CIS 580 Spring 2012 - Lecture 1

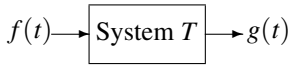
January 11, 2012

Notes and figures by Matthieu Lecce.

Linear Shift-Invariant Systems

Consider a continuous signal $f : \mathbb{R} \to \mathbb{R},\ t \to f(t)$ and a filter $f(t) \to g(t) = T\{f(t)\}$:

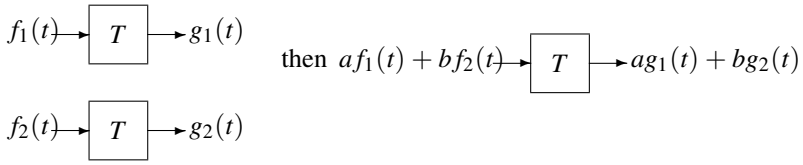
Figure 1: A filter represented as a block



Linear system

A system T is **linear** when $T\{af_1(t) + bf_2(t)\} = aT\{f_1(t)\} + bT\{f_2(t)\}$:

Figure 2: Linear System



Examples:

- $T\{f\}(t) = f(t) - f(t-1)$ is linear:

$$\begin{aligned} g_1(t) &= f_1(t) - f_1(t-1) \\ g_2(t) &= f_2(t) - f_2(t-1) \\ T\{af_1 + bf_2\}(t) &= [af_1(t) + bf_2(t)] - [af_1(t-1) - bf_2(t-1)] \\ &= a(f_1(t) - f_1(t-1)) + b(f_2(t) - f_2(t-1)) \\ &= aT\{f_1\}(t) + bT\{f_2\}(t) \end{aligned}$$

- $g(t) = f(t)^2$, $g(t) = \max(f(t), f(t-1))$ are not linear.

Shift invariant system

A system is **shift-invariant** when $T\{f(t-t_0)\}(t) = T\{f(t)\}(t-t_0)$:

Figure 3: Shift-Invariant System

Examples:

- $T\{f\}(t) = f(t) - f(t-1)$ is shift-invariant.

$$\begin{aligned} T\{f(t-t_0)\} =& f(t-t_0) - f(t-t_0-1) \\ =& g(t-t_0) \end{aligned}$$

- $g(t) = tf(t)$ is not shift-invariant.

Dirac function and impulse response

Definitions of the Dirac function:

1. $\delta(t) = \lim_{a\to 0} \frac{1}{a}\text{rect}(\frac{t}{a})$,

$$\text{rect}(t) = \begin{cases} 1 & |t| \le 1/2 \\ 0 & \text{elsewhere} \end{cases}$$

2. Absorption property $\int_{-\infty}^{\infty} f(t)\delta(t)dt = f(0)$
3. $\delta(t) = 0$ for $t \neq 0$ and $\int_{-\infty}^{\infty} \delta(t)dt = 1$

The *impulse response* of a system T is the output of the system when the input is a Dirac:

$$h(t) = \int_{-\infty}^{\infty} \delta(t')h(t-t')dt',$$

holds because of the absorption property $t \to t - t'$.

LSI as a convolution

Question: is there a formula describing the action of a general LSI system?

Answer: Yes, it is the convolution of the signal with the *impulse response* $h(t)$:

$$g(t) = \int_{-\infty}^{\infty} f(t')h(t-t')dt'$$

.

Notice that $h(t-t')$ is a reflection and shift of the impulse response (see figure 4). To understand why we take the reflection of the impulse response, we consider a special type of LSI's: **causal** systems, where $g(t)$ only depends on values of $f(t')$ for $t' \le t$. This means the impulse response will be non-zero for $t \ge 0$.

For example let's consider $T : f(t) \to T\{f(t)\} = g(t) = f(t) + f(t-1) + f(t-2)$. The impulse response is $h(t) = \delta(t) + \delta(t-1) + \delta(t-2)$: notice that the non-zero values of $h(t)$ correspond to positive t's. When we apply the convolution sum, we will need to be careful to take the reflection of h, so that $g(t)$ depends indeed on values of f for $t' < t$.

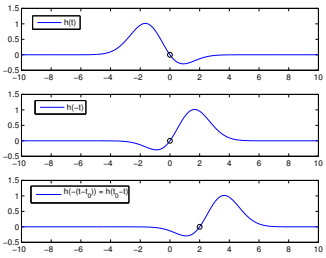


Figure 4: Reflection and shift of the impulse response when computing the convolution.

Fourier Transform

Quick reminder on complex numbers:

- $a + jb$, $j^2 = -1$
- Harmonic exponentials $e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)$.

$$f(t) \rightarrow F(\omega) = \mathcal{F}\{f(t)\}$$
$$F : \mathbb{R} \rightarrow \mathbb{C}$$
$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt,$$

where ω denotes the frequency.

Inverse Fourier transform:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$$

CIS 580 Spring 2012 - Lecture 2

January 23, 2012

Notes and figures by Matthieu Lecce.

Last lecture's main result: linear shift-invariant (LSI) systems can be represented as a convolution.

The Fourier Transform

Definition of the Fourier Transform:

$$f(t) \circ\!\!-\!\!\bullet F(\omega) = \mathcal{F}\{f(t)\}$$
$$F : \mathbb{R} \to \mathbb{C}$$
$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt,$$

where ω denotes the frequency. This definition is sometimes called non-unitary Fourier transform, with angular frequency (ω is referred to as angular frequency, and s such that $\omega = 2\pi s$ is the ordinary frequency).

Inverse Fourier transform:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega$$

Domains:

- $f(t)$ defined in time (or space for x, y) domain
- $F(\omega)$ defined in frequency (or spatial frequency) domain

The Fourier transform can be defined as a function of s, the frequency, where $\omega = 2\pi s$, in which case the definitions can be rewritten as follows:

$$F(s) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi st} dt$$
$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(s) e^{j2\pi st} ds$$

Quick reminder on complex numbers:

- $a + jb \in \mathbb{C}$, $j^2 = -1$
- $e^{j\omega t} = \cos(\omega t) + j \sin(\omega t)$.

Function symmetry and Fourier

Definitions:

- A function f_e is said *even* when $f_e(-t) = f_e(t)$
- A function f_o is said *odd* when $f_o(-t) = -f_o(t)$

Any function $f(t)$ can be decomposed into an odd and an even part:

$$f(t) = f_e(t) + f_o(t)$$
$$\text{where } f_e(t) = \frac{1}{2}(f(t) + f(-t)) \text{ is even}$$
$$f_o(t) = \frac{1}{2}(f(t) - f(-t)) \text{ is odd}$$

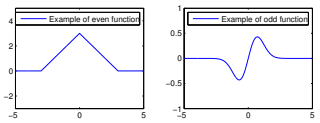


Figure 1: Even and odd functions.

If we apply the Fourier transform to this decomposition, we obtain the following:

$$\begin{aligned}&\int_{-\infty}^{\infty}(f_e(t)+f_o(t))(\cos\omega t - jsin\omega t)dt\\ =&\int_{-\infty}^{\infty}f_e(t)\cos(\omega t)dt - j\int_{-\infty}^{\infty}(f_o\sin\omega t)dt\end{aligned}$$

The even part maps to the real part of the Fourier transform and the odd part to the imaginary part (and vice versa).

Observe that $\int_{-\infty}^{\infty} g_e(t)g_o(t)dt = 0$

Theorems

Shift theorem

$$f(t-t_0) \circ\!\!-\!\!\bullet\; F(\omega)e^{-j\omega t_0}$$

Example: for $f(t)$ even, $f(t-\frac{T}{2}) \circ\!\!-\!\!\bullet\; F(\omega)e^{-j\omega\frac{T}{2}}$, where $F(\omega)$ is real.

Modulation theorem

$$f(t)e^{j\omega_0 t} \circ\!\!-\!\!\bullet\; F(\omega-\omega_0)$$

Multiplying by a complex exponential causes a shift in the frequency domain.

Similarity theorem

$$f(at) \circ\!\!-\!\!\bullet\; \frac{1}{|a|}F(\frac{\omega}{a})$$

Convolution This theorem is extensively used in image processing:

$$\begin{array}{ccccc} f(t) & \rightarrow & \boxed{h(t)} & \rightarrow & g(t)=\int_{-\infty}^{\infty} g(t')h(t-t')dt' = f(t)*h(t) \\ \circ\!\!-\!\!\bullet & & \circ\!\!-\!\!\bullet & & \circ\!\!-\!\!\bullet \\ F(\omega) & & H(\omega) & & G(\omega)? \end{array}$$

What happens in the Fourier domain?

$$\begin{aligned}G(\omega) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(t')h(t-t')dt'e^{-j\omega t}dt\\ &= \int_{t'=-\infty}^{\infty} f(t')\left\{\int_{t=-\infty}^{\infty} h(t-t')e^{-j\omega t}dt\right\}dt'\\ &= \int_{t'=-\infty}^{\infty} f(t')H(\omega)e^{-j\omega t'}dt' \text{ (shift theorem)}\\ &= H(\omega)F(\omega)\end{aligned}$$

$H(\omega)$ is call the transition function (as opposed to impulse response).

Inverse convolution

$$f(t)h(t) \circ\!\!-\!\!\bullet\; \frac{1}{2\pi}F(\omega)*H(\omega)$$

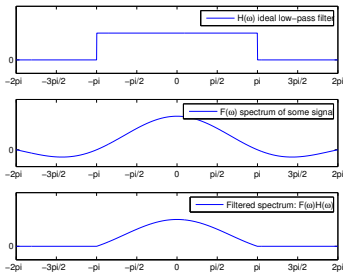


Figure 2: Low pass filtering by taking the product $H(\omega)F(\omega)$.

Fourier of some interesting functions

1. Recall the absorption property

$$\int_{-\infty}^{\infty} \delta(t-t_0) f(t) dt = f(t_0)$$

Then we have:

$$\int_{-\infty}^{\infty} \delta(t) e^{-j\omega t} dt = 1$$

We will remember the two following results:

$$\begin{aligned} \delta(t) &\multimap 1 \\ 1 &\multimap 2\pi\delta(t) \quad \text{(DC-component)} \end{aligned}$$

2. Fourier of a harmonic exponential $e^{j\omega_0 t}$:

$$\begin{aligned} e^{j\omega_0 t} &\underbrace{\multimap}_{\text{modulation}} 2\pi\delta(\omega-\omega_0) \\ \cos(\omega_0 t) &\multimap 2\pi.\tfrac{1}{2}(\delta(\omega-\omega_0)+\delta(\omega+\omega_0)) \\ \sin(\omega_0 t) &\multimap 2\pi.\tfrac{1}{2j}(\delta(\omega-\omega_0)-\delta(\omega+\omega_0)) \end{aligned}$$

Note: using the similarity theorem, we can derive simpler expressions using the ordinary frequency s instead of ω, for example:

$$\cos(2\pi s_0 t) \multimap \frac{1}{2}\delta(s+s_0)+\delta(s-s_0)$$

3. Fourier of the comb function $\text{Ш}(t) = \sum_{n=-\infty}^{\infty} \delta(t-nT)$:

$$\sum_{n=-\infty}^{\infty} \delta(t-nT) \multimap \frac{1}{|T|} \sum_{n=-\infty}^{\infty} \delta(s-\frac{n}{T})$$

(or $\frac{2\pi}{|T|}\sum_{n=-\infty}^{\infty}\delta(\omega-\frac{2\pi n}{T})$ when using the non-unitary Fourier Transform with angular frequency).

4. Fourier of a 1D Gaussian The 1D Gaussian distribution is defined as follows:

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}$$

When trying to integrate an exponential that contains the variable to the power 2, we will always try to boil it down to the famous *Gaussian integral*:

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

Here is a loose proof of the Gaussian integral:

$$\begin{aligned} I &= \int_{-\infty}^{\infty} e^{-x^2} dx \\ I^2 &= \int_{-\infty}^{\infty} e^{-x^2} dx \int_{-\infty}^{\infty} e^{-y^2} dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-(x^2+y^2)} dxdy \end{aligned}$$

We perform a substitution to use polar coordinates, the integral I^2 now takes the following form:

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \\ dxdy &= rdrd\theta \end{aligned}$$

$$\begin{aligned} I^2 &= \int_{r=0}^{\infty} \int_{\theta=0}^{2\pi} e^{-r^2} r dr d\theta \\ &= 2\pi \int_{r=0}^{\infty} r e^{-r^2} dr \\ &= 2\pi \left(\frac{1}{-2}\right) \left[e^{-r^2}\right]_0^{\infty} \quad ((e^{-r^2})' = -2re^{-r^2}) \\ &= \pi \end{aligned}$$

NOW LET'S COMPUTE the Fourier transform of a Gaussian distribution:

$$\begin{aligned} \mathcal{F}\{f(t)\} &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2}} e^{-2\pi st} dt \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2} - 2\pi s \frac{t}{\sigma\sqrt{2}} \sigma\sqrt{2} - \pi^2 s^2 \delta^2 2 + \pi^2 s^2 2} dt \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-(t + j\pi\sigma s\sqrt{2})^2} \\ &= \frac{1}{\sigma\sqrt{2\pi}} e^{-2\pi^2\sigma^2 s^2} \sqrt{2}\pi \end{aligned}$$

Therefore the Fourier Transform of a Gaussian is a Gaussian in terms of s!

Sampling

Definition and problem

Sampling is a multiplication of the signal by the comb function $\text{Ш}_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$. T is the sampling interval:

$$f_S(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

(S for "sampled") After sampling, we forget about the T: we just obtain a sequence of numbers, that we note $f_s[k]$.

Examples:

(a) Video of a rotating wheel. Depending on T, no motion is perceived, or even worse, a backward motion is perceived. (We will cover this example in HW2)

(b) Sinusoidal signal: appearant frequency of sampled signal is different

(c) Same for a checker pattern (Sequence of step functions)

Problem: we want to find the lowest possible sampling frequency, such that the sampled signal is not corrupted. **Solution**: Let's use the frequency domain to analyze the action of the comb (sampling) function.

Sampling in the frequency domain

Let's compute the Fourier transform of the comb function:

$$\sum_{n=-\infty}^{\infty} \delta(t-nT) \multimap\bullet \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T}) = \sum_{n=-\infty}^{\infty} \delta(s - \frac{n}{T})$$

Remember we defined the frequency s where $\omega = 2\pi s$

Sampling in the time domain is a **multiplication** with the comb function $\text{Ш}(t)$, therefore in the frequency domain it is a **convolution** with the Fourier of the comb, which is a sum of impulses $\delta(\omega - \omega_0)$.

The convolution with one impulse $\delta(\omega - \omega_0)$ corresponds to shifting the spectrum such that it is centered around w_0 instead of 0.

While in the time domain sampling is very simple, in the Fourier domain it is a complete mess: it is equivalent to "xeroxing" the signal (making several shifted copies of it) in the frequency domain:

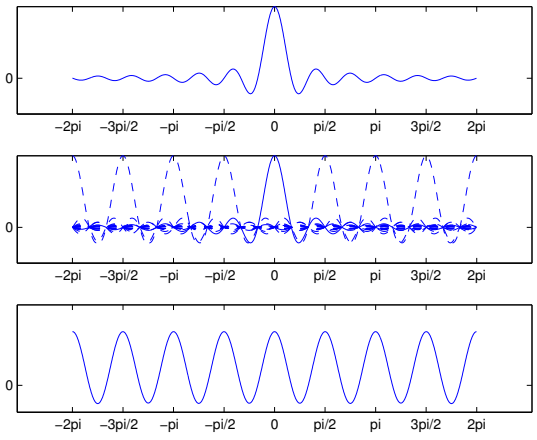


Figure 3: Sampling in the time domain corresponds to xerozing in the Fourier domain. Plot 1: spectrum of a rectangle (sinc function). Plot 2: spectrum and replicas after convolving with impulses. Plot3: result of convolution with comb function (sum of spectrum and replicas).

Fourier transform of a discrete signal To recover the signal, we need to be able to *isolate* the spectrum from its replicas: generally this is done by applying a low-pass filter $\text{rect}(T_s)$(like in figure 2). Even if the signal is

band-limited, i.e. with maximum frequency $\omega_{\max}$, we need to have:

$$\underbrace{\omega_{\text{sampling}}}_{=\frac{2\pi}{T_s}} \geq 2\omega_{\max}, \quad \text{i.e.} \quad \omega_{\max} \leq \frac{\pi}{T_s}$$

This result is known as the **sampling theorem**.

CIS 580 Spring 2012 - Lecture 3

January 25, 2012

Notes and figures by Matthieu Lecce.

Review from last lecture:

- Sampling (or time-sampling) is a multiplication with the comb function $\text{Ш}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$ (infinite trail of Dirac functions).
- It corresponds to a *convolution* with $\sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T})$ in the frequency domain.
- Concretely this means that in the frequency domain, the signal is *replicated* ("xeroxed") at frequencies $\frac{2\pi n}{T},\ n \in \mathbb{Z}$
- **Question**: Can we recover the original signal? In other words, can we isolate the original Fourier of $f(t)$ for this convolution (replication).
- **Answer**: Yes, *if* $2\omega_{\max} \leq \omega_{\text{sampling}} = \frac{2\pi}{T}$, in which case we multiply the Fourier of the sampled function with a rectangle function (low-pass filter).

