Computer Vision

Diego Oniarti

Anno 2024-2025

Contents

1 24-02-2025 2

2 28-02-2025 3

3 Morphology 5

3.1 Dilation 5
3.2 Erosion 5
3.3 Closing and Opening 6

4 Models 6

4.1 Pinhole camera model 6
4.2 Orthographic projection model 7
4.3 Illumination models 7
4.3.1 Lambertian surface 8
4.4 Lenses 8
4.5 resolving power 8
4.6 Autofocus 9

5 Motion Detection 9

5.1 Motion Fields 9
5.2 Optical flow / image flow 10
5.3 Motion detection in practice 11
5.4 Gaussian average 12

6 Tracking 12

6.1 Region based 13
6.2 Lucas-Kanade optical flow 13
6.3 Bayesian tracking 13
6.4 Kalman filter 13

7 Camera Geometry and Multi-View 13

7.1 Affine Transformations 13

1 24-02-2025

Main topics of the course

1. acquisition
2. motion detection
3. motion analysis
4. stereo/multi -view
5. 3D point cloud
6. feature extraction / classification

Evaluation

The written exam will be 40% of the vote.
You can choose between a project and an oral exam. If you're not satisfied with the result of the written you can take an oral later.

Reading groups are also a thing.

- 24 mar: teams + project ideas
- 31 mar project titles assignment for those who haven't chosen one

For the project we'll use python, openCV, and ffmpeg.

You can deliver the project and written exam in different sessions. But the written exam expires in 1 year.

2 28-02-2025

Bayer Pattern

is a pattern used in camera sensors to optimize the distribution of colors. Since the human eye is more sensitive to green light, the pattern is composed of a checkerboard pattern where one color is green and the other is divided between red and blue.

Quantization

Usually we use 8bpp (bits per pixel) because it is byte aligned and because it's plenty enough for the human eye. At lower bpp, contouring appears.

Video

Static images loose the temporal and movement information, so we need videos. The frame rate of an image must be compliant with the thing that is being captured. With an high enough rate we can ensure a smooth transition between frames without loosing information.

This is the reason video-cameras usually have a lower resolution that photo-cameras. With too high of a resolution, there is too much information that needs to be processed and it can't be done at a fast enough rate.

Humans also focus less on image quality while watching a video, as they're more captivated by the evolution of events than the single frames.

Relevant features

The relevant features in an image are color, edges, and contrast. In a video the features are the same but also their progression through time.

Image compression

Image take up a lot of space and videos take up even more. Compression standards exist to reduce the amount of data required.
Compression requires there to be an **encoder** and a **decoder**. Some examples are JPEG, MPEG, and DIVX. Both visualization and processing are executed on the uncompressed image, since humans can't visualize raw compressed data, and filters can't work on the compressed image.

Some compression algorithms are lossy while some other are lossless.

Histogram

is a simple way to describe the color distribution of a picture by approximating a probability function.

$$hist(p) = \frac{\#pixels : I(x,y) = p}{N \cdot M} \approx f(p)$$

Where N, M are the size of the picture in pixels.

Various filters can be applied to an image by manipulating the histogram with operations like stretching and thresholding.

We can equalize an histogram defining a partial sum $CHist_I(p) = \sum_{k=0}^{p} hist(k)$ e assegnando $hist_{eq}(p) = \frac{CHist(p) - CHist_{min}}{M \cdot N - 1} \cdot 255$.
Even equalizing we can not get to a flat histogram, but we can do our best to get to that point.

Edge extraction

Usual Sobel su X, Y, thresholding, etc. Convolution in 1D and its natural translation in two dimensions.
A convolution in the space domain is equivalent to a product in the frequency domain and vice versa.

Low-pass filtering

The easiest way to implement a discrete low pass filter is to design a kernel that takes the average of the values surrounding a pixel.
A better visual result is given by a Gaussian filter. Funnily enough, the Fourier transform of a gaussian curve is still a gaussian curve.

Low Pass vs Median

Low pass filtering can reduce noise in an image, but it also spreads the noise over the image. In some cases this may be undesirable. The common approach would be to threshold the filtered image, but finding the threshold value can be cumbersome.

Some other filters to denoise is the **median filter**. It's not *isotropic* and it doesn't work with a normal convolution, but it requires a *sorting* operator.

Gaussian and averaging filters introduce in the image values that were not in the original image. The median filter, instead, only ”selects” values from the image, not inventing new ones.

3 Morphology

A form of non linear filtering that refers to the shape of a region.

Goals:

- check whether a certain shape fits into another
- check whether a picture has holes of a certain size
- remove areas smaller than a threshold

Binary morphology

We need a **binary image**[1] and **structuring elements** and implement four main operations:

- erosion
- dilation
- opening
- closing

Erosion and dilation are intuitive, enlarging or reducing the size of a region. Opening and closing are combinations of erosion and dilation in sequence.

