M = \sigma_D^2 g(\sigma_I) \otimes \begin{bmatrix} I_x^2(p, \sigma_D) & I_x(p, \sigma_D) I_y(p, \sigma_D) \\ I_x(p, \sigma_D) I_y(p, \sigma_D) & I_y^2(p, \sigma_D) \end{bmatrix} \tag{61}$$

avec

$$I_x(p, \sigma_D) = \frac{\partial}{\partial x} g(\sigma_D) \otimes I(p) \tag{62}$$

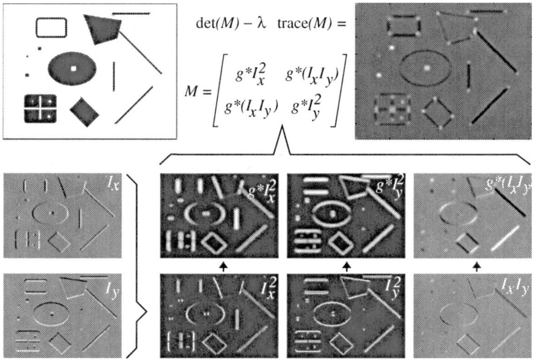
$$g(\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{63}$$

Definition

Cornerness is defined as

$$\mathbf{det}(M) - \lambda \mathbf{trace}(M) \tag{64}$$

Détecteur de Harris II



$\det(M) - \lambda\ \mathrm{trace}(M) =$
$M = \begin{bmatrix} g*I_x^2 & g*(I_xI_y) \\ g*(I_xI_y) & g*I_y^2 \end{bmatrix}$
I_x
I_y
$g*I_x^2$
$g*I_y^2$
$g*(I_xI_y)$
I_x^2
I_y^2
I_xI_y

Laplacian of Gaussians I

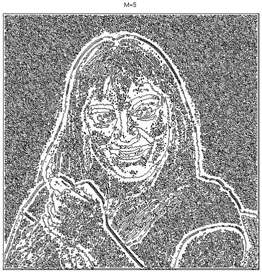
Principes:

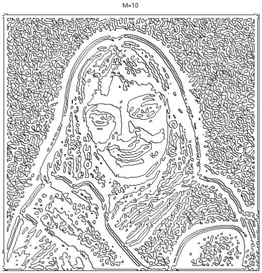
- application d'un filtre gaussien
- calcul du laplacien de l'image filtrée
- détection des points de passage par zéro

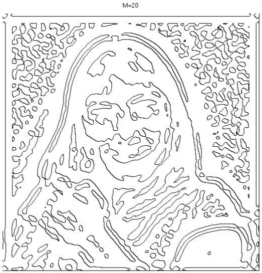
Paramètre:

- variance du filtre gaussien

Laplacian of Gaussians II







Bibliographie

- M. Petrou and C. Petrou, Image Processing: The Funamentals, Wiley, 2010.
- T. Tuytelaars, K. Mikolajczyk, *Local Invariant Feature Detectors: A Survey*, now, 2008.

Outline

Principles of template and feature matching

Elements:

- What do we search for in an image?
- Metric for comparison?
- Performance evaluation?
- Implementation in a real environment?

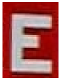
Template matching techniques

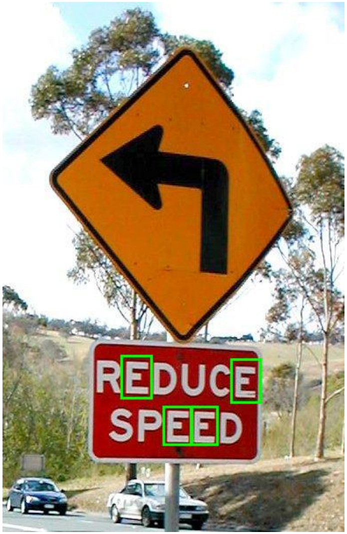
1. *template-based* matching (or *global* matching).
    - Use of an entire template.
    - Based on templates which are samples or examples. Their size conditions the computation time.
    - Templates can be expressed in terms of eigenspaces.
    - The metric is generally a sum-comparing metric (like the Sum of Absolute Differences [*SAD*] or cross-correlation).
2. *feature-based* matching.
    - Features are edges, corners, etc
    - Match-measures are metrics to find the best matching location
    - Features need to be “strong”.