Fourier and sampling

Reconstructing a sampled signal

$$F_S(\omega) = F(\omega) * \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T})$$

The rectangle function $\Pi(t)$ is defined as follows:

$$\Pi(t) = \begin{cases} \frac{1}{2\pi} & -\pi \leq \omega \leq \pi \\ 0 & \text{anywhere else} \end{cases}$$

What signal do we recover by multiplying with $\Pi(\omega)$?

$$f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) * \mathcal{F}^{-1}(\Pi(\omega))$$

Let's compute the Fourier transform of a box (rect) filter:

$$\Pi(t) = \text{rect}(t) \begin{cases} 1 & |t| \leq \frac{1}{2} \\ 0 & \text{anywhere else} \end{cases}$$

Similarly, the inverse transform is the following (this is the one we need to understand the effect of low-pass filtering)

$$\begin{aligned} \mathcal{F}\{\text{rect}(t)\} &= \int_{-1/2}^{1/2} 1 e^{-j\omega t} dt \\ &= \frac{1}{-j\omega} \left[e^{-j\omega t} \right]_{-1/2}^{1/2} \\ &= \frac{1}{-j\omega} \left[e^{-j\omega/2} - e^{j\omega/2} \right] \\ &= \frac{1}{-j\omega} \left(-2j \sin \frac{\omega}{2} \right) \\ &= \frac{\sin \frac{\omega}{2}}{\frac{\omega}{2}} = \text{sinc}\left(\frac{\omega}{2} \right) \end{aligned}$$

$$\begin{aligned}\mathcal{F}^{-1}\{\text{rect}(\omega)\} &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\text{rect}(\omega)e^{j\omega t}dt \\ &= \frac{1}{2\pi}\int_{-1/2}^{1/2}e^{j\omega t}dt \\ &= \frac{1}{2\pi}\frac{1}{j\omega}\left[e^{j\omega t}\right]_{-1/2}^{1/2} \\ &= \frac{1}{2\pi}\text{sinc}\left(\frac{t}{2}\right)\end{aligned}$$

We will remember the following definitions and results:

$$\frac{1}{2\pi}\text{sinc}(t/2) \circ\!\!-\!\!\bullet \text{rect}(\omega)$$
$$2\pi\text{sinc}(\pi t) \circ\!\!-\!\!\bullet \frac{1}{2\pi}\text{rect}\left(\frac{\omega}{2\pi}\right)$$

Definitions:

$$\Pi(\omega) = \begin{cases}\frac{1}{2\pi} & |\omega| \le \pi \\ 0 & \text{anywhere else}\end{cases}$$

$$\text{rect}(\omega) = \begin{cases}\frac{1}{2\pi} & |\omega| \le \frac{1}{2} \\ 0 & \text{anywhere else}\end{cases}$$

Remember the scaling theorem:

$$f(\alpha t) \circ\!\!-\!\!\bullet \frac{1}{|\alpha|}F\left(\frac{\omega}{t}\right)$$

Reconstructed signal:

$$\begin{aligned}f_{\text{reconstr}}(t) &= f(t)\sum_{n=-\infty}^{\infty}\delta(t-nT) * 2\pi\text{sinc}(\pi t) \\ &= \sum_{n=-\infty}^{\infty} f[n]2\pi\text{sinc}(\pi t)\end{aligned}$$

Discrete Fourier Transform

Definition of the Discrete Fourier Transform, for a discrete signal $f[n]$:

$$f[n] \circ\!\!-\!\!\bullet \sum_{n=0}^{L-1} f[n]e^{-j\frac{2\pi k}{L}n} = F[k]$$

Note the implicit definition $\omega = \frac{2\pi k}{L}$

with finite length: $n = 0 \ldots L-1$

Important: A discrete signal still has a continuous Fourier transform! The Discrete Fourier Transform corresponds to a *sampling in the (continuous) frequency domain*.

$$f[n] \circ\!\!-\!\!\bullet \sum_{n=0}^{L-1} f[n]e^{-j\omega n}$$

What does sampling in the frequency domain (as in figure 1) correspond to in the time/space domain? It corresponds to a **convolution**:

- FREQUENCY DOMAIN: multiplication with $\sum\delta(\omega - \frac{2\pi k}{L})$
- TIME DOMAIN: convolution with with $\sum\delta[n-kL]$, equivalent to **replication of the signal at multiples of its length**.

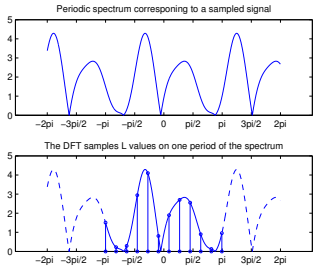


Figure 1: The DFT samples L values of one period of the spectrum by multiplying by the comb $\sum\delta\left(\omega - \frac{2\pi k}{L}\right)$

Fourier and derivation

Exact derivative in the Fourier domain

We are interested in the Fourier of the *derivative* of a function:

$$f(t) \circ\!\!-\!\!\bullet F(\omega)$$
$$\frac{df(t)}{dt} \circ\!\!-\!\!\bullet ?$$

Bad idea:

$$\int_{-\infty}^{\infty} \frac{df}{dt} e^{-j\omega t} dt$$

Smart idea:

$$\begin{aligned} f(t) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega t} d\omega \\ \frac{df(t)}{dt} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) \boxed{j\omega} e^{j\omega t} \\ &= j\omega \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{j\omega t} d\omega \\ &= j\omega F(\omega) \end{aligned}$$

Therefore, when taking a **DFT** (spectrum is periodic), the derivation is equivalent to multiplying the spectrum by the periodic function represented by figure 2:

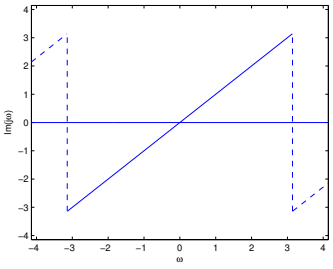


Figure 2: Derivating is equivalent to multiplying the spectrum by $j\omega$

Approximating the derivation with a filter

In many applications we need the derivative of the signal with respect to x or t. The goal of this section is to approximate the function in figure 2 (multiplication by $j\omega$ in the frequency domain) with an LSI filter (we will call it a derivative filter). More specifically, we want to find a discrete filter of impulse response $h[k],\ k = 1...K$ (K small) such that:

1. The DFT of h is close to $j\omega$
2. We "dump" high frequencies (DFT also corresponds to box sampling in frequency domain, so no approximation is needed for high frequencies)

Then, when we are given a discrete signal $f[n]$ represented in figure 3, all we have to do is to apply the filter to obtain $g[n]$, an approximation of the derivative of f.

$$g[n] = \sum_{l=-\infty}^{\infty} f[l]h[n-l],$$

Preliminary: The step function is defined such that $\frac{d}{dt}u(t) = \delta(t)$

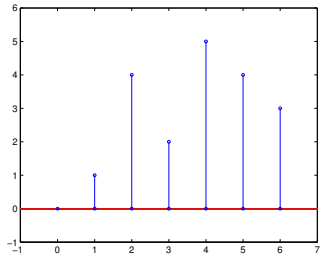


Figure 3: Discrete signal f[n]: how to compute an approximation of the derivative of f[n]?

The sign function comes handy: $\text{sgn}(t) = 2u(t) - 1,\ u(0) = 1/2$, or equivalently $u(t) = \frac{1}{2} + \frac{1}{2}\text{sgn}(t)$.

We have the following straightforward Fourier transforms:

$$u(t) \circ\!\!-\!\!\bullet \mathcal{U}(\omega) = \frac{1}{2}\delta(\omega) + \frac{1}{j\omega}$$

$$\frac{d}{dt}\text{sgn}(t) = 2\delta(t) \circ\!\!-\!\!\bullet j\omega\mathcal{F}(\text{sgn}(t)) = 2.$$

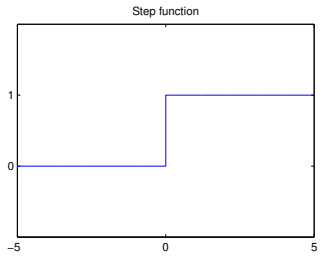


Figure 4: Step function.

A simple derivative filter Let's take the example of the following simple filter designed to approximate the derivative:

$$\begin{aligned} g[n] =& f[n] - f[n-1] \\ h[l] = \{1, -1\} \circ\!\!-\!\!\bullet H(\omega) =& \sum_{l=0}^{1} h[l]e^{-j\omega l} \\ =& 1 - e^{-j\omega} \\ =& 1 - \cos\omega + j\sin\omega \\ =& 2\sin^2\frac{\omega}{2} + j2\sin\frac{\omega}{2}\cos\frac{\omega}{2} \\ =& 2\sin\frac{\omega}{2}(\cos\frac{\omega}{2} + j\sin\frac{\omega}{2}) \end{aligned}$$

Therefore the magnitude of the transform is the following

$$|H(\omega)| = \left|2\sin\frac{\omega}{2}\right|$$

(to be continued)

CIS 580 Spring 2012 - Lecture 4

January 30, 2012

Notes and figures by Matthieu Lecce.

Review from last lecture:

- A discrete signal still has a continuous Fourier:

$$h[n] \circ\!\!-\!\!\bullet \sum_{n=0}^{L-1} h[n]e^{-j\omega n}$$
$$\omega = \frac{2\pi s}{L}$$
$$\text{DFT} \quad \omega = \frac{2\pi k}{L} \to H[k]$$

- Remember the effect of derivation in Fourier domain:

$$\frac{d}{dt}f(t) \circ\!\!-\!\!\bullet \boxed{j\omega} F(\omega)$$

Short Matlab tutorial

- fft of a $\cos(2\pi t/8)$: two spikes in the Fourier domain:

$$\cos(2\pi t/8)$$
$$\text{cont. FT} \quad \tfrac{1}{2}(\delta(\omega - \tfrac{2\pi}{8}) + \delta(\omega + \tfrac{2\pi}{8}))$$
$$\text{discrete FT} \qquad \omega = \tfrac{2\pi k}{L} = \tfrac{2\pi k}{32}$$
$$\omega = \tfrac{2\pi}{8}, \quad \tfrac{2\pi k}{32} = \tfrac{2\pi}{8}, \qquad k = 4$$
$$\omega = -\tfrac{2\pi}{8} \qquad k = -4$$

- In Matlab: $n = 0 \ldots L-1, k = 0 \ldots L-1$
- fftshift(fft(f)) performs a shift by $\frac{L}{2}$ (modulation of the signal by $e^{\frac{-j2\pi L/2}{L}} = e^{-j\pi} = -1$)

Derivative Filters (continued)

Two simple filters

REMEMBER THE PROBLEM WE STATED in the previous lecture:

- We are given a sampled signal and want to approximate the derivation of such a signal by "designing" a discrete filter h.
- In the Fourier domain, $H(\omega) = H(\omega + 2\pi)$ since the signal and filter are discrete.
- We want $H(\omega)$ to be as close as possible to $j\omega$, for $\omega \in [-\pi, \pi]$

Last time we found that the filter $h[n] = [1 \ -1]$ verifies:

$$h[n] \underbrace{\circ\!\!-\!\!\bullet}_{DTFT} H(\omega), \quad |H(\omega)| = \left|2 \sin \frac{\omega}{2}\right|$$

A more symmetrical form would be:

$$f[n] * h[n] = \frac{1}{2}(f[n+1] + f[n-1])$$

(In this case we are trying to approximate $\lim_{\varepsilon \to 0} \frac{f(x+\varepsilon)+f(x-\varepsilon)}{2\varepsilon}$)

Here $h[n] = [1/2 \ 0 \ -1/2] \circ\!\!-\!\!\bullet \ \frac{1}{2}e^{-j\omega(-1)} - \frac{1}{2}e^{-j\omega 1} = j \sin(\omega)$

THE TWO APPROXIMATIONS are represented in figure 1. Note that while the second filter we tried is a much better approximation of the first one, these two filter are a good approximation only for low frequencies.

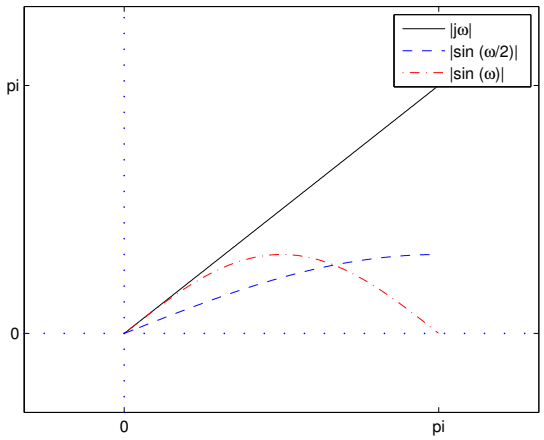


Figure 1: The derivation and its approximation by two simple filters. We represented the magnitude of the spectrum.

Ideal filter

WHAT WOULD BE THE IDEAL DERIVATIVE FILTER? Remember that we multiplied the **spectrum** with the box function to prove the sampling theorem, which corresponded to convolving the **original signal** with $\text{sinc}\left(\frac{\pi t}{T}\right)$ (where T is the sampling interval).

If the original signal is $h[n]$, the reconstructed signal is the following (convolution with the sinc interpolation function): (best reconstruction according to Shannon's theorem)

$$h(t) = \sum_{n=0}^{L-1} h[n] \text{sinc}\left(\frac{\pi}{T}(t-n)\right)$$

Note the above is now a continuous signal. Let's compute its derivative:

$$\frac{d}{dt}h(t) = \sum_n h[n]d(t-nT)$$

$$\text{where} \quad d(t) = \frac{d}{dt}\left(\frac{\sin(\frac{\pi t}{T})}{\frac{\pi t}{T}}\right) = \frac{1}{(\pi t/T)^2}\left[\frac{\pi t}{T}\cos\frac{\pi t}{T} - \frac{\pi}{T}\sin\frac{\pi t}{T}\right]$$

What we get is an infinite signal discretized as follows (see figure 2)

$$\ldots \quad 1/5 \quad -1/4 \quad 1/3 \quad -1/2 \quad 1 \quad 0 \quad -1 \quad 1/2 \quad -1/3 \quad 1/4 \quad -1/5 \quad \ldots$$

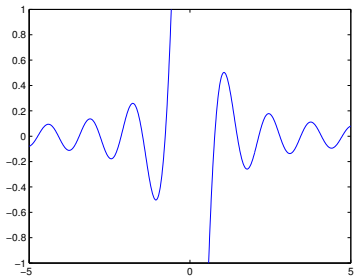


Figure 2: Derivative of the sinc interpolation function.

If we compute the Fourier transform of this, we see that it approximates $j\omega$ but is wiggly.

Least squares filter design

Given constraints on the length of my filter, how can we obtain the best derivative filter?

- The length should be L, so there are L unknowns
- The target is $j\omega$: more concretely we want $\sum_{n=0}^{-L} h[n]e^{-j\omega n} \approx j\omega$ for all ω

It is not possible to solve this system for all ω, so we solve a least-squares problem:

$$\min_{h[n]} \int_0^{\pi} \left(\sum_{n=0}^{L-1} h[n]e^{-j\omega n} - j\omega\right)^2 d\omega$$

This solves the length constraint problem but does not solve the smoothing problem. For smoothing we apply the Fourier transform of a Gaussian:

$$j\omega e^{-2\sigma^2\omega^2}$$

2D Fourier Transform

$$f(x,y) \circ\!\!-\!\!\bullet \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j\omega_x x+\omega_y y}dxdy$$
$$F(\omega_x,\omega_y) \bullet\!\!-\!\!\circ \left(\frac{1}{2\pi}\right)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\omega_x,\omega_y)e^{\cdots}d\omega_x d\omega_y$$

Shift

$$f(x-x_0,y-y_0) \circ\!\!-\!\!\bullet F(\omega_x,\omega_y)e^{-j(\omega_x x_0+\omega_y y_0)}$$

The shift is not as obvious to obtain as in the 1D case. This can be a problem in motion estimation.

Affine transformation We note $A\begin{bmatrix} x \\ y \end{bmatrix}$ a linear transformation of the 2D space.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$\begin{aligned}
& f\left(a_{11}x+a_{12}y, a_{21}x+a_{22}y\right) \\
& \updownarrow \\
& \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f\left(A(x;y)\right)\exp(-j\begin{bmatrix} \omega_x & \omega_x \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix})dxdy \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')\exp\left(-j\begin{bmatrix} \omega_x\omega_y \end{bmatrix}A^{-1}\begin{bmatrix} x' \\ y' \end{bmatrix}\right)det(A^{-1})dx'dy' && \text{Substitution } \begin{bmatrix} x' \\ y' \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix}) \\
&= det(A^{-1})F\left(A^{-T}\begin{bmatrix} w_x \\ w_y \end{bmatrix}\right) && A^{-T} = \frac{1}{det(A)}\begin{bmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{bmatrix}
\end{aligned}$$

If A is a rotation R ($R^T R = I$, also $det(R) = 1$), the formula is even simpler:

$$R\begin{bmatrix} x \\ y \end{bmatrix} \circ\!\!-\!\!\bullet \mathcal{F}\left(R\begin{bmatrix} w_x \\ w_y \end{bmatrix}\right)$$

Separable functions

$$f(x,y) = f_1(x)f_2(y) \circ\!\!-\!\!\bullet \int_{-\infty}^{\infty} f_1(x)e^{-j\omega_x x}dx \int_{-\infty}^{\infty} f_2(y)e^{-j\omega_y y}dy = F_1(\omega_x)F_2(\omega_y)$$

Examples:

- $f(x,y) = \cos(\omega_0 x)$ (see figure 3)

  It is important to notice that $f(x,y) = f(x)f(y)$ where $f(y) = 1$ is a constant function. Therefore the Fourier transform is the following:

$$f(w,y) \circ\!\!-\!\!\bullet \frac{1}{2}(\delta(\omega_x-\omega_0)+\delta(\omega_x+\omega_0)).\delta(\omega_y)$$

  .