Structuring elements can be squares, circles, other primitives, or custom shapes. For every structuring element we need to define a ”center”. It is usually the geometric center of the image but it doesn’t have to be.

3.1 Dilation

Dilation performs an $\oplus$ (or) operation between the image and the element. More specifically:

- sweep the element over the image
- if the origin of the element touches the image (a 1 in the image).
  - perform the or, ”stamping” the element onto the image

It is important to note that the output of the filter has to be stored in a separate image, to avoid it recursively dilating a pixel across the whole image.

3.2 Erosion

Erosion works in a similar way by scanning the element over the image: We don’t check with the center of the element anymore but we ”activate” the filter when every 1 in the filter overlaps a 1 in the image.

[1] A binary image is not grayscale but an image composed only of true and false

Question: In the output image, do we only put the center of the element or the whole element?

3.3 Closing and Opening

Closing: dilate and then erode
Opening: erode and then dilate.

Closing fills the holes in the image with the dilation, and then removes the excess added by the first operation with erosion.
Similar but inverse result is gotten by opening. The holes are enlarged, eating away at the shape. Then the remaining bits are consolidated.

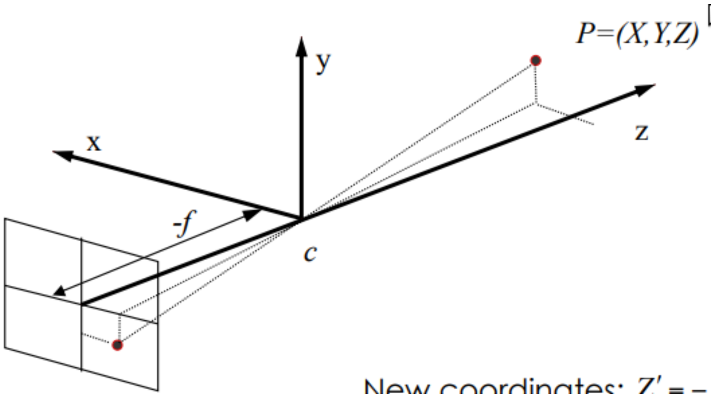
4 Models

4.1 Pinhole camera model

It consists of a box with a small hole on one of the walls. The light that passes through that hole projects a reflected image of the outside world onto the opposite wall.
One clear problem is that only a small amount of light can pass through the hole, making the projection very dim. We can fix this by making the hole bigger but this also makes the image blurry.

If the *image plane* is the plane opposite to the camera, the *virtual image plane* is an imaginary plane parallel to the image plane and equally spaced with the pinhole in the other direction.

Model to reality How do we map the pixel location to a location in space? Following the pinhole camera model we would also need the focal length f of the camera.
Then we know the point in space is somewhere on the line that passes through the pixel and the origin shifted by the focal length.



new coordinates From the camera

$$\begin{cases} Z' = -f \\ X' = -f\frac{X}{Z} \\ Y' = -f\frac{Y}{Z} \end{cases} \qquad (X,Y,Z) \to (x,y,f) = (f\tfrac{X}{Z}, f\tfrac{Y}{Z}, f)$$

It's easy to see that we loose some information, since a point in the camera plane is mapped to a whole line in the real world, loosing the distance.
We can approximate the distance with context clues, knowing additional information about the space etc. but these are not means of **measuring** the distance, only approximating it.

Multiple cameras To solve the problem of the loss of information we can use *two cameras*, or even more, to measure depth.

Properties of the pinhole model

- Parallel lines converge to a single vanishing point
- Parallel lines on the same plane lead to collinear vanishing points
- The line is called the horizon for a plane
- Vertical lines are perpendicular to the horizon

4.2 Orthographic projection model

The *orthographic projection model* assumes that all rays originated from the 3D object and from the scene are parallel among each other. The image plane is parallel to (X, Y)

Mathematically we're just taking the x and y of the point we're capturing, completely disregarding the depth component. This works since the model assumes object behave the same way regardless of distance.

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

4.3 Illumination models

Illumination is an important component in understanding the content of an image. Different lighting can change the colors perceived in the image, change the shape of perceived edges, etc.

Some materials and surfaces respond to light in different manners, depending how much they **absorb**, **reflect**, and **transmit** it.

Reflections can be

- Specular: more energy is concentrated in the light source direction

- Diffuse: constant in all directions

Surfaces vary in *specularity*, going from matte to glossy. Glossy materials are harder to work with, because they introduce things in the image that are not "real".