Possible elements of template matching theory

- Template matching as testing
- Robust similarity estimators
- Matching measures
- Matching variable patterns
- Matching linear structure: the Hough transform
- Low-dimensionality representations and matching
- Deformable templates
- Computational aspects of template matching
- Matching point sets: the Hausdorff distance

Template-based matching



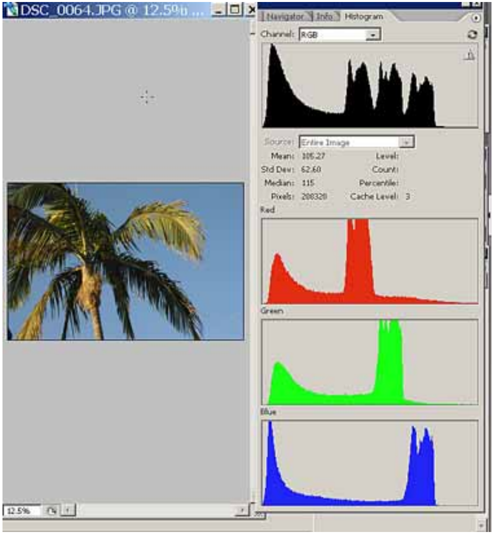


Feature-based matching

Steps:

- select a set of robust features in the reference frame (or sub-frame)
- compute the same set of features in the target image
- establish the relationship between the two sets of features. Many strategies are possible:
  - global optimization
  - local optimization
  - RANSAC algorithm

Histograms



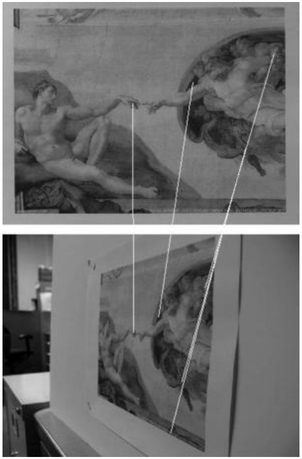
Channel: RGB

Source: Entire Image

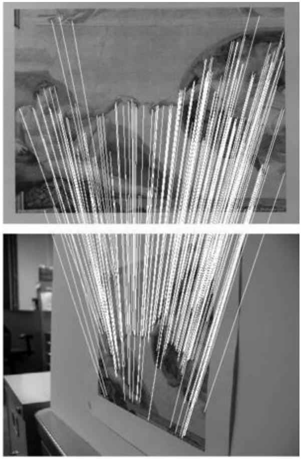
Mean: 105.27 Level:
Std Dev: 62.60 Count:
Median: 115 Percentile:
Pixels: 208320 Cache Level: 3

Feature detectors

MSER: 5 matches



ASIFT: 202 matches



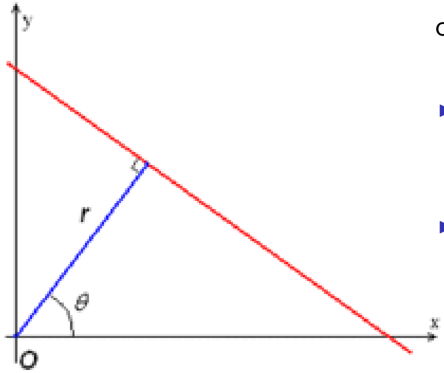
Detecting lines

Challenge: detect lines in an image

Towards the Hough transform

- Difficulty: matching a set of points arranged as a line
- Idea: instead of considering the family of points (x, y) that belong to a line $y = ax + b$, consider the two parameters
  - the slope parameter a (but a is unbounded for vertical lines)
  - the intercept parameter b

Definition of the Hough transform I



With the Hough transform, we consider the (r, θ) pair where

- the parameter r represents the distance between the line and the origin,
- while θ is the angle of the vector from the origin to this closest point

With this parametrization, the equation of the line becomes

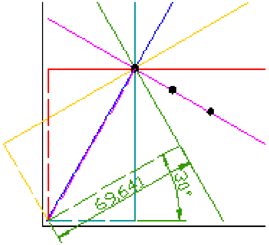
$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \tag{65}$$