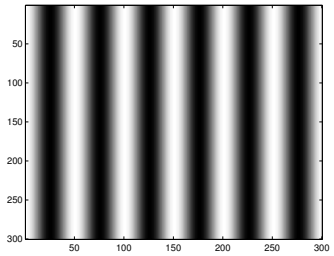


Figure 3: Unidirectional cosine

- $f(x,y) = \cos(\omega_1 x)\cos(\omega_2 y)$ corresponds to four points in the Fourier domain.
- $\cos(\frac{\sqrt{2}}{2}x - \frac{\sqrt{2}}{2}y)$: unidirectional cosine rotated of $\pi/4$

(to be continued)

CIS 580 Spring 2012 - Lecture 5

February 2, 2012

Notes and figures by Matthieu Lecce.

2D Fourier Transform

Review:

$$f(x,y) \multimap\!\bullet \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j\omega_x x+\omega_y y}dxdy$$

$$F(\omega_x,\omega_y) \bullet\!\multimap \left(\frac{1}{2\pi}\right)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\omega_x,\omega_y)e^{\cdots}d\omega_x d\omega_y$$

RESULTS FROM LAST LECTURE:

- Shift
- Affine transformation:

$$f\left(A\begin{bmatrix} x \\ y \end{bmatrix}\right) \overset{2D}{\multimap\!\bullet} \frac{1}{det(A)}F\left(A^{-T}\begin{bmatrix} \omega_x \\ w_y \end{bmatrix}\right))$$

- *Separability*: if a function is separable, the Fourier transform is separable too and the Fourier transform can be applied separately as a 1D transform along each axis.

  Examples:

- $f(x,y) = \cos(\omega_0 x) \multimap\!\bullet \frac{1}{2}(\delta(\omega_x-\omega_0)+\delta(\omega_x+\omega_0)).\delta(\omega_y)$
- $f(x,y) = \cos(\omega_1 x)\cos(\omega_2 y)$ corresponds to four points in the Fourier domain.
- $\cos(\frac{\sqrt{2}}{2}x - \frac{\sqrt{2}}{2}y)$: two points in the Fourier domain, along the direction $\pi/4$

Sampling in 2D

The 2D equivalent of the comb function is a "bed of nails":

$$\sum_n\sum_m \delta(x-n\Delta x, y-m\Delta y) = \sum_n \delta(x-n\Delta x)\sum_m \delta(y-m\Delta y) \multimap\!\bullet \sum_n\sum_m \delta\left(s_x-\frac{n}{\Delta x}, s_y-\frac{m}{\Delta y}\right)$$

In order to avoid sampling artifacts, we need a result similar to the 1D case:

$$\Delta x \leq \frac{\lambda_{x\min}}{2}$$

$$\Delta y \leq \frac{\lambda_{y\min}}{2}$$

Rotation

Using the result on affine transformations, we have:

$$f\left(R\begin{bmatrix} x \\ y \end{bmatrix}\right) \multimap\!\bullet\ F\left(R\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}\right)$$

Therefore the spectrum of $\cos(\omega_0(x\cos\phi - y\sin\phi))$ corresponds to two points along the direction ϕ, at distance ω_0 of the origin.

Wave signals

Consider an xyt-signal (video), in which we keep only one line (fixed y) of each frame. The effect of a translation in the video can be seen as a *wave* moving along the line at fixed y at constant speed u (in px/frame):

$$f(x,t) = f_0(x-ut)$$

Example: $f_0(x) = \cos(\omega_0 x)$, and $f(x,t) = \cos(\omega_0(x-ut))$

The Fourier transform of f is the following:

$$F(\omega_x,\omega_t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_0(x-ut)e^{-j\omega_x x+\omega_t t}dxdt$$

If we note $f_0(x) \multimap\!\bullet\ F_0(\omega_x)$ and make the substitution $x' = x-ut$, we have

$$\begin{aligned} F(\omega_x,\omega_t) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_0(x')e^{-j\omega_x(x'+ut)+\omega_t t}dx'dt \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_0(x')e^{-j\omega_x x'}e^{-j(\omega_x u+\omega_t)t}dx'dt \\ &= F_0(\omega_x)\delta(\omega_x u+\omega_t) \end{aligned}$$

Representing the spectrum $F(\omega_x,\omega_t)$ of the wave is equivalent to taking the original $F_0(\omega_x)$ spectrum, rotating it and stretching it by $\sqrt{1+u^2}$.

Examples:

- In our initial example $F_0(\omega_x) = \frac{1}{2}(\delta(\omega_x+\omega_0)+\delta(\omega_x-\omega_0))$: the $F_0(\omega_x,\omega_t)$ is made of two diracs located at $(\omega_0,-u\omega_0)$ and $(-\omega_0,+u\omega_0)$, which are at distance $\omega_0\sqrt{1+u^2}$ of the origin.
- If the spectrum $F_0(\omega_x)$ is a Gaussian, we have:

$$f_0(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}} \multimap\!\bullet\ F_0(\omega_x) = e^{-\sigma^2\omega_x^2/2},$$

which again corresponds to "rotating and stretching" the Gaussian.

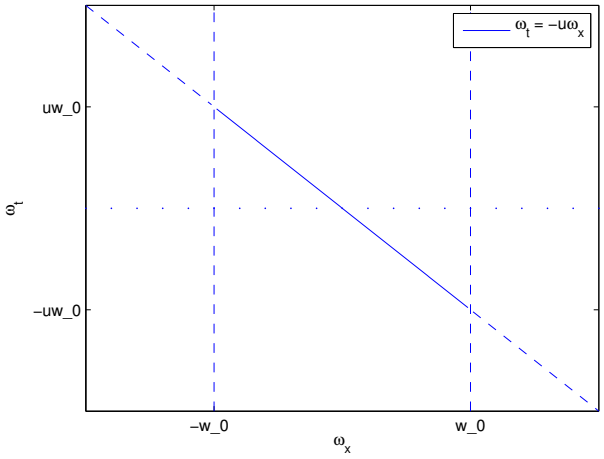


Figure 1: Wave signal in the xt Fourier domain: the spectrum $F_0(\omega_x)$ is multiplied by the indicator of te line $\omega_t = -u\omega_x$

Aliasing in video (Simoncelli, 1991)

Sampling in time is a multiplication by $\sum_{n=-\infty}^{\infty} \delta(t - n\Delta t)$. It corresponds to a convolution with $\sum_{n=-\infty}^{\infty} \delta(s_t - \frac{n}{\Delta t})$ (or $\sum_{n=-\infty}^{\infty} \delta(\omega_t = \frac{2\pi n}{\Delta t})$).

See 7. Reconstruction by boxing: aliasing causes perception of a speed of opposite sign.

$$\underbrace{u\omega_0}_{\max \omega_t} \leq \frac{\pi}{\Delta t} = \frac{\omega_{\text{sampling}}}{2}$$

Filters for detection

How to build a filter to recognize a cosine wave $\cos(\omega_0 x)$? In the frequency domain it corresponds to two impulses.

It is not possible to build a filter that looks like an impulse in Fourier domain, because it would correspond to a function of infinite support in time domain. Instead we could use a box filter: $H(\omega) = \Pi_{a/2}(\omega_0)$. The inverse function is still complicated ($\text{sinc}(\pi x/q)e^{j\omega_0 x}$, the harmonic exponential is here for modulation). A better candidate is a Gaussian filter centered around the frequency to detect, since the inverse transform of a Gaussian is a Gaussian.

(to be continued)

CIS 580 Spring 2012 - Lecture 6
Filters for detection

February 6, 2012

Notes and figures by Matthieu Lecce.

In this section, we are interested in building filters to detect patterns in a signal. By pattern we mean a portion of the signal with a distinctive behavior: edge, corner, portion oscillating at a specific frequency.

Gabor functions

Gabor functions can be considered as local band-pass filters: they are designed to detect portions of the signal that oscillate at a specific frequency ω_1: namely the output of the convolution tells us how much each position of the signal looks locally like a sinusoidal wave of frequency ω_1.

Gaussian filter

As an example, let's first consider a simple Gaussian filter:

$$g(t) = \frac{1}{\sigma\sqrt{(2\pi)}} e^{-t^2/2\sigma^2} \multimap\bullet\ G(\omega) = e^{-\sigma^2\omega^2/2}$$

What is the output of this filter for a cosine input? We have:

$$f(t) = \cos(\omega_0 t) \multimap\bullet\ \frac{1}{2}(\delta(\omega-\omega_0) + \delta(\omega+\omega_0))$$

Therefore in the Fourier domain the output spectrum is (see figure 1):

$$F(\omega)G(\omega) = e^{-\sigma^2\omega_0^2/2}(\frac{1}{2}(\delta(\omega-\omega_0) + \delta(\omega+\omega_0)))$$

And the output signal is (by taking the inverse transform and noticing that $e^{-\sigma^2\omega_0^2/2}$ *acts as a constant*):

$$\cos\omega_0 t * g(t) = e^{-\sigma^2\omega_0^2/2}\cos\omega_0 t$$

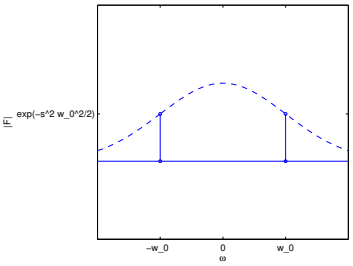


Figure 1: Spectrum of a cosine filtered by a Gaussian: $F(\omega)G(\omega)$

Gabor functions are modulated Gaussians

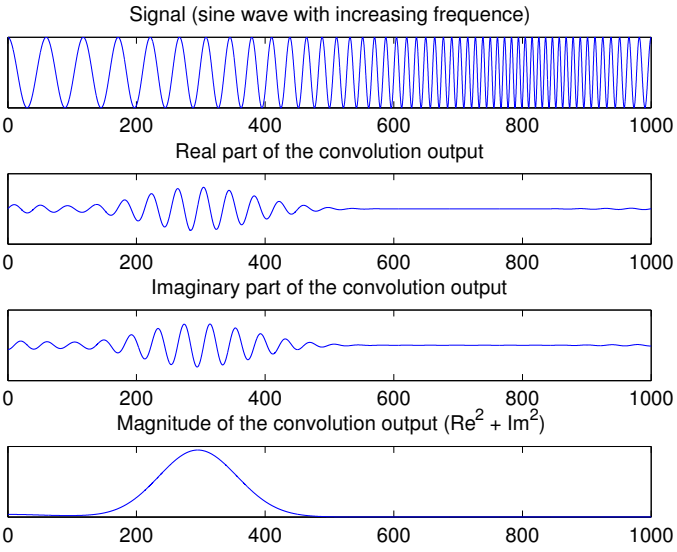
The Gaussian filter can be seen as a low-pass filter: it eliminates frequencies too far away from its mean, zero. How to obtain a band-pass filter instead? We can simply *modulate* the Gaussian filter, and we obtain what we call a *Gabor function*:

$$\begin{aligned} h(t) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-t^2/(2\sigma^2)} e^{j\omega_1 t} \\ &\updownarrow \\ H(\omega) &= e^{-\sigma^2(\omega-\omega_1)^2/2} \end{aligned}$$

See figure 2 for an example of Gabor filter (real and imaginary parts). What is the output of this modulated filter for $\cos(\omega_0 t)$? In the Fourier domain we obtain (see figure 3):

$$\frac{1}{2}(e^{-\sigma^2(\omega_1+\omega_0)^2/2}\delta(\omega+\omega_0)+e^{-\sigma^2(\omega_1-\omega_0)^2/2}\delta(\omega-\omega_0))$$

When the Gaussian is relatively narrow in the frequency domain, the small peak at $-\omega_0$ in figure 3 can be neglected, and the output is essentially a complex exponential of frequency ω_0, and of magnitude $e^{\sigma^2(\omega_1-\omega_0)^2}$. The good news is that as ω_0 gets away from ω_1, the magnitude decreases, producing the desired band-pass effect. The bad news is that the real and imaginary part of the output depend on t: they are *"phase-dependent"*, which means that even when we convolve a sine wave at fixed frequency with a Gabor filter, the real/imaginary components of the output are not constant but are waves of same frequency. Because the components are in *quadrature* (the phase difference is $\pi/4$), we simply take the norm of the output and obtain a non-oscillating response, as demonstrated in figure 4.



The phase-invariance of the norm of the Gabor output is important, since we want to be able to detect frequency patterns (and potentially edges and ridges) no matter what their phase is (how much they locally look like a sine or a cosine, or how they are translated).

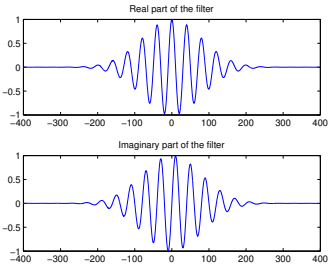


Figure 2: Example of Gabor filter: notice the exponential envelope, and the real and imaginary part are in *quadrature* (cos and sin)

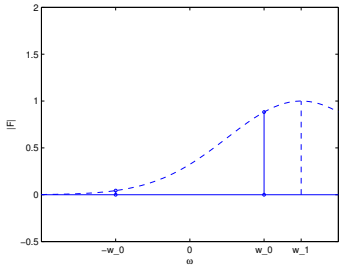


Figure 3: Spectrum of a cosine filtered by a modulated Gaussian: $F(\omega)G(\omega)$

Figure 4: Convolution of our Gabor example (fig. 2) with a signal at increasing frequency: notice that the outputs have same frequency as the input, only the magnitude change: it reaches a maximum exactly when the signal is oscillating at the frequency of the filter. *All these plots are in the time domain.*

Signal/spectrum spread: time localization VS frequency accuracy

We can use Gabor filters to localize pieces of a signal that oscillate at a certain frequency: for instance, if we were looking for the character "Waldo" in a 2D image, we could use the fact that he is wearing a distinctive red-and-white striped shirt and use a Gabor filter of appropriate frequency on the red channel of the image.

WHEN USING SUCH A FILTER FOR DETECTION, we are interested in the **spread** of a filter and its spectrum, because:

- a filter of small spread in the time/space domain enables precise localization of the pattern
- a filter whose *spectrum* has a narrow spread (think of it as a narrow band-pass filter) will enable precise detection of a specific frequency.

(for example in figure 4, the localization of the desired frequency is rough, as the filter in figure 4 has quite a large support)

The spread of a signal of finite support f is the second order moment $\int_{-\infty}^{\infty} t^2 f^2(t)dt$ and gives a measure of how wide the support is.

As we have observed in the past lectures, signals of small spread tend to have a wide spectrum.

This is formalized by a result named the **"uncertainty principle of signal processing"**:

- For a signal/filter of spread $\Delta t^2 = \int_{-\infty}^{\infty} t^2 f^2(t)dt$...
- ...whose spectrum has spread $\Delta\omega^2 = \int_{-\infty}^{\infty} \omega^2 F^2(\omega)d\omega$...
- ... we have $\Delta t \Delta\omega \geq$ lower bound.

2D Gabor function

We can derive a 2D version of the Gabor function to detect oscillating patters in images, at a given frequency ω_1 and *orientation* θ. Let's start from the **horizontal** version of the filter:

$$h(x, y, \sigma_1, \sigma_2, \omega_1) = \frac{1}{\sigma_1\sigma_2 2\pi} e^{-(x^2/2\sigma_1^2 + y^2/2\sigma_2^2)} e^{j\omega_1 x}$$

Notice that the Gaussian envelope is 2D, but the harmonic exponential is along one direction (x-xaxis here). If we want to detect oscillations at a different orrientation, we need a *rotated Gabor*:

$$h(x, y, \sigma_1, \sigma_2, \omega_1, \theta) = \frac{1}{\sigma_1\sigma_2 2\pi} e^{-1/2(x\ y)R^T()R(x\ y)^T} e^{j\omega_1(x\cos(\theta) - y\sin(\theta))}$$

Let's compute the Fourier of the 2D Gabor function:

$$e^{-(x^2/2\sigma_1^2 + y^2/2\sigma_2^2)} \multimap e^{-(\sigma_1^2\omega_x^2 + \sigma_2^2\omega_y^2)/2}$$

$$e^{-(x^2/2\sigma_1^2 + y^2/2\sigma_2^2)} e^{j\omega_1 x} \multimap e^{-(\sigma_1^2(\omega_1 - \omega_x)^2 + \sigma_2^2\omega_y^2)/2}$$

Now for the rotated version, notice that:

$$\frac{x^2}{2\sigma_1^2} + \frac{y^2}{2\sigma_2^2} = \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix},$$

and we apply a rotation R_θ to x and y [1]:

$$\begin{bmatrix} x' & y' \end{bmatrix} \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$= \begin{bmatrix} x & y \end{bmatrix} R_\theta \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix} R_\theta^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

[1] $\begin{bmatrix} x \\ y \end{bmatrix} = R_\theta \begin{bmatrix} x' \\ y' \end{bmatrix}$ where $R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$)

Since the diagonal matrix commutes and $R_\theta R_\theta^T = I$. Intuitively the Fourier of the rotation is the original Fourier, rotated of the same amount. Indeed, remember this result from last lecture:

$$h\left(R\begin{bmatrix} x \\ y \end{bmatrix}\right) \multimap\!\bullet\, H\left(R\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}\right)$$

Edge detection with Gaussian derivative filters

We now look at a different set of filters: Gaussian derivatives. They can also be seen as band-pass filters, but here instead of modulating a Gaussian, we look at its derivatives. Intuitively, because of the derivation rule for convolution, convolving an image with a Gaussian derivative is the same as smoothing the image (applying a Gaussian filter) and then looking at the derivatives of the output.