Illumination from one light source Problem: determine how the surface is irradiated by the light source assumption: light is far, we can assume all rays can be represented by a single unit vector s (ortho projection)

For each surface element the light is irradiated considering the cosine of the angle between the surface normal and the light direction

4.3.1 Lambertian surface

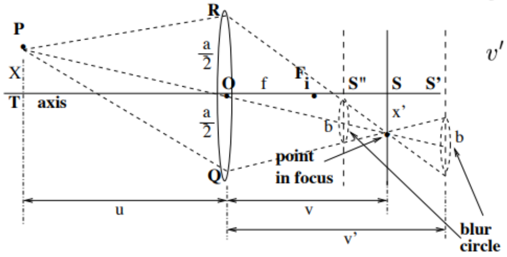
Model for diffuse reflection, so the specular reflections are ignored. It is possible to make this assumption when the surface is rough enough.

The luminance of a surface in this model is the same regardless of the viewing angle.

This model assumes every surface has a property ρ (*albedo*) that describes how much of the light is reflected by the object.

4.4 Lenses

The problem with the amount of light being allowed through a pinhole camera can be solved through the use of lenses.



$$v' = \frac{a+b}{a} v \quad : in \;\; case \;\; v' > v$$

$$v' = \frac{a-b}{a} v \quad : in \;\; case \;\; v' < v$$

$$u_n = \frac{u(a+b)}{a + \frac{bu}{f}}$$

$$u_r = \frac{u(a-b)}{a - \frac{bu}{f}}$$

.

4.5 resolving power

The **resolving power** is a property of a lens

$$R_p = \frac{1}{2\Delta}$$

Where Δ is the pixel spacing [in inches or millimeters].

Esercizio. Non stavo seguendo

4.6 Autofocus

There are *active*, *passive*, and fibred autofocus systems.

Active An infrared signal is sent and the time between sending and receiving is measured to compute distance. The distance is then used to focus.

This method has problems with obstacles and glossy/bright surfaces scattering and distorting the signal.

Passive Checks the contrast of the pixels for a row of pixels. If the contrast is the lens is adjusted until the image is sharp enough.

This method struggles with flat surfaces.

Cameras can switch between these two systems to achieve better results when one of them struggles.

5 Motion Detection

In video analysis and processing motion is a fundamental feature.

Projection of 3D motion onto the 2D image plane:

$$3D : D(X; t_1; t_2) = X' - X = [Dx, Dy, Dz]$$
$$2D : d(x; t_1; t_2) = x' - x = [dx, dy]$$

Once again, we cannot capture motion in 3D space on a 2D plane because of the dimensional mismatch. This results in the fact that the camera cannot perceive the motion of an object coming straight at it or moving straight away from it. Moreover, the points appear to be moving *outwards* as the object gets closer and the perspective makes it larger.

5.1 Motion Fields

Each frame is divided in 16x16 pixels macro blocks. Between a frame and the next, each block in frame a is compared with every other block in frame b and the best match is taken. The vector $a \to b$ is taken as the motion vector for that specific block.

This method is costly and makes a lot of assumptions:

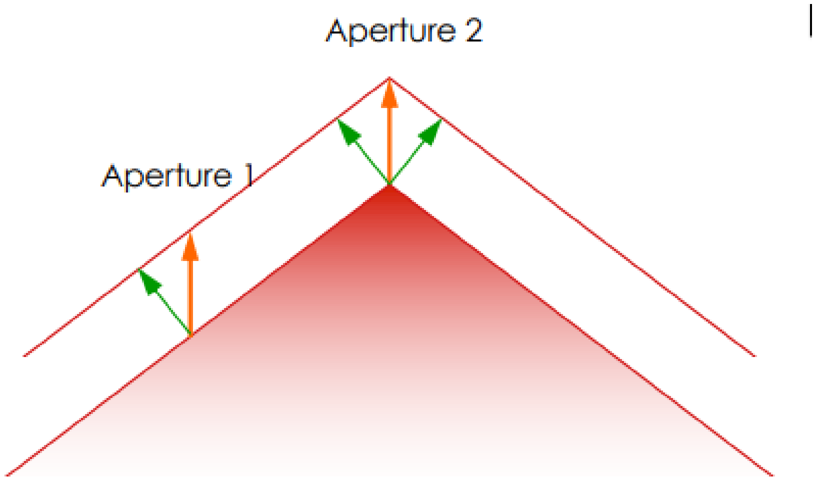
- static lighting

- rigid movement
- no occlusion
- etc...

on top of this it also struggles with flat surfaces.

Occlusion The surface is covered/revealed by the movement of an object in front of it.

Aperture The movement of an object can only be observed in the edges. Moreover, we can see only a component of the movement, since the one parallel to the edge is lost



5.2 Optical flow / image flow

To calculate the optical flow we make some assumptions:

- object illumination does not change in $[t, t+dt]$
- distances do not change significantly
- each point [x,y] is shifted in $[x+dx, y+dy]$

This is a translational model. It's not perfect but it's sufficient in a small enough time interval.