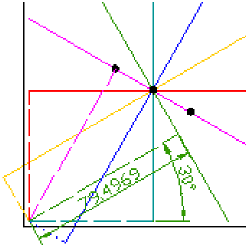
Definition of the Hough transform II

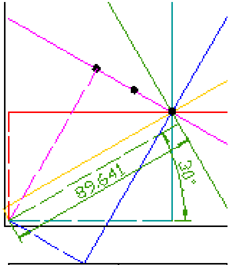
Each line is associated to a pair (r, θ) with $\theta \in [0, 2\pi[$ and $r \geq 0$. For an arbitrary point on the image plane with coordinates, e.g., (x_0, y_0), the lines that go through it are

$$r(\theta) = x_0 \cos\theta + y_0 \sin\theta$$

Example





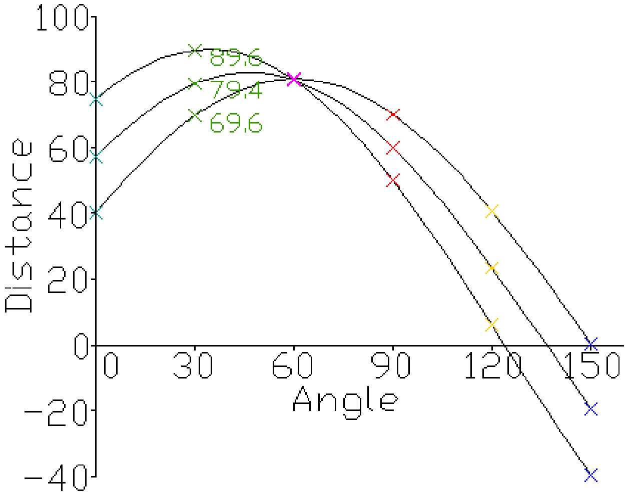


Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

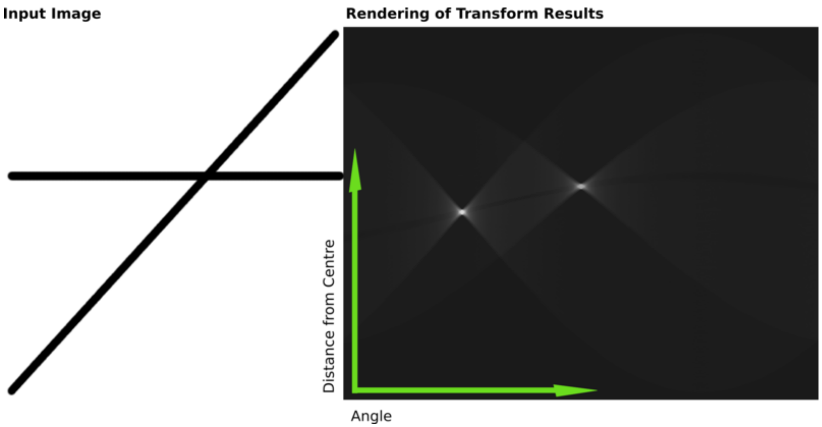
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6

Hough space



Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves.

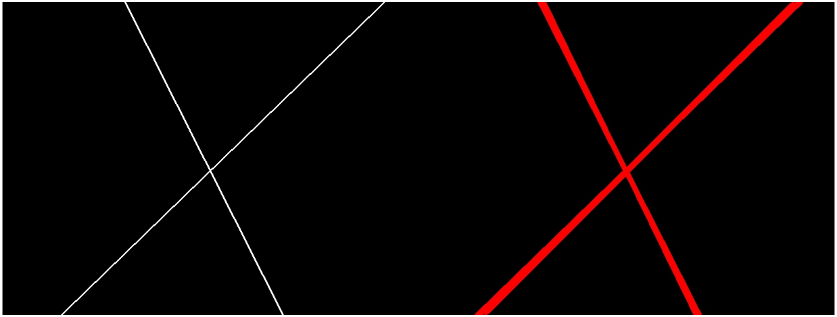
Hough space


Input Image
Rendering of Transform Results
Distance from Centre
Angle

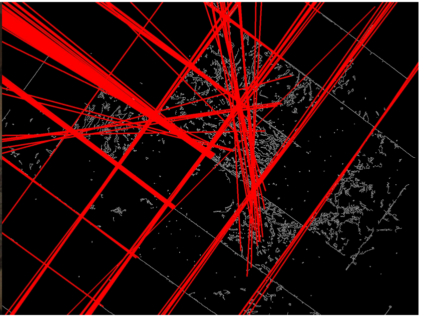
Algorithm for detecting lines

1. Detect edges in the original image.
2. Select “strong edges” for which there is enough evidence that they belong to lines.
3. For each point, accumulate values in the corresponding bins of the Hough space.
4. Threshold the accumulator function to select bins that correspond to lines in the original image.
5. Draw the corresponding lines in the original image.

“Toy” example



Real example



Bibliography

- R. Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice.* John Wiley & Sons, 2009.
- Wikipedia page on the Hough Transform.

Outline