1D edge

An ideal edge is a step function $u(t)$. Instead of detecting the ideal edge, we can detect a smoothed edge. We have the following result from HW1:

$$\frac{d}{dt}(u(t) * g(t)) = g(t)$$

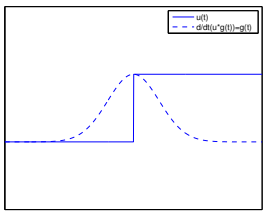


Figure 5: Edge detection with a Gaussian filter: $\frac{d}{dt}(u(t) * g(t)) = g(t)$.

Therefore the location of the edge can be given by $\max_t \frac{d}{dt}(f * g)(t)$, as shown in figure 5.

2D edge detection

The 2D version can be see as an image intensity that goes from a low to a high along a direction θ.

The isotropic Gaussian is the following:

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

We have:

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{1}{\sigma^2 2\pi} - \frac{2x}{2\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Now again, when we apply a rotation of angle $\theta > 0$
The function $g_x(R^T(x;y))$ takes the following very simple form:

$$g_x(R^T(x;y)) = \cos\theta g_x + \sin\theta g_y = (\cos\theta \ \sin\theta)\nabla g.$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = R_\theta \begin{bmatrix} x' \\ y' \end{bmatrix}$$
where $R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$.

To compute the derivative of an image in direction θ, all we need to do is the following:

$$\cos\theta(g_x * I) + \sin\theta(g_y * I)$$

This property is called **steerability**: we will cover it more extensively in the next lecture.

Recall that the 1D step edge can be localized as the maximum of $\frac{d}{dt}(u(t) * g(t))$.

In the 2D case, there is an edge at (x,y) if the first derivative of the convolution with the Gaussian has a maximum in the direction of the gradient.

Remark: the maximum is a non-linear operation. Is there a linear way to find edges? Yes, by taking the derivative of the above quantity: $\frac{d^2}{dt^2}(u(t) * g(t))$. Therefore an edge corresponds to a **zero-crossing of the second derivative of the smoothed signal.**

CIS 580 Spring 2012 - Lecture 8

February 10, 2012

Notes and figures by Matthieu Lecce.

Note on steerability of derivative filters

A FILTER IS "STEERABLE" when its response in any orientation θ can be computed as a **linear combination** of the responses from a few filters, called the "basis".

Steerability of the first Gaussian derivative Recall from the end of last lecture that the first Gaussian derivative is steerable. Indeed, if $g'(x,y,\theta)$ is the derivative of the Gaussian along direction θ, we have:

$$\begin{aligned} g'(x,y,\theta) &= \cos\theta g_x(x,y) + sin\theta g_y(x,y) \\ &= \underbrace{\begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} g_x \\ g_y \end{bmatrix}}_{\nabla G} \end{aligned}$$

This comes from the fact that derivation itself is steerable: the derivative of an image along a direction θ is simply the gradient of the image multiplied by a unit vector in direction θ: $I'(x,y,\theta) = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \nabla I$.

THE NICE PROPERTY OF STEERABILITY is that we can compute the derivative of a smoothed image in any direction by simply computing two derivatives! See figures 1,2 and 3.

The general definition is the following:

$$h(x,y,\theta) = \sum_{k=1}^{K} A_k(\theta) h_k(x,y) \qquad (1)$$

Figure 1: Original image.

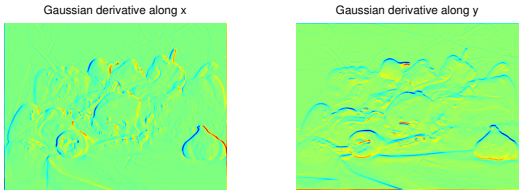


Figure 2: Convolution with horizontal and vertical Gaussian derivative filters: $\nabla I(x,y)$.

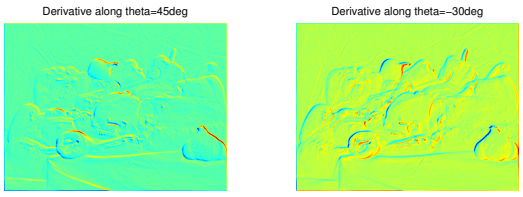


Figure 3: Derivative along any angle θ by simply taking the linear combination of the two derivatives shown in figure 2, with weights $\cos\theta$ and $\sin\theta$

Steerability of Gaussian derivatives of higher order Remember that the m-th derivative of a Gaussian has the following Fourier:

$$\frac{\partial^m g(x,y)}{\partial x^m} \multimap\!\bullet\ (j\omega_x)^m e^{-\omega_x^2\sigma^2/2} e^{-\omega_y^2\sigma^2/2}$$

What happens when we rotate the Gaussian?

$$\frac{\partial^m g(x\cos\theta + y\sin\theta)}{\partial x^m} \multimap\!\bullet\ j^m(\omega_x\cos\theta + \omega_y\sin\theta)^m e^{-\omega_x^2\sigma^2/2} e^{-\omega_y^2\sigma^2/2}$$

For $m = 2$, we have the following steerability formula as in equation 1:

$$\underbrace{\cos^2\theta}_{A_1(\theta)}\omega_x^2 + \underbrace{\sin^2\theta}_{A_2(\theta)}\omega_y^2 + \underbrace{2\sin\theta\cos\theta}_{A_3(\theta)}\omega_x\omega_y$$

Note that the exponential part is unchanged, as $\begin{bmatrix} \omega_x & \omega_y \end{bmatrix} R^T R \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} = \omega_x^2 + \omega_y^2$.

$$H_1(\omega_x,\omega_y) = j^2\omega_x^2 e^{-\sigma^2(\omega_x+\omega_y)^2/2}$$

$$\updownarrow$$

$$A_1(\theta)h_1(x,y) = \frac{\partial^2 g}{\partial x^2}\cos^2\theta$$

$$A_2(\theta)h_2(x,y) = \frac{\partial^2 g}{\partial y^2}\sin^2\theta$$

$$A_3(\theta)h_3(x,y) = \frac{\partial^2 g}{\partial x\partial y}2\cos\theta\sin\theta$$

Note that there are three components in the linear combination instead of 2. This is generalizable to higher order.

Discrete derivative filters Exact Gaussian derivatives satisfy steerability, but in concrete applications we sometimes need to use discrete approximations. For example we approximate the first derivative with the following filter: $d[k] = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix}$. Is such a filter steerable? We can check that steerability is satisfied in the frequency domain. Indeed, if a filter is steerable, its *Fourier transform* is steerable too:

$$f(x,y) \multimap\!\bullet\ F(\omega_x,\omega_y)$$

$$f\left(R\begin{bmatrix} x \\ y \end{bmatrix}\right) \multimap\!\bullet\ F\left(R\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}\right)$$

We would like to have the following (steerability of the FT of the filter):

$$D\left(R\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}\right) = \cos\theta D_1(\omega_x,\omega_y) + \sin\theta D_2(\omega_x,\omega_y).$$

The DTFT of the discrete filter we are trying to use is the following:

$$\begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \end{bmatrix} \multimap\!\bullet\ \frac{1}{2}e^{-j\omega_x(-1)} - \frac{1}{2}e^{-j\omega_x 1} = j\sin\omega_x.$$

Therefore we have $D(\cos\theta\omega_x + \sin\theta\omega_y) = j\sin(\cos\theta\omega_x + \sin\theta\omega_y)$ which is **not** the linear combination we would like to obtain (as in equation 1): this simple filter is not steerable. In general we use **binomial filters** (see Kosta's slides on those filters) to closely approximate Gaussian filters, and the steerability is approximately satisfied.

Scale Space and Scale Invariant features

We would like to build a (LSI) system that detects features (blobs, edges, corners...), **independently from their scale**: ideally it would output the location *and scale* ("intrinsic size") of the features.

Example: Detecting edges (steps) in a 1D cosine with the first Gaussian derivative filter $g'_\sigma = \frac{dg_\sigma}{dx}$ where $g_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$:

$$\cos \omega_0 x \quad \rightarrow \boxed{g'_\sigma(x)} \rightarrow \qquad \text{"smoothed derivative"}$$

$$\frac{1}{2}(\delta(\omega-\omega_0)+\delta(\omega+\omega_0))\, j\omega e^{-\omega^2\sigma^2/2}$$
$$= \frac{1}{2}(\delta(\omega-\omega_0)-\delta(\omega+\omega_0))\, j\omega_0 e^{-\omega_0^2\sigma^2/2}$$

(remember this trick from previous lectures)

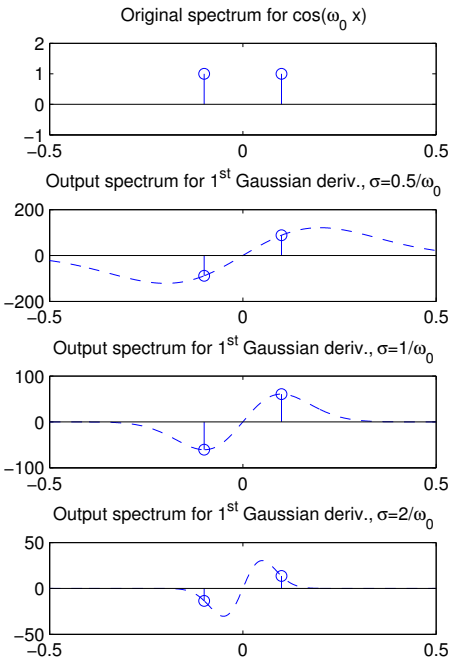


Figure 4: Cosine wave in Fourier domain, and spectrum of the convolution with a Gaussian first derivative filter. *This figure shows the imaginary part of the spectrum* (as shown above the output is purely imaginary). The peaks of the output are maximal when the sigma of the Gaussian filter matches the scale (ω_0) of the cosine.

As shown in figure 4, the output spectrum looks like the spectrum of a sine at frequency ω_0, which makes sense since we are computing a derivative here[1]. On top of simulating a multiplication by $j\omega$ (derivation), the Gaussian filter (actually its FT) acts as a *band-pass filter*: noise frequencies are filtered out.

Figure 4 also clearly introduces the problem of **scale**: if the σ of the Gaussian is not calibrated according to the "scale" (here period) of the cos wave,

[1] Note that of course here the edges are the inflexion points of the wave, since they are midway between high and low intensity points.

the amplitude of the output is not as high. Figure 5 shows what happens in the time domain: all outputs look like derivatives of the original wave, but it seems like a Gaussian derivative of sigma $\sigma = \frac{1}{\omega_0}$ yields maximum amplitude on edges. Can we explain the optimal $\sigma = \frac{1}{\omega_0}$? Also, how to devise a system that scans the wave at different scales (different σ) and automatically detects ω_0 for any wave?

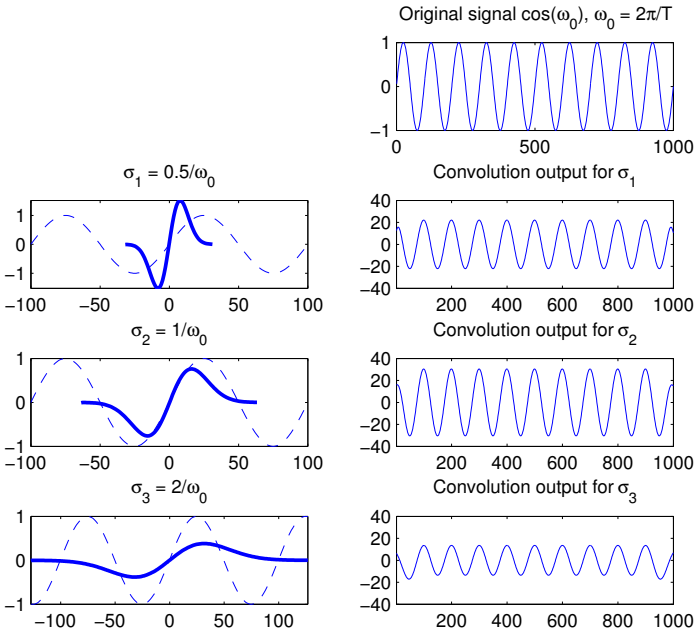


Figure 5: Cosine wave as in figure 4, but here we show the time domain. The left column compares the "size" of the filter used and the period of the wave, and the right column shows the output of the convolution. The amplitude of the output is maximal when the scale of the filter matches the one of the wave.

Gaussian filters and scale of cosine waves We generalize the example above (cosine wave filtered by a Gaussian first derivative) to a Gaussian filter of any order m ($m = 1$ in the example): $\frac{d^m g_\sigma}{dx^m}$.

The amplitude of the peaks (as in figure 4) of the spectrum of $(\cos \omega_0 x) \star \frac{d^m g_\sigma}{dx^m}$ is the following:

$$\left[|(j\omega)^m e^{-\omega^2\sigma^2/2}| \right]_{\omega=\omega_0} = \omega_0^m e^{-\omega_0^2\sigma^2/2}. \quad (2)$$

If we plot this as a function of σ, the argmax $\sigma_{\max}$ should indicate the intrinsic size of the wave. This maximum corresponds to a zero of the following derivative:

$$\frac{\partial(\omega_0^m e^{-\omega_0^2\sigma^2/2})}{\partial\sigma} = \omega_0^m \left(-\frac{\omega_0^2 2\sigma}{2} e^{-\omega_0^2\sigma^2/2} \right) = 0.$$

The obvious solution is $\sigma_{\max} = 0$ (a quick look at equation 2 confirms it, as it is a Gaussian centered at $\sigma = 0$!). Did we do something wrong?

Why is this theoretic result different from what figures 4 and 5 show, (i.e. maximum output amplitude is reached for a non-zero σ that corresponds to the scale of the wave)? The answer lies in the normalization of the Gaussian: figures 4 and 5 were obtained by using what we call a **scale normalized first Gaussian derivative**. The reason why we have to do scale normalization is that *as a Gaussian gets wider and wider, its peak value gets lower and lower*. Indeed if $g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$, and we multiply σ by a scaling factor s, we have:

$$g_{s\sigma}(sx) = \frac{1}{\sqrt{2\pi}s\sigma} e^{-(sx)^2/2(s\sigma)^2} = \frac{1}{s} g_\sigma(x). \tag{3}$$

This result, illustrated in figure 6, helps us understand why $g'_{2\sigma} \star \cos\left(\frac{1}{2}\omega_0 x\right)$ and $g'_\sigma \star \cos(\omega_0 x)$ do not have the same amplitude, although they *should* if we want to detect scale correctly. If we want to be able to compare the amplitudes of the convolutions for different σ, we need to multiply with something proportional to σ (*scale normalization*).

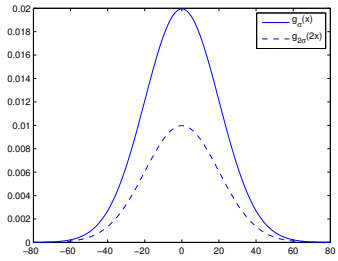


Figure 6: As a Gaussian gets wider, its peak gets lower (here $\sigma = 20$, and $g_{2\sigma}(2x) = \frac{1}{2} g_\sigma(x)$).

Scale normalization

It is interesting to notice that scale normalization is actually not necessary at order 0 (simple smoothing with no derivatives, i.e. convolution with g_σ). Indeed, if $I'(x) = I\left(\frac{x}{s}\right)$ is the scaled version of a signal $I(x)$ (I' is twice as big as I for $s = 2$), we have:

$$\begin{aligned}
(I' \star g_{s\sigma})(x) &= \int_{-\infty}^{\infty} g_{s\sigma}(x - x') I\left(\frac{x'}{s}\right) dx' \\
&= \int_{-\infty}^{\infty} \frac{1}{s} g_\sigma\left(\frac{x - x'}{s}\right) I\left(\frac{x'}{s}\right) dx' \quad \text{applying equation 3} \\
&= \int_{-\infty}^{\infty} g_\sigma\left(\frac{x}{s} - u'\right) I(u') du' \quad (u' = x'/s,\ du' = \frac{1}{s} du') \\
&= (I \star g_\sigma)\left(\frac{x}{s}\right)
\end{aligned}$$

$$\boxed{(I' \star g_{s\sigma})(x) = (I \star g_\sigma)\left(\frac{x}{s}\right)} \tag{4}$$

In English: scaling by s and then applying a *scaled* Gaussian is the same as applying a Gaussian and then scaling by s. The intuition behind this result is that the scaled Gaussian $g_{s\sigma}$, $s > 1$ has a lower peak than g_σ, but we integrate over s as many values when convolving with $I(sx)$, yielding the same intensity range. Figure 7 shows the application of this result in 2D.