Optical Flow Equation Blah Blah Taylor expansion

$$\frac{\delta\psi}{\delta x}v_x + \frac{\delta\psi}{\delta y}v_x + \frac{\delta\psi}{\delta t}v_x = 0$$

$$or$$

$$\nabla\psi^T v + \frac{\delta\psi}{\delta t} = 0$$

This equation still only captures the projection of the motion along the perpendicular axis to the edge.

5.3 Motion detection in practice

Motion detectors can be *intensity based* or *feature based*, with the latter one taking into considerations corners (because of aperture) etc

Two problems with motion detection are:

- how to rappresenta the motion field
- what do choose as a discrimination parameter?

Change detection - image differencing The most trivial method to perform motion detection is to calculate the difference between each frame and the previous one.
This method only highlights edges of moving objects, and only the edges perpendicular to the motion direction

Background subtraction Similar to change detection. You take the first frame as the *background*. Then every subsequent frame is compared to that original background.

Of course the background image must be taken in a ”neutral” moment. An object that was captured in the background image and then moved will be shown in **every** frame, and it’s called a ”ghost”.

This model is **VERY** susceptible to scene change, lighting difference, camera motion, and weather condition.

Adaptive background subtraction Introduce a learning rate α to update the background as the video goes on

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

- $\alpha = 0$ background subtraction
- $\alpha = 1$ frame differencing
- $\alpha \in (0, 1)$ something in between. To be tuned for the specific application.

The background is constantly updated, so that objects that stop moving get absorbed into the background and gradually fade from the change detector.

5.4 Gaussian average

$$\mu_t = \alpha I_t + (1 - \alpha)\mu_{t-1}$$

Build a probability density function for the value of a pixel and update it through time. If the pixel value exceeds the mean by a certain amount count it as movement.

We put a pixel "in the foreground" if $|I_t - \mu_t| > k\sigma_t$ where k is a constant (chosen by heart)

This technique requires to "train" the model at the start, giving it some frames to calculate the gaussiana and adjust their initial values.

Improved Gaussian average The technique can be improved with a binary M that decides if something will be added to the background or not $\mu_t = M\mu_t + (1-M)(\alpha I_t + (1-\alpha)\mu_{t-1})$. Even then this method is quite finicky.

Mixture of Gaussian distributions To make the method more robust we can use k Gaussian distributions instead of 1.

$$P(x_t) = \sum_{i=0}^{k} \omega_{i,t}\eta(x_t, \mu_{i,t}, \Sigma_{i,t})$$

This is also called "*multimodal*" Gaussian average, since it takes into account multiple "models" of motions that may affect the image.
Take the example of movement that may be induced by *rain*, *wind*, and *clouds*. Each one will induce some fake movement in the image, which can be accounted by a mixture of gaussiana.

We then rank the Gaussians on their weight (decreasing) and select a threshold T. We take the first B Gaussians until the sum of their weights reaches T.
The first B Gaussians model the background, while the remaining ones model the foreground.

6 Tracking

Identify moving objects and track their motion through time.

We're tracking the path of moving objects on a 2D image gotten from the 3D world. We either get a projection of the motion, or we use stereo-cameras/lidar/other methods to capture a 3D path.

6.1 Region based

Tracking regions. It is good for real-time applications.

> **wow.** By encoding an image in HSB colorspace we can ignore the B component and automatically filter out the shadows

- Get regions
- Calculate the histogram for each region
- Use the histogram to ID the regions and track them through video

This method can be tricked by changes in illumination or color of the moving objects

To make this method more robust we can further subdivide objects into smaller regions (like subdividing people in head-legs-torso). Each region has its own histogram and color vector

6.2 Lucas-Kanade optical flow

ρ-bust

6.3 Bayesian tracking

6.4 Kalman filter

7 Camera Geometry and Multi-View

7.1 Affine Transformations

An transformation is called *affine* if it can be described as the product between a matrix and the original coordinates.
The three main affine transformations are:

- Scaling
- Rotation
- Translation

But we can combine multiple of these in a single matrix.

Scaling

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\theta) & -y\sin(\theta) \\ x\sin(\theta) & y\cos(\theta) \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ 1 \end{bmatrix}$$

For the translation we introduce one more dimension to the array, and this is called *"homogeneous coordinates"*.

Putting these all together we can define a transformation matrix with 4 parameters c, θ, x_0, y_0

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & x_0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

We ended up with 6 parameters to determine and we can find them using 3 matching pairs.
What is usually done is putting a known pattern in some part of a video or a camera setup. The pattern is then used to calibrate the camera matrix for the entire image.

This technique can be effected by errors, as any imprecision on the capture of the known pattern is then reflected on the rest of the image.
It is hence better to take a large amount of samples and use the least-square method to minimize the error deviation.