The scalability result we derived (eq. 4) does not work for higher order ($m > 0$) Gaussian filters! Figure 9 shows what happens in our 1D cosine example when we scale the wave and convolve it with scaled first order Gaussian derivatives: the output (third row) has lower and lower amplitude as the scale increase. The fourth row shows how to normalize (simply multiply the output by the scale) so that the output amplitudes are comparable across

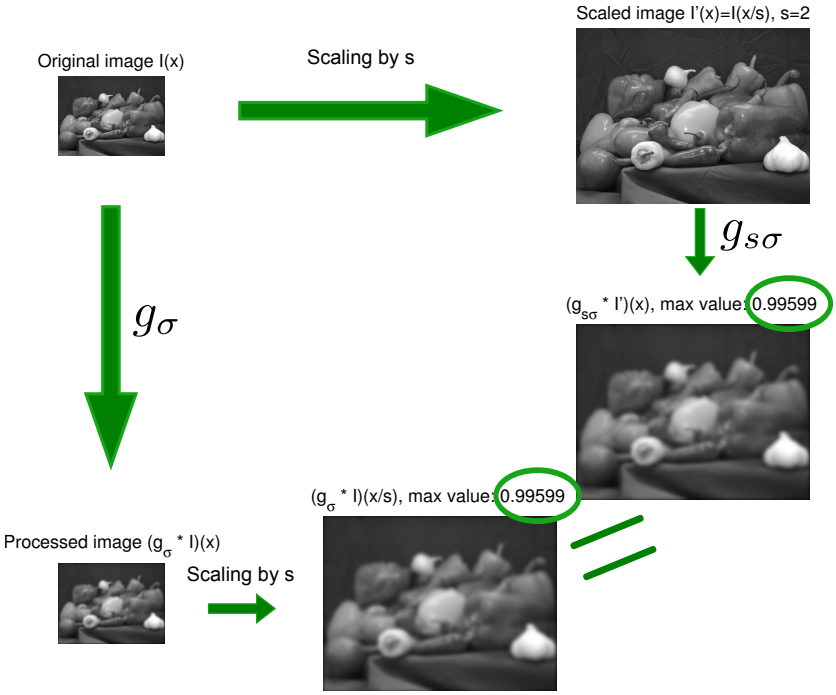


Figure 7: Scaling by s and then applying a *scaled* Gaussian $g_{s\sigma}$ is the same as applying a Gaussian g_σ and then scaling by s. We use the loose notation $I(x)$ for the image although of course it has two coordinates.

scales. Figure 8 shows the same problem in 2D (here we take the first order derivative w.r.t. x, $g'_\sigma = \frac{\partial g_\sigma}{\partial x}$). Here again, multiplying the output images by the scale would make them identical. The following simple derivation shows the result in 1D for $m = 1$:

$$\begin{aligned}
(I' \star g'_{s\sigma})(x) &= \left(I' \star \frac{dg_{s\sigma}}{dx}\right)(x) \\
&= \frac{d}{dx}[(I' \star g_{s\sigma})(x)] \\
&= \frac{d}{dx}\left[(I \star g_\sigma)\left(\frac{x}{s}\right)\right] \quad \text{(result we just showed at order 0)} \\
&= \frac{d}{d\xi}[(I \star g_\sigma)(\xi)]\frac{d(x/s)}{dx} \quad \text{(chain rule, } \xi = x/s\text{)} \\
&= \frac{d}{d\xi}[(I \star g_\sigma)(\xi)].\frac{1}{s}
\end{aligned}$$

$$(I' \star g'_{s\sigma})(x) = \frac{1}{s}(I \star g'_\sigma)\left(\frac{x}{s}\right) \tag{5}$$

As an exercise, you can practice generalizing this derivation to $m > 1$ and show that it introduces a factor $1/s^m$ instead of $1/s$, we will do the derivation in class next time:

$$\left(I' \star \frac{d^m g_{s\sigma}}{dx^m}\right)(x) = \frac{1}{s^m}\left(I \star \frac{dg_\sigma}{dx}\right)\left(\frac{x}{s}\right)$$

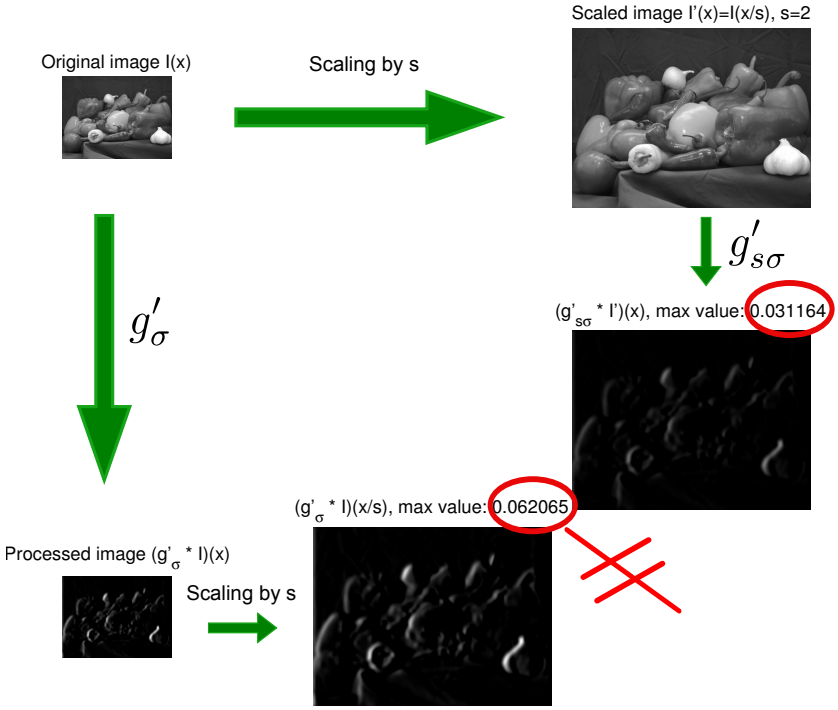


Figure 8: The scaling result we showed at order 0 (for g_σ) **does not apply** at order 1 (here we note g'_σ the first derivative of g_σ along x). We applied a scaling $s = 2$, and it seems like $(g'_{s\sigma} \star I')(x) = \boxed{\frac{1}{s}}(g'_\sigma \star I)(x/s)$. We formalize this result in this section.

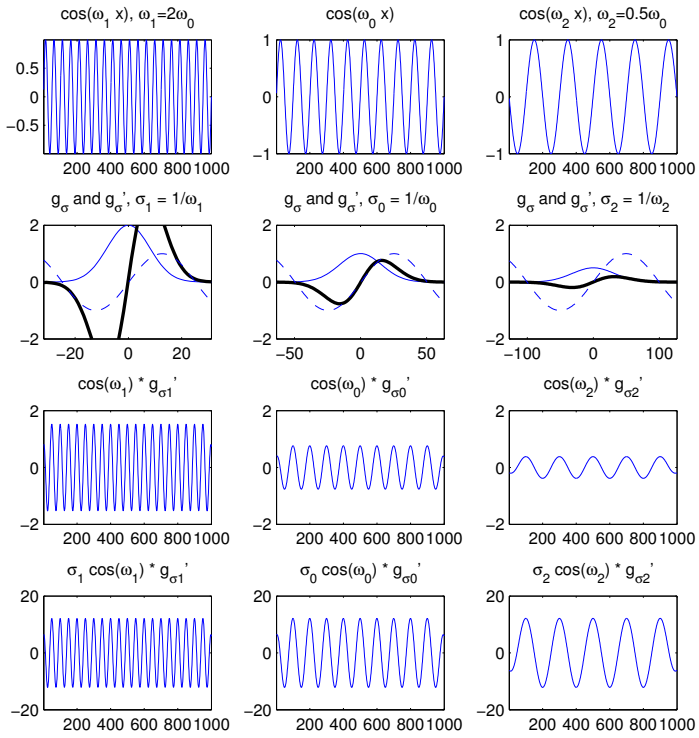


Figure 9: Scale normalization applied to our 1D example $g'_\sigma \star cos(\omega x)$.

No matter what the scale of the feature is (period ω_i of the wave here), the magnitude of the output *should* reach the **same** maximum value when the filter size matches the scale of the feature ($\sigma_i = 1/\omega_i$ here). Each column corresponds to a signal at a given scale.

The first row shows the three wave signals of same amplitude and different frequency.

The second row shows Gaussian filters (g_σ: thin blue curve, g'_σ: thick black curve) at optimal scale $\sigma_i = 1/\omega_i$ for each of the three waves: without multiplying by σ_i, the derivative filters g'_σ have lower and lower peak values as σ increases (one period of the signal signal is shown as a dashed curve, pay attention to the x-axis values).

The third row shows the convolution with the unnormalized derivative filters at optimal scale.

The last row shows the effect of multiplying by σ_i ("scale normalization"): the optimal output amplitudes are now comparable.

We can now go back to our initial example, the 1D cosine wave. Now that we now that the outputs of the m^{th} Gaussian derivative are shrunk by s^m when the scale is increased by s, we introduce a factor σ^m (scale normalization) to compensate for the shrinkage:

$$\begin{aligned}\frac{\partial(\sigma^m\omega_0^m e^{-\omega_0^2\sigma^2/2})}{\partial\sigma} =&(\omega_0^m e^{-\omega_0^2\sigma^2/2})\left(m\sigma^{m-1} - \frac{\omega_0^2}{2}2\sigma\sigma^m\right) = 0\\ \Leftrightarrow m\sigma^{m-1} =&\omega_0^2\sigma^{m+1}\\ \Leftrightarrow m =&\omega_0^2\sigma^2\\ \Leftrightarrow \sigma =&\frac{\sqrt{m}}{\omega_0} = \frac{\sqrt{m}}{2\pi}T\end{aligned}$$

This explains why in our example (first order) the optimal scale is $\sigma = \frac{1}{\omega_0}$. The gist of the detection framework would be the following: (1) build a scale space by smoothing the image/signal with different sigmas; (2) get the maximum responses in position and scale; indeed like a Gabor filter (see the corresponding lecture) detections can be local (the output shows local oscillations of high amplitude) (3) the size (wave period) T (or λ) of each local detection (local maximum in position and scale) is $T = \frac{2\pi}{\sqrt{2}}\sigma$ by applying the formula we just derived.

The famous SIFT features are computed with second order Gaussian derivatives ($m = 2$).

We introduced the concept of scale by using an example where scale is naturally interpreted as the period of a cosine wave. The concept can be generalized to any detection task where scale matters (blob, ridges...). We will formalize scale selection and extend it to 2D images in the following section and lecture.

Formalizing the 2D scale space

Lindeberg, 1991 (PhD thesis)

Notation: t will denote the *scale* $t = \sigma^2$, not the time.

Gaussians and heat diffusion One of the many interests of using Gaussians is that they satisfy the following **heat diffusion** partial differential equation (t is the scale):

$$\begin{aligned}\frac{\partial L}{\partial t} \underset{1D}{=}& \frac{1}{2}\frac{\partial^2 L}{\partial x^2}\\ \frac{\partial L}{\partial t} \underset{2D}{=}& \frac{1}{2}\nabla^2 L\\ L(x,y,t=0) =& \underbrace{I(x,y)}_{\text{image}}\end{aligned}$$

The solution to the PDE has the following simple form:

$$L(x,y,t) = g(x,y,t) \star I(x,y).$$

where $g(x,y,t)$ is the 2D Gaussian of variance $\sigma^2 = t$.

Scaling Gaussians Here we generalize the 1D result that we illustrated in figure 6 to the 2D case: the only difference is that a factor s^2 appears in the 2D version, instead of s in the 1D version (equation 3).

Let I' be a scaled version of an image I: $\boxed{I(x) = I'(sx)}$, where s is a scaling factor, which is also the potential size of the feature to detect).

- $L(x,t) = g(x,t) \star I(x)$ is the filtered *original* image
- $L'(sx,t') = g(sx,t') \star I'(sx)$ is the filtered *scaled* image.
- What should t' be so that $L(x,t) = L'(sx,t')$?
- We showed (eq. 5) that $t' = s^2 t$ verifies the above condition.
- Another simple proof would use Fourier:

  $I'(x) \circ\!\!-\!\!\bullet\, F(\omega_x)$, $I'(sx) \circ\!\!-\!\!\bullet\, \frac{1}{s} F\left(\frac{\omega_x}{s}\right)$

  (to be continued)

CIS 580 Spring 2012 - Lecture 9

February 13, 2012

Notes and figures by Matthieu Lecce.

Scale Space and Scale Invariant features

LAST TIME, we introduced the diffusion equation $\frac{\partial L}{\partial t} \underset{2D}{=} \frac{1}{2}\nabla^2 L$ with boundary conditions $L(x,y,t=0) = \underbrace{I(x,y)}_{\text{image}}$, and said that the solution is the image $I(x,y)$ convolved with a Gaussian ($t = \sigma^2$):

$$L(x,y,t) = g(x,y,t=\sigma^2) * I(x,y).$$

Indeed the necessary condition is $\frac{\partial g}{\partial t} = \frac{1}{2}\nabla^2 g$, in 1D $\frac{\partial g}{\partial t} = \frac{1}{2}\frac{\partial^2 g}{\partial x^2}$

WE INTRODUCED the *scalability* property of the Gaussian. Let I' be a scaled version of an image I ($I(x) = I'(sx)$, where s is the size)) and $L'(x',t') = g(x',t') * I'(x')$, we have:

Do not confuse s with t or σ, s in the intrinsic scale or "size" of a feature. We define $x' = sx$

$$\boxed{L(x,t) = L(x',t') \text{ for } t' = s^2 t}.$$

(Last time we showed it two ways: by computing the convolution and by using Fourier)

WE ALSO SHOWED that this is **not true for the derivatives** of the diffused image (recall they are equal to the image convolved with the Gaussian derivative):

$$\begin{aligned}\frac{\partial^m L(x,t)}{\partial x^m} &= \frac{\partial^m L'(x',t')}{\partial x'^m}\frac{\partial^m x'}{\partial x^m} \\ &= \frac{\partial^m L'(x',t')}{\partial x'^m} s^m\end{aligned}$$

since $x' = sx$

Last time we introduced and explained, with a 1D and a 2D example, the need for a scale normalization to compensate for this shrinkage of the output amplitude at bigger scales.

WE NOW FORMALIZE THIS CONCEPT of **normalized derivative** by substituting $\xi = \frac{x}{t^{\gamma/2}}$ ("γ-normalized" derivative). We have:

$$\begin{aligned}\frac{\partial^m L(x,t)}{\partial \xi^m} &= \frac{\partial^m L(x,t)}{\partial x^m}\frac{\partial^m x}{\partial \xi^m} \\ &= s^{m(1-\gamma)}\frac{\partial^m L(x',t')}{\partial x'^m} \\ &= \frac{\partial^m L(x',t')}{\partial x'^m} \quad \text{for } \gamma = 1\end{aligned}$$

We are mostly interested in the second derivative (Laplacian), because of the diffusion equation. In the non-normalized case, we have:

$$\frac{\partial g}{\partial t} = \frac{1}{2}\frac{\partial^2 g}{\partial x^2}$$

Normalized case (with $\gamma = 1$):

$$\frac{\partial^2 g}{\partial \xi^2} = \frac{\partial^2 g}{\partial x^2}\frac{\partial^2 x}{\partial \xi^2} = \sigma^2 \frac{\partial^2 g}{\partial x^2}$$

$$\xi = \frac{x}{t^{\gamma/2}} = \frac{x}{\sigma} \qquad t = \sigma^2, \quad \gamma = 1$$

$$\frac{\partial g}{\partial \sigma} = \sigma \frac{1}{\sigma^2}\frac{\partial^2 g}{\partial \xi^2} = \frac{1}{\sigma}\frac{\partial^2 g}{\partial \xi^2}$$

LET'S TAKE A MOMENT TO REALIZE HOW USEFUL THIS RESULT IS:

- Instead of filtering the image with a normalized 2^{nd}-derivative filter w.r.t. x, we can use $\frac{\partial g}{\partial \sigma}$

  The 2^{nd}-derivative filter is also called Laplacian of Gaussian (LoG)

- $\frac{\partial g}{\partial \sigma}$ can be simply approximated as a finite difference
- Therefore we have a simple and efficient way to compute the scale normalized second derivative filtering of an image I: we take a simple difference of two blurred versions of I.

  We saw that scale normalization is critical if we want responses to be comparable across scales.

APPROXIMATING $\frac{\partial g}{\partial \sigma}$

$$\frac{\partial g}{\partial \sigma} \approx \frac{g(x, \sigma + \Delta\sigma) - g(x, \sigma)}{\Delta\sigma}$$

We note $\sigma + \Delta\sigma = \kappa\sigma$ ($\kappa = 1.01$ would be a good approximation), and we have:

$$\frac{\partial g}{\partial \sigma} \approx \frac{g(x, \kappa\sigma) - g(x, \sigma)}{\sigma(\kappa - 1)} = \textbf{DoG}$$

A Difference of Gaussians (DoG) is the difference of two Gaussians of same center and different σ.

$$\text{DoG} = (\kappa - 1)\sigma\frac{\partial g}{\partial \sigma} = (\kappa - 1) \underbrace{\frac{\partial^2 g}{\partial \xi^2}}_{\text{normalized LoG}}$$

"Every DoG is a LoG" in the normalized case (which is the only case we care about of course).

We apply this recursively to build a pyramid, starting from original image:

- We convolve it with a Gaussian $g(x, \sigma)$
- We subsample it by 2
- We obtain $I_2(x) \circ\!\!-\!\!\bullet\, I(\omega) * \sum_{n=-\infty}^{\infty} \delta(\omega - n\pi)$

Because of the effect of subsampling in the frequency domain, we should eliminate all frequency components $|\omega| > \frac{\pi}{2}$. A rect-filter is not an option because it has an infinite impulse response (sinc).

Given that we want to maintain the scale space properties, we have to find out the effect of a *discrete Gaussian filter* (e.g. the binomial filter introduced in lecture 7). Remember that the DTFT of a discrete filter is:

$$H(\omega) = \sum_{k=-n}^{n} h[k] e^{-j\omega k}$$

the binomial filters also have the nice property that they sum to powers of 2 ($\sum_{k=1}^{n} \begin{pmatrix} n \\ k \end{pmatrix} = 2^n$), so they are easily normalized

(Slides)

Scale for feature matching

Scale is important to match features: the window size to compute local features (histogram) needs to be right in order to match features correctly.

Many recent algorithms apply scale detection (SIFT) at all positions in the image, but ideally we are interested in finding *interest points*, by finding

$$\max_{x,y} \max_{\sigma} \left(\sigma^2 \frac{\partial^2 g}{\partial x^2} * I(x) \right)$$

Note that some approaches don't take the maximum over positions and just compute the features at optimal scale for every point on a grid.

Blob detection We will implement a simple multi-scale blob detection system in the next homework. We will use the LoG filter because of its appearance similar to a blob. Inside the same octave, we will take simple differences of Gaussians to approximate the normalized Laplacian.

We will not implement octaves, which are obtained by subsampling.

After computing the DOG, we compute maximum at every pixel (wrt. σ), and **then** we try to find maximum with respect with position (using derivative approximation: Taylor expansion, see slides).

Play with Andrea Vedaldi's VLFeat toolbox for SIFT detection and descriptor computation.

CIS 580 Spring 2012 - Lecture 16

March 14, 2012

Notes and figures by Matthieu Lecce.

Review: Projective plane $\mathbb{P}^2$

$\mathbb{P}^2$ is the set of all projective equivalence classes, with *projective equivalence* defined as:

$$\begin{bmatrix} x \\ y \\ \omega \end{bmatrix} \sim \begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} \quad \text{iff} \quad \exists \lambda \neq 0 \quad \begin{bmatrix} x \\ y \\ \omega \end{bmatrix} = \lambda \begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} \quad \text{for} \quad \begin{bmatrix} x \\ y \\ \omega \end{bmatrix}, \begin{bmatrix} x' \\ y' \\ \omega' \end{bmatrix} \in \mathbb{R}^3 - \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\}$$

$\mathbb{P}^2$ is not a linear space, since it is made of equivalence classes.

- Both points and lines are represented with elements of $\mathbb{P}^2$.
- A line $l \sim \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ corresponds to equation $l^T x = ax + by + cw = 0$
- Points at infinity $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$ lie on the line $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ (corresponding to the equation $w = 0$).

Projective transformations

See Szeliski, chapter 9.

Projective transformations (homographies) are ubiquitous in computer vision. They are the most general type of linear transformations in $\mathbb{P}^2$, mapping $p \in \mathbb{P}^2$ to $p' \sim Ap \quad (\det A \neq 0)$.

Note that we will either write $p' \sim Ap$ or $\lambda p' = Ap$.

Properties

- Projective transformations preserve colinearity and concurrency.
- If $p'_1 \sim Ap_1$ and $p'_2 \sim Ap_2$ and $l \sim p_1 \times p_2$ is the line between p_1 and p_2, then the image of l is:

$$l' \sim Ap_1 \times Ap_2 \sim A^{-\top}(p_1 \times p_2), \quad \text{i.e.} \boxed{l' \sim A^{-\top} l}$$

$Ap_1 \times Ap_2 = \frac{1}{\det A} A^{-\top}(p_1 \times p_2)$

Determining a projective transformation for matching points Given a set of correspondences $\{(p_i, p'_i)\}$ known to be related $p'_i \sim Ap_i, \ \det A \neq 0$

Fact: A is the same as λA because $Ap \sim \lambda Ap$, hence A can be computed up to a scale factor.

Let's consider the mappings A_1, A_2, the points a', b', c', d' matched to

a, b, c, d, and the following equalities are verified:

$$\begin{bmatrix} a & b & c \end{bmatrix} \sim A_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} a' & b' & c' \end{bmatrix} \sim A_2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \sim A_2 A_1^{-1} \begin{bmatrix} a & b & c \end{bmatrix}$$

We have:

$$\begin{bmatrix} \vdots & \vdots & \vdots \\ a & b & c \\ \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots \\ \alpha a & \beta b & \gamma c \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Three points are not enough to determine the parameters. Let's add a fourth point:

$$d \sim A_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \sim \begin{bmatrix} \alpha a & \beta b & \gamma c \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \alpha a + \beta b + \gamma c$$

We have $\lambda d = \alpha a + \beta b + \gamma c$.

W.L.O.G we can set $\lambda = 1$, and $\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix}^{-1} d$.

Therefore 4 points are enough to determing the coefficients of the transformation: $\begin{bmatrix} a & b & c & d \end{bmatrix} \sim A_1 \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$.

Repeat the same for $\begin{bmatrix} a' & b' & c' & d' \end{bmatrix} \sim A_2 \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$.

We then have $p' \sim A_2 A_1^{-1} p$.

Camera model

(See slides) If matching points cannot be found but the rotation of the camera is known, the projection can simply be recovered from the following camera model:

- Pixel $u/w, v/w$
- $\frac{u}{w} = s_x \frac{X_c}{Z_c} + u_0$
- $\frac{v}{w} = s_y \frac{Y_c}{Z_c} + v_0$

- This is summarized as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \sim \underbrace{\begin{bmatrix} s_x & 0 & u_0 \\ 0 & s_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & T \end{bmatrix}}_{P \text{ projection matrix}} \underbrace{\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ W_w \end{bmatrix}}_{\in \mathbb{P}^3}$$

Important: P is 3×4, it is a projection but it is **not** a projective transformation, obviously because it is not invertible. If we consider only the 3D rotation (no translation) and the projection using K, we then have a projective transformation.

CIS 580 Spring 2012 - Lecture 17

March 19, 2012

Notes and figures by Matthieu Lecce.

$\mathbb{P}^2$ projective plane.

Projective transformation $x' \sim Ax$, $x, x \in \mathbb{P}^2$.

Projection model for a pinhole camera (see slides):

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \sim \underbrace{\begin{bmatrix} s_x & 0 & u_0 \\ 0 & s_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & T \end{bmatrix}}_{P \text{ projection matrix}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\in \mathbb{P}^3}$$

$(w \neq 0 \rightarrow u/w, b/w)$

Examples of projective transformations

Transformation from a horizontal plane to the image plane

Suppose that all points lie on a horizontal plane $Z = h$. There is a projective transformation between the plane in the world and the the camera screen.

Notation: $R = \begin{bmatrix} | & | & | \\ r_1 & r_2 & r_3 \\ | & | & | \end{bmatrix} \in SO(3)$
(special orthogonal group, $R^T R = I$)

$$\begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = Xr_1 + Yr_2 + hr_3 + t$$

$$= \begin{bmatrix} r_1 & r_2 & t + hr_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

replace $\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$ with $\begin{bmatrix} X \\ Y \\ W \end{bmatrix}$, the pixel coordinates of the projection verify:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \sim K \begin{bmatrix} r_1 & r_2 & t + hr_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

This clearly defines a projective transformation mapping (X, Y, W) to (u, v, w).

$\det \begin{bmatrix} r_1 & r_2 & t + hr_3 \end{bmatrix} = 0$ iff $(r_1 \times r_2)^T (t + hr_3) = r_3^T t + h = 0$

Parenthesis on rotations Consider a point (X, Y, Z) to which we apply a rotation followed by a translation to obtain $(X', Y'Z')$. If the columns of R define the axes of the rotated and translated frame (images of $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$) we have the following:

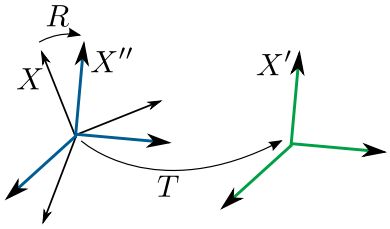


Figure 1: Coordinate transform.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} + T$$
$$= R \begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix}$$

To summarize if the camera coordinates are expressed as follows in terms of world coordinates:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T,$$

then we can simply transform camera coordinates back to world coordinates with the following expression:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R^T \left(\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} - T \right)$$

Horizontal plane expressed in the camera camera coordinate system

Let's consider a horizontal plane in the world coordinate system, of equation $\Pi_h : Z = h$. We would like to find the equation of this same plane in the **camera** coordinate system, i.e. we would like to find n and d veryfying the following:

$$(X_c, Y_c, Z_c) \in \Pi_h \Leftrightarrow n^T \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = d$$

Using the result we just derived, we can express the "world" Z in terms of camera coordinates:

$$Z = r_3^T \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} - r_3^T T$$

Therefore the equation of Π_h in camera coordinates simply is:

$$r_3^T \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = h + r_3^T T$$

The parameters we were looking for are $n = r_3^T$ and $d = h + r_3^T T$.

Panoramic stitching

(Ch.9 Szeliski)

We consider a set of images taken from the same point of view: the camera transformation is a pure rotation.

Note that in practice it is hard to do achieve a perfect pure rotation (a robot would need pan-tilt unit).

For a given 3D point, we note X_0, Y_0, Z_0 the coordinates of the point in a first camera frame and X_1, Y_1, Z_1 in a second camera frame, rotated w.r.t. the first frame.

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = R \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix}$$

If the camera zoom is used or if auto-focus is enabled, the intrinsic parameters are also different for the two images: K_0, K_1 denote the two intrinsic matrices. The two camera models enable us to derive the following equalities:

We will use the symbols X, Y, Z for metric coordinates coordinates and u, v, w for pixel coordinates.

$$\begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} \sim K_0 \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \sim K_0 \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix}$$

$$\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} \sim K_1 \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix}$$

$$\boxed{\begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} \sim K_0 R K_1^{-1} \begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix}}$$

$\det(K_0 R K_1^{-1}) = \frac{\det K_0 \det R}{\det K_1} = \frac{s_{x0} s_{y0}}{s_{x1} s_{y1}} \neq 0$

Therefore, if we know the camera parameters, we can warp and align images resulting from any pure camera rotation without knowing the scene.

Change in focal length When the only change that affects the intrinsics is a change of focal length, the intrinsic matrix K_1 takes the following form:

$$\begin{bmatrix} \lambda s_x & 0 & u_0 \\ 0 & \lambda s_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Assuming $K_0 = \begin{bmatrix} s_x & 0 & u_0 \\ 0 & s_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$. Note that we do not put a λ in the last diagonal coefficient, otherwise it would cancel out : $u = \lambda s_x \frac{X}{\lambda Z} + u_0$

Projections of vanishing points

Let's relate the projections of the vanishing points to the extrinsic and intrinsic camera parameters. Consider three orthogonal vanishing points as in figure 2.

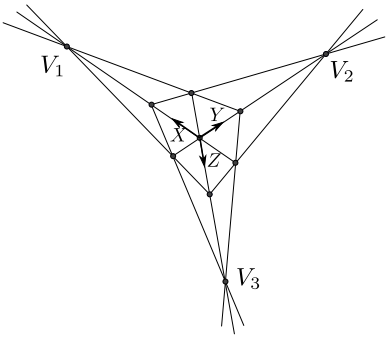


Figure 2: Three orthogonal vanishing points.

The horizontal vanishing point takes the following form:

$$v_1 \sim \begin{bmatrix} R & T \end{bmatrix} \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\in \mathbb{P}^3} \sim K r_1$$

$R = \begin{bmatrix} | & | & | \\ r_1 & r_2 & r_3 \\ | & | & | \end{bmatrix}$

A similar reasoning applied to v_2, v_3 yields the following result:

$$\begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \sim K \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}$$

We can therefore derive a constraint on K, using the orthogonality of R and the above result:

$$r_i \sim K^{-1} v_i, \qquad r_i^\top r_j = 0, \, i \neq j$$
$$\boxed{v_i^T K^{-\top} K^{-1} v_j = 0}$$

If we assume the following simple form of K:

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

the expression in the constraint takes the following form:

$$K^{-\top} K^{-1} = \begin{bmatrix} 1/f^2 & 0 & 0 \\ 0 & 1/f^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For two of the vanishing points, the constraint translates as follows:

$$\frac{v_{xi} v_{xj}}{f^2} + \frac{v_{yi} v_{yj}}{f^2} + v_{wi} v_{wj} = 0$$

For $i = 1, 2, 3$, we note the vanishing points $v_i = \begin{bmatrix} v_{xi} & v_{yi} & v_{wi} \end{bmatrix}$, $v_{wi} = 0$ if the vanishing point is at infinity.

Therefore, **we can compute the focal length** f if $v_{wi} v_{wj} \neq 0$, i.e. if none of the two vanishing points are at infininty.

CIS 580 Spring 2012 - Lecture 18

March 26, 2012

Notes and figures by Matthieu Lecce.

Review: examples of projective transformations

- There is a projective transformation between any horizontal plane $Z = h$ and the camera screen.
- More generally there is a projective transformation betwee any plane and the camera screen.
- A purely rotating camera induces a projective transformation.

Single-view geometry

Vanishing points

We consider the simple case where the intrinsic matrix of a camera takes the following form:

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Last time, we saw how to compute the focal length of a camera, given the projections of the vanishing points.

We saw that under the condition that vanishing points are not at infinity, we can derive f from the constraint on K.

Can we recover the image center from the projections of three orthogonal vanishing points?

THEOREM: The image center (u_0, v_0) is the orthocenter of the triangle formed by the projections of three orthogonal vanishing points.

PROOF: See figure 1. Let $C = (u_0, v_0, 1)$ denote the homogeneous coordinates of the image center: it is defined as the intersection of image plane $V_1V_2V_3$ with the optical axis, which is the line through O and perpendicular to $V_1V_2V_3$.

$$OC \perp V_1V_2V_3 \Rightarrow OCV_1 \perp V_1V_2V_3$$
$$OV_1 \perp OV_2 \Rightarrow OCV_1 \perp \text{any line contained in } V_1V_2V_3$$

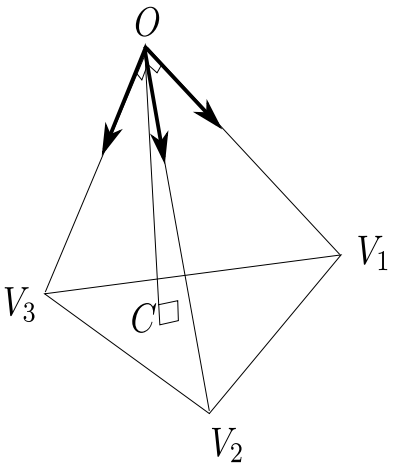


Figure 1: Three orthogonal vanishing points and image center.
We used the fact that the two non-parallel lines OV_2 and V_2V_3 contained in the plane OV_2V_3 are perpendicular to OCV_1, therefore $OV_2V_3 \perp OCV_1$.

In particular $OCV_1 \perp V_2V_3$. Moreover, $OV_1 \perp OV_2$, therefore $\boxed{OCV_1 \perp OV_2V_3}$.

When two planes are perpendicular, their intersections with a third plane are also perpendicular. Therefore, the intersection of OCV_1 with $V_1V_2V_3$ is perpendicular to intersection of OV_2V_3 with $V_1V_2V_3$.

In other words $V_1C \perp V_2V_3$. A similar reasoning leads to $V_2C \perp V_3V_1$ and $V_3C \perp V_1V_2$, therefore C is the orthocenter of $V_1V_2V_3$.

This elegant result would enable one to calibrate a camera using only the vanishing points in an image. In practice we don't use this method because it

is reliable only when there is a strong perspective in the image, i.e. when vanishing points have low coordinates. When it is not the case, the intersection of parallel lines projected in the image have large coordinates (thousands of pixels), and the relative error is too big to guarantee precise calibration.

Cross-ratios

See fig. 3.

Definition Given four points A, B, C, D, we define the cross-ratio of their distances as follows: $CR(A,B,C,D) = \frac{AC}{AD} : \frac{BC}{BD}$

Invariance of the cross-ratio It is easy to prove that $CR(A,B,C,D)$ remains invariant under projective transformations $\mathbb{P}^n \to \mathbb{P}^n$:

$$\frac{AC}{AD} : \frac{BC}{BD} = \frac{A'C'}{A'D'} : \frac{B'C'}{B'D'}$$

Simple example Consider the following projective transformation in $\mathbb{P}^1$:

$$\begin{bmatrix} u' \\ w' \end{bmatrix} \sim \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix}, \quad ad - bc \neq 0$$

We have the following:

$$\frac{u'}{w'} = \frac{au + bw}{cu + dw}$$

W.L.O.G we can set $w = w' = 1$, which yields:

$$u' = \frac{au + b}{cu + d}$$

If we consider four points of coordinates $u_1, u_2, u_3, u_4 \in \mathbb{R}^1$, we have:

$$\begin{aligned}
\frac{u'_3 - u'_1}{u'_4 - u'_1} : \frac{u'_3 - u'_2}{u'_4 - u'_2} &= \frac{\frac{au_3+b}{cu_3+d} - \frac{au_1+b}{cu_1+d}}{\frac{au_4+b}{cu_4+d} - \frac{au_1+b}{cu_1+d}} : \frac{\frac{au_3+b}{cu_3+d} - \frac{au_2+b}{cu_2+d}}{\frac{au_4+b}{cu_4+d} - \frac{au_2+b}{cu_2+d}} \\
&= \frac{(cu_4+d)(cu_1+d)}{(cu_3+d)(cu_1+d)} \frac{(cu_3+d)(cu_2+d)}{(cu_4+d)(cu_2+d)} \cdot \\
&\quad \frac{(au_3+b)(cu_1+d) - (au_1+b)(cu_3+d)}{(au_4+b)(cu_1+d) - (au_1+b)(cu_4+d)} \frac{(au_3+b)(cu_2+d) - (au_2+b)(cu_3+d)}{(au_4+b)(cu_2+d) - (au_2+b)(cu_4+d)} \\
&= \frac{(au_3+b)(cu_1+d) - (au_1+b)(cu_3+d)}{(au_4+b)(cu_1+d) - (au_1+b)(cu_4+d)} \frac{(au_3+b)(cu_2+d) - (au_2+b)(cu_3+d)}{(au_4+b)(cu_2+d) - (au_2+b)(cu_4+d)} \\
&= \frac{(ad-bc)(u_3-u_1)}{(ad-bc)(u_4-u_1)} \frac{(ad-bc)(u_3-u_2)}{(ad-bc)(u_4-u_2)} \\
&= \frac{u_3-u_1}{u_4-u_1} : \frac{u_3-u_2}{u_4-u_2}
\end{aligned}$$

Point ordering and cross-ratios For the same set of four points A, B, C, D, there are 24 ways to write a cross-ratio (permutations of A, B, C, D) to obtain $\lambda, 1-\lambda, \frac{1}{\lambda}, \frac{1}{1-\lambda}$.

Cross-ratios containing vanishing points The invariance holds even when the cross-ratio contains vanishing points. For $A, B, C \in \mathbb{P}^3$, we obtain the following result by starting from the cross-ratio defined above, and taking the limit when one of the points goes to infinity:

$$CR(A, B, C, \infty) = \lim_{D \to \infty} \frac{AC}{BC} : \frac{AD}{BD} = \frac{AC}{BC}$$

Notational abuse: $\infty \leftrightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

This trick provides us with a useful geometrical reasoning to "transfer" distances in an image:

- Let's assume we know the pixel positions of A', B', C', the projections of three points A, B, C,
- Let's assume we also know the pixel position of V', projection of the vanishing point V

We know that there is a projective transformation between the plane containing A, B, C and the camera screen. Therefore, if we know AC we can obtain BC and vice versa:

$$\underbrace{\frac{A'C'}{A'D'} : \frac{B'C'}{B'D'}}_{\text{Known pixel positions}} = \frac{AC}{BC}$$

EXAMPLE 1: See fig. 4. Given this picture, what is the distance to the finish line? *Assume D* is at infinity (the two lines are parallel):

$$\lambda_{px} = \frac{x}{1+x} \rightarrow x = \dots$$

.

EXAMPLE 2: Given a picture of the William Penn statue (Town Hall) and the Liberty tower # 1, and the horizon of the ground plane, find the height of the Lierty tower given the fact the W. Penn statue has a height from the ground of 167m.

See fig. 5.

The horizon is the intersection of two horizontal vanishing points. We know the vertical vanishing point of the line through the Liberty tower (LP=Liberty Place).

How can we find which point on Liberty place has a height of 167m? Q, the intersection of the horizon and AA', is a *horizontal vanishing point*, therefore any line through Q is parallel to AA' ! Therefore the point we are looking for is B (on figure 6, intersection of QB' and vertical line through LP).

In the world, the pre-image of BB'Q is parallel to the ground, which means that the pre-image of AB is 167m long.

$$\{A, B, L, V\}_{\text{pixels}} = \frac{AL}{A\infty} : \frac{BL}{B\infty} = \frac{AL}{BL} = \frac{AL}{AL - 167} = f(AL)$$

Therefore we can transfer the length from Town Hall to Liberty place if we know the points above. In practive, A and A' are hard to find because of occlusions.

ANOTHER EXAMPLE: picture of a man standing in front of his house. See fig. 7.

$$\{A,B,C,V\}_{\text{pixels}} = \frac{AC}{A\infty} : \frac{BC}{B\infty} = \frac{AC}{BC} = \frac{AC}{AC-h} = \frac{220}{220-h} \rightarrow h = \ldots$$

CIS 580 Spring 2012 - Lecture 19

March 30, 2012

Notes and figures by Matthieu Lecce.

Review: single-view geometry

- The image center is the orthocenter of the projections of three orthogonal vanishing points.
- The cross-ratio $CR(A,B,C,D) = \frac{AC}{AD} : \frac{BC}{BD}$ is invariant w.r.t. to projective transformations.
- If D is at infinity, $CR(A,B,C,D) = \frac{AC}{BC}$. The knowledge of a point at infinity and *one* distance enables us to recover many other distances by computing corresponding cross-ratios in the image. *Remark*: no calibration is needed.

Cross-ratios and distance transfer (continued)

See figure.

$$\frac{AC}{BC} = \lambda_1, \quad \frac{AE}{CE} = \lambda_2, \quad AE = AC + CE$$

($(A,B,C,D)_{px}$ and $(A,C,E,D)_{px}$) x.

From the above we can retrieve CE.

This setting enables us to reconstruct all possible rectangular boxes in the view. See figure: P is on the same plane as the bottom face of the cube "ground plane"

Pose from a single view

We are interested in answering the question "Where is the camera?"

Given n points in the world coordinate system (measured as metric distances, not just ratios as previously) and their projections (in pixels), we want to compute the **camera extrinsics**.

Extrinsics: translation and rotation (T,R) w.r.t. the world coordinate system.

This problem is at the core of many applications: augmented reality, special effects, etc. The challenge is actually not to solve the linear system to recover the camera extrinsics (which we will do in this section), but to track and match beacon points, i.e. establish a correspondence betwee 2D points in the camera image and 3D points in the world coordinates

In precise applications like movie post-production, the tracking is done in a semi-automatic way, and sometimes hundreds of CG artists just have to click on thousands of images to manually track the reference points.

Remember the pixel position (u,v) of the projection of a beacon point verifies the camera model:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} \sim \underset{3\times3}{K} \underbrace{\begin{bmatrix} R & T \end{bmatrix}}_{3\times4} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

In most applications, we assume K does not vary and is obtained by calibration, and R, T are the unknowns

When intrinsics are unknown

Assume we have a set of 2D-3D matchings $(u, v, w) \leftrightarrow (X_w, Y_w, Z_w)$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} M & m \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + m$$

We define $M = KR$ and $m = KT$ for clarity.

The following results simply states that a 3D point is on the ray going through the camera center and the point on the screen:

$$\boxed{\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = \lambda M^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - M^{-1} m}$$

Notice that $-M^{-1}m$ is the projection (camera) center in world coordinates. This equation defines a ray (line) that goes through the camera center and the point on the screen.

Given n 2D-3D correspondences (associations between 3D beacon points and their 2D projections), we can recover $P = \begin{bmatrix} M & m \end{bmatrix}$ using the following equations:

$$u = \frac{p_1^T X_w}{p_3^T X_w}, \quad v = \frac{p_2^T X_w}{p_3^T X_w}$$

We note $P = \begin{bmatrix} p_1^\top \\ p_2^\top \\ p_3^\top \end{bmatrix}$.

We obtain two such equations per point. The projection matrix P can be computed up to a scale factor, so it has $3 * 4 - 1 = 11$ independent unknowns. Therefore, 6 points determine a unique (up to a scale) P-matrix *if and only if 4 of them are not coplanar*.

When the intrinsics are known (photogrammetry augmented reality)

Now, let's assume K is known, therefore we only have to estimate R and T. We know the **back-projection** of the point on the screen:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Note that (x, y) are metric coordinates, whereas (u, v) are pixel coordinates.

The following simple equation is similar to the one we derived when K is unknown:

$$\boxed{\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T}$$

See fig. 3. For three 3D points A, B, C, we know AB,BC,AC,δ_{AC},δ_{AB},δ_{BC}. Unknowns are $\lambda_A, \lambda_B, \lambda_C$.

We use the law of cosines:

$$AC^2 = \lambda_A^2 + \lambda_C^2 - 2\lambda_A \lambda_C \cos \delta_{AC}.$$

(to be continued)

CIS 580 Spring 2012 - Lecture 20

April 2, 2012

Notes and figures by Matthieu Lecce.

Review:

Single-view geometry

- We showed how focal length and projection center of a camera could be recovered from the projections of three orthogonal vanishing points on the screen.
- We showed how the camera pose can be recovered from a set of 2D-3D matchings, when K is unknown (*uncalibrated* pose estimation)
- This time we will cover *calibrated* pose estimation.

Single-view geometry (continued)

Pose from a single view, using 3 points

Here we assume the camera is *calibrated*, i.e. K is known.

Given A, B, C in world (object) coordinate system, their projections a, b, c in calibrated coordinates ($a \sim K^{-1}a_{\text{pixels}}$). We want to recover the camera extrinsics (R, T). This problem is very similar to the problem of GPS localization (triangulation).

See figure 1. From the law of cosines, we have:

$$\begin{aligned} BC^2 &= d_B^2 + d_C^2 - 2d_B d_C \cos\delta_{BC} \\ AC^2 &= d_A^2 + d_C^2 - 2d_A d_C \cos\delta_{AC} \\ AB^2 &= d_A^2 + d_B^2 - 2d_A d_B \cos\delta_{AB} \end{aligned}$$

If we introduce u, v such that $d_B = ud_A$ and $d_C = vd_A$, we have:

$$\begin{aligned} BC^2 &= d_A^2(u^2 + v^2 - 2uv\cos\delta_{BC}) \\ AC^2 &= d_A^2(u^2 + v^2 - 2uv\cos\delta_{AC}) \\ AB^2 &= d_A^2(u^2 + v^2 - 2uv\cos\delta_{AB}) \end{aligned}$$

Therefore we have the following:

$$\begin{aligned} \frac{BC^2}{(u^2 + v^2 - 2uv\cos\delta_{BC})} &= \frac{AC^2}{(u^2 + v^2 - 2uv\cos\delta_{AC})} \\ \frac{AB^2}{(u^2 + v^2 - 2uv\cos\delta_{AB})} &= \frac{AC^2}{(u^2 + v^2 - 2uv\cos\delta_{AC})} \end{aligned}$$

These two equations of second order in u, v yield one equation of fourth order: we find four solutions for u^2, which correspond to eight pairs (u, v).

Each of the pairs (u, v) yields a set of d_A, d_B, d_C values, and finally:

$$\begin{aligned} OA &= d_A a = RA + T \\ OB &= d_B b = RB + T \\ OC &= d_C c = RC + T \end{aligned}$$

(where a, b, c have been normalized to unit vectors)

Absolute pose or absolute orientation

Let's consider two sets of 3D points (as measured by a Kinect sensor for instance), and assume we want to estimate the transformation that maps one set to the other. See figure 2. $\boxed{P_1^i = RP_2^i + T}$ $(R \in SO(3), T \in \mathbb{R}^3)$.

It can be easily shown that the **least-squares** translation estimate between the two point clouds is given by the following formula:

$$T = \underbrace{\frac{1}{n}\sum P_1^i}_{\overline{P_1}} - R\underbrace{\frac{1}{n}\sum P_2^i}_{\overline{P_2}} = \overline{P_1} - R\overline{P_2}$$

We now want to find R verifying the following equation:

$$P_1^i = RP_2^i + \overline{P_1} - R\overline{P_2}$$

We can rewrite the above equation in the simple form $P = RQ$. As usual, this equation cannot generally be solved exactly, so perform a least-squares estimation. The squared Frobenius norm of $P - RQ$ can be expressed as follows:

$P = P_1 - \overline{P_1} + R\overline{P_2}$, $Q = P_2$

$$\begin{aligned} \|P - RQ\|_F^2 &= \text{tr}\left((P - RQ)^T (P - RQ)\right) \\ &= \text{tr}\left(P^T P + Q^T \underbrace{R^T R}_{I} Q - P^T RQ - Q^R T\right) \\ &= -\text{tr}\left(P^T RQ + Q^T RP\right) \\ &= -2\text{tr}\left(R \underbrace{QP^T}_{H}\right) \end{aligned}$$

We use the fact that $\text{tr}(A^T) = \text{tr}(A)$ and the trace is invariant to circular permutation of the matrix product

To summarize we are trying to find $\max_R \text{tr}\left(R \underset{3\times3}{H}\right)$

THEOREM: If H is symmetric and positive definite, and R is orthogonal, then:

$$\text{tr}(H) \geq \text{tr}(RH)$$

To maximize $\text{tr}(RH)$, we want to find an R such that $\text{tr}(RH)$ is positive definite. We can achieve this by performing an SVD decomposition of H: $H = USV^T$, set $R = VU^T$ then $RH = VU^T USV^T = VSV^T$ which is symmetric, positive and definite.

Therefore we can use the following recipe to estimate R:

- take the SVD of $QP^T = H = USV^T$,
- return optimal $R = VU^T$.

Is it that simple? **No**, we forgot to take into account that we want $R \in \mathcal{SO}(3)$. If we set $R = VU^T$ with $V^TV = U^TU = I$, we have $R^TR = UV^TVU^T = I$, and $\det R = 1$ or -1. Therefore instead of $R = VU^T$, we should return the following *normalized* estimate:

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(UV^T) \end{bmatrix} U^T$$

.

Two views uncalibrated

- 1966 soccer game, goal incorrectly awarded because of perspective error.
- *Goal-directed Video Metrology*, I. Reid and A. Zisserman

Two views synced Let's assume that we know the projective transformation of the ground $x_1 \sim Px_2$, $b_1 \nsim Pb_2$

$l_1 \sim b_1 \times v_1$, $l_2 \sim b_2 \times v_2$, f_1, f_2 are the back-projections of l_1, l_2 on ground plane, f_1, f_2 have to intersect at F because both planes (l_1, l_2) and

If $x_1 \sim Px_2$ then $l_1 \sim P^{-T}l_1'$, l_1' is the projection of f_1 on the second image plane. Then $f \sim l_1 \times l_2$

We have done all this without any knowledge of the focal length.

Two views of a plane

See figure 4. $X_1 = RX_2 + T$, $x_1 = \begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix}$, $x_2 = \begin{bmatrix} x_2' \\ y_2' \\ 1 \end{bmatrix}$, $Z_1x_1 = RZ_2x_2 + T$.

Plane equation w.r.t second camera $N^TX_2 = d$, or equivalently $\frac{N^TX_2}{d} = 1$

CIS 580 Spring 2012 - Lecture 21

April 4, 2012

Notes and figures by Matthieu Lecce.

Review: (Camera) pose estimation

- Given 2D-3D point correspondences

$$p \sim \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} P \\ 1 \end{bmatrix} \rightarrow p' = \lambda p$$

- 3D-3D motion estimation between two 3D point clouds: least-squares estimation of R and T. When correspondences are not established, ICP (iterative closest point) is performed by iteratively matching points in the first cloud to their closest neighbors in the second cloud, estimating (R, T) and recomputing matches between closest neighbors.

See ARToolKit augmented reality toolbox available on most platforms: `http://en.wikipedia.org/wiki/ARToolKit`

Pose estimation: special cases

Pose estimation on a plane

Consider a 2D transformation $P' = RP + T$ for $P, P' \in \mathbb{R}^2$

This estimation problem can be applied in a 3D setting where all the perceived 3D points are in the same plane, along with the camera center: for instance, consider the situation depicted in figure 1, where you are standing on the ground (plane $Y = 0$), taking a picture of two lighthouses (say two monuments that are visible far away, located at the same height $Y = 0$) and trying to determine your position and orientation, knowing the absolute positions of the lighthouses. Given "lighthouse" points (X_i, Z_i) and the bearings (projections in the image), we want to find T.

See figure 1. The projection of a lighthouse point (X_i, Z_i) in your camera is given by the following equations:

$$x_i = \frac{X_i \cos\theta - Z_i \sin\theta + T_x}{X_i \sin\theta + Z_i \cos\theta + T_z}, \qquad y_i = 0.$$

We therefore need two lighthouse points to solve for (T_x, T_z).

Useful geometry result: The locus of points "seeing" two points with a fixed angle is a circle. If three points are given then the intersection of two circles is the solution and one of the points.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, T = \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

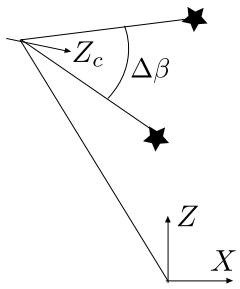


Figure 1: World frame X, Y, Z, camera frame X_c, Y_c, Z_c and two "lighthouse" points of known 3D. Here all the points are on the plane $Y = 0$, therefore we only show X, Z. $\Delta\beta$ is called the relative bearing.

Motion estimation on a plane

Let's consider the same setting as previously, with two cameras now: all points observed and the two camera centers are in the same plane. Let's assume we don't know the world frame.

Two 3D points P and Q are observed by two cameras 1 and 2, as depicted in figure 2. Can we recover the orientation and length of the **baseline**? We can recover the orientation of the baseline and the rotation between the two cameras, but the translation between the two cameras will be up to a scaling factor.

The baseline is the vector between the two camera centers

We denote P_1 the coordinates of P in the frame of camera 1, and p_1 the projection of P_1 on the screen of camera 1:

$$P_1 = RP + T, \quad P_1 = \lambda p_1$$

The problem can be formulated as follows: given (p_1, p_2) (cameras are calibrated i.e we know intrinsics), find R, T with T up to a scaling factor.

If the points lie on a plane $N^T p_2 = d$ w.r.t the second coordinate system, we have: $\frac{N^T p_2}{d} = 1$

$$\begin{aligned}\lambda_1 p_1 =& \lambda_2 R p_2 + T\frac{N^T p_2}{d} \\ =& \lambda_2\left(R + \frac{TN^T}{d}\right)p_2\end{aligned}$$

The baseline and reconstructions are up to a scaling factor because an object that is twice as big but placed twice as far away from the camera centers, with a baseline twice as long, producces the same images.

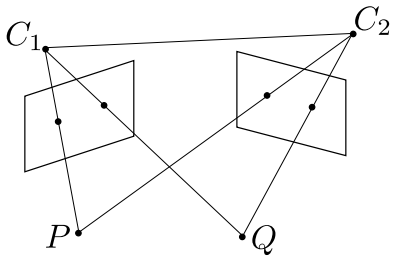


Figure 2: We assume C_1, C_2, P, Q are coplanar.

Suppose we obtain H such that $p_{1i} \sim Hp_{2i}$ $(i = 1, \ldots, 4)$ using four pairs of matching points, then the transformation and plane parameters are related to the homograpy as follows:

$$\underset{3\times3}{H} = \lambda\left(R + \frac{TN^T}{d}\right), \quad \|N\| = 1$$

R has 3 unknowns since $R^T R = I$, T has 3 unknowns and N has two unknowns. Note that although H is defined up to a scaling factor, the relationship $p_1^T H p_2 = 0$ is independent of the depth λ.

Can we recover rotation and translation of the plane from the homography? Recovering them would for instance enable us to take two pictures of a facade, and recover the normal vector to the facade and the relative rotation of the cameras. We have:

$$\begin{aligned}H^T H =& \lambda^2\left(R^T + \frac{NT^T}{d}\right)\left(R + \frac{TN^T}{d}\right) \\ =& \lambda^2\left(I + \frac{R^T TN^T}{d} + \frac{NT^T R}{d} + \frac{1}{d^2}NT^T TN^T\right) \\ =& \lambda^2(I + uN^T + Nu^T + Nu^T uN^T)\end{aligned}$$

We have to check that the system is not singular, but it looks ok because it's an orthogonal matrix + a matrix of form UV^T.

We define $u = \frac{1}{d}R^T T$.

By noticing that almost all the terms in the parentheses define projections on N or u, we obtain the following simple equation by multiplying both sides of the equation by the vector $u \times N$:

$$H^T H(u \times N) = \lambda^2(u \times N) \tag{1}$$

Therefore λ^2 is one of the eigenvalues of $H^T H$. Which of the three eigenvalues of $H^T H$ is λ^2? Consider the terms within the parentheses in the previous derivation:

$$\begin{aligned} uN^T + Nu^T + Nu^T uN^T &= wv^T + vw^T + vw^T wv^T \\ &= (v+w)(v+w)^T - ww^T \end{aligned}$$

We define $v = \|u\|N, \qquad w = \frac{u}{\|u\|}$

By defining $Q = (v+w)(v+w)^T - ww^T$ (such that $H = \lambda^2(I+Q)$), we obtain a result similar to equation 1:

$$Q(v \times w) = 0$$

Q has eigenvalues $\mu_1 > 0, \mu_2 = 0, \mu_3 < 0$. The eigenvalue λ^2 of $H^T H$ is associated to the eigenvalue $\mu_2 = 0$ of Q. Therefore λ^2 is the second eigenvalue of H.

It remains to find the eigenvectors of $Q = \frac{H}{\lambda^2} - I$. The task is now to extract T and N from Q. We summarize the definition of Q:

$$\begin{aligned} Q &= (w+v)(w+v)^T - ww^T \\ u &= \frac{1}{d} R^T T \\ v &= \|u\| N \\ w &= \frac{u}{\|u\|} \end{aligned}$$

Indeed if the eigenvalues of Q are the follwoing:

$$\mu_1,\ \mu_2 = 0,\ \mu_3,$$

then the eigenvalues of $H^T H$ are $\lambda^2(1+\mu_1),\ \lambda^2,\ \lambda^2(1+\mu_3)$:

$$\begin{aligned} Qu &= \mu u \\ \Leftrightarrow \lambda^2(I+Q)u &= \lambda^2(1+\mu) \\ \Leftrightarrow H^T H u &= \lambda^2(1+\mu)u \end{aligned}$$

CIS 580 Spring 2012 - Lecture 23

April 11, 2012

Notes and figures by Matthieu Lecce.

Reconstruction from two calibrated views

Structure from motion, epipolar geometry

We want to estimate the transformation between two camera and the 3D positions of a set of observed points, given 2D-2D correspondences in the two images: (u_1, v_1) matched to (u_2, v_2).

Assuming a simple camera model with focal length f, the calibrated rays are directed by the following vectors:

$$p_1 = \begin{bmatrix} \frac{u_1 - c_{x1}}{f_1} \\ \frac{v_1 - c_{y1}}{f_1} \\ 1 \end{bmatrix}, \quad p_2 = \begin{bmatrix} \frac{u_2 - c_{x2}}{f_2} \\ \frac{v_2 - c_{y2}}{f_2} \\ 1 \end{bmatrix}$$

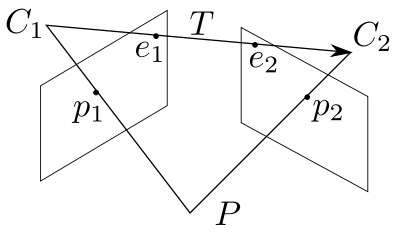


Figure 1: R, T transform the axes of 1 into the axes of 2, therefore in terms of coordinates $P_1 = RP_2 + T$.

The epipolar constraint relates p_1, p_2 and the motion parameters:

$$\boxed{p_1^T (T \times R p_2) = 0}, \tag{1}$$

and it just states that the two calibrated rays are coplanar in a plane that goes through the baseline T.

Note that $\lambda_2 p_2 = R\lambda_2 p_2 + T$, but the epipolar constraints enables us to get rid of the depth.

If we fix p_1, the constraint defines a line which is simply the projection of the ray p_1 in the screen of 2. Such a line is called an **epipolar line**.

All epipolar lines in 2 go through e_2, the projection of C_1 in the screen of 2. e_1, e_2 are called the **epipoles**.

GEOMETRIC INTERPRETATION: In figure 1, notice that the plane C_1C_2P intersects screen 1 at the epipolar line $e_1 \times p_1$, and it intersects screen 2 at the epipolar line $e_2 \times p_2$. Both lines are defined from the epipolar constraint (eq. 1), by fixing p_1 or p_2. Using this geometric interpretation, it is easy to derive e_1, e_2 either from the epipolar constraint:

$$e_1 \sim T, \quad e_2 \sim -R^T T$$

Note on coordinate transforms If the axes of camera 2 are translated of T and rotated of R with respect to the axes of camera 1, the relationship between the coordinates is the following:

$$P_1 = RP_2 + T$$

Notice that what happens to the coordinates is the "opposite" of what happens to the axes: if C_2 is translated of 100 along X with respect to C_1, then we need to *subtract* 100 to the X-coordinate of P_1 to obtain P_2 if P_1, P_2 describe the same point P

Therefore we have another way to derive e_1, e_2:

$$\begin{aligned} e_1 &= R.0 + T, \quad \text{i.e. } e_1 = T \\ 0 &= Re_2 + T, \quad \text{i.e. } e_2 = -R^T T \end{aligned}$$

The essential matrix

We define the essential matrix $\boxed{E = \widehat{T}R} \in \mathbb{R}^{3\times3}$, where $\widehat{T} = \begin{bmatrix} 0 & -t_z & -t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$

Using the anti-symmetric matrix $\widehat{T}$ enables us to write the cross-product with T as a matrix-vector product: for any $P \in \mathbb{R}^3$, $T \times P = \widehat{T}P$.

The epipolar constraint can then be reformulated very simply:

$$p_1^T E p_2 = 0$$

Naturally, this linear equation involving the parameters of E encourage us to estimate E using several pairs of matching points (p_1, p_2). The interest of estimating the essential matrix is two-fold:

- Once we have E, we can plot epipolar lines: given a point p_2, then the corresponding point in the first image is on the following line:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \underbrace{Ep_2}_{3\times1} = 0$$

- Since E is defined with the help of T and R, we can first estimate E from a set of correspondences, and then recover the motion parameters from E.

Estimating E As usual, we want to formulate and solve a linear system $Ae = 0$ where A is some matrix containing the point positions and e contains the parameters of E, to estimate. Given a pair of matching points p_1, p_2, we can further rewrite the constraint as follows, writing E in form of a vector:

$$\underbrace{\begin{bmatrix} p_{2x}p_1^T & p_{2y}p_1^T & p_{2z}p_1^T \end{bmatrix}}_{1\times9} \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}}_{9\times1} = 0 \qquad (2)$$

The vector $\begin{bmatrix} p_{2x}p_1 \\ p_{2y}p_1 \\ p_{2z}p_1 \end{bmatrix} \in \mathbb{R}^9$ is called the Hadamard product of p_1 and p_2 and is noted $p_1 \otimes p_2$.

Therefore equation 2 can be rewritten as $(p_1 \otimes p_2)^T e_s = 0$.

where $e_s = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \in \mathbb{R}^9$ is just E written as a vector, i.e. $E = \begin{bmatrix} | & | & | \\ e_1 & e_2 & e_3 \\ | & | & | \end{bmatrix}$.

For every correspondence p_1, p_2 we obtain a linear homogeneous equation w.r.t. $\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = e_S$: we need **8 independent equations**, i.e. 8 matching points.

E can be estimated up to a scale factor

The 8-point algorithm

By stacking eight independent equations as defined in equation 2, we obtain a system of form $Ae_S = 0$ (A is 8×9): e_s is in the **null-space** of A, hence $e_s = v_9$ if $A = U\Sigma V^T$ is the SVD of A and $V = \begin{bmatrix} | & | & | & | \\ v1 & v_2 & \dots & v_9 \\ | & | & | & | \end{bmatrix}$.

If we recover E with this method, are we sure it will be an "acceptable" essential matrix? We have to define what it means from a matrix to be essential: concretely it means that is is the product of an antisymmetric and a special orthogonal matrix. Can any 3×3 real matrix E be decomposed into $E = \widehat{T}R$?

We know other examples of decompositions that work for *any* 3×3 matrix:

- QR factorization: $A = QR$ (Q orthogonal, R upper triangular),
- SVD decompositions, $A = U\Sigma V^T$

Is "$\widehat{T}R$" a valid decomposition for any 3×3 matrix?

E-properties

Here are a few properties that will be useful in subsequent derivations:

- $E^T = R^T \widehat{T}^T$
- $\widehat{T}^T = -\widehat{T}, \widehat{T}a = T \times a$
- $E^T T = 0$.
- $\det(E) = \det \widehat{T}. \det R = 0$, therefore $\sigma_3 = 0$ (σ_3 is the smallest singular value of E)

Remember that the determinant of a matrix is the product of its eigenvalues, and for any matrix A, the null-space of A is the same as the null-space of $A^T A$.

IN THIS SECTION, we are going to show that *an essential matrix can be characterized by its singular values.* Namely if E is an essential matrix (i.e. decomposable into $\widehat{T}R$), its singular values verify $\sigma_1 = \sigma_2$, and $\sigma_3 = 0$. We just proved that $\sigma_3 = 0$.

Remember that the singular values of E are the eigenvalues of $E^T E$ or EE^T, i.e. the solutions of the following characteristic polynomial:

$$\det EE^T - \sigma I = 0.$$

We want to characterize the singular values without solving the characteristic polynomial. We exploit the fact that $R^T R = I$ to find a nice form for EE^T:

$$\begin{aligned} EE^T &= \widehat{T}\widehat{T}^T \\ &= TT^T - T^T T I \\ &= \begin{bmatrix} t_x^2 & t_x t_y & t_x t_z \\ 0 & t_y^2 & t_y t_z \\ 0 & 0 & t_z^2 \end{bmatrix} - \|T\|^2 I \end{aligned}$$

If $\|T\| = 1$, then there exists a rotation U such that $UT = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = T_z$

We can always rotate a vector to align it to the Z-axis.

And we are going to try to express EE^T as a function of the following simple matrix:

$$\widehat{T}_z \widehat{T}_z^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

LEMMA: If Q is orthogonal ($Q^T Q = I$), then

$$\widehat{Qa} = Q\widehat{a}Q^T$$

PROOF: $\widehat{Qa}b = Qa \times b = Q(a \times Q^T b) = Q\widehat{a}Q^T b$.

We are now able to express $\widehat{T}$ as $\widehat{T_z U^T} = U^T \widehat{T} U$

$$\begin{aligned} EE^T = \widehat{T}\widehat{T}^T &= U^T \widehat{T}_z U U^T \widehat{T}_z^T U \\ &= U^T \widehat{T}_z \widehat{T}_z^T U \\ &= U^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} U \end{aligned}$$

EE^T is symmetric hence $\sigma_1 = 1, \sigma_2 = 1, \sigma_3 = 0$. We can now formulate conditions that characterize essential matrices.

NECESSARY CONDITION: If E is an essential matrix (i.e. $E = \widehat{T}R$) then E has two equal singular values and one singular value equals zero.

SUFFICIENT CONDITION:

$$\begin{aligned} E &= U \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \\ &= \sigma U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \\ &= \sigma U \widehat{T}_z^T R_z V^T \\ &= \sigma \underbrace{\widehat{UT_z}}_{\text{antisymmetric}} \underbrace{URV^T}_{\text{orthogonal}} \end{aligned}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = T_z^T R_{z,\pi/2}$$

Observe $UT_z = U \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, which is the last column of U

We just showed that there is at least one such decomposition, but is it unique?

NECESSARY AND SUFFICIENT CONDITION: E is essential **iff** $\sigma_1(E) = \sigma_2(E) \neq 0$ and $\sigma_3(E) = 0$.

How many ways can we decompose E?.

We showed the following decomposition:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{-\widehat{T}_z} \underbrace{\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_{z,\pi/2}}$$

This is important, not just for mathematical purposes: we need to find *all* solutions of the system, in case some of them are not correct (do not satisfy some constraints)

But we could similarly write $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \widehat{T}_z R_{z,-\pi/2}$.

Pose recovery from the essential matrix

If $E = U\Sigma V^T = U \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$, there are two solutions for the pair $(\widehat{T}, R)$:

$$(\widehat{T}_1, R_1) = (UR_{z,+\pi/2}\Sigma U^T, UR_{z,+\pi/2}^T V^T)$$
$$(\widehat{T}_2, R_2) = (UR_{z,-\pi/2}\Sigma U^T, UR_{z,-\pi/2}^T V^T)$$

CIS 580 Spring 2012 - Lecture 24

April 16, 2012

Notes and figures by Matthieu Lecce.

Review: pose recovery from E:

Theorem: A real 3×3 matrix is essential (i.e. $E = \widehat{T}R$) **iff** it has two equal singular values and one zero singular value:

$\widehat{T}$ anti-symmetric, R orthogonal

$$E = U \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

$$T = \pm u_3 \quad \text{(third column of } U\text{)}$$

$$R = U \begin{bmatrix} 0 & \pm 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$$

To summarize, when trying to recover the motion parameteres from E, there is more than one solution: if (T, R) is a solution, then $(-T, R_{T,\pi}R)$ is a also a solution .

$R_{T,\pi}$ is a rotation of π around T ($R_T = I + 2\widehat{T}^2$)

This phenomenon is called the **twisted pair ambiguity**: it can be proved that if $p_1^T(T \times Rp_2) = 0$ then $p_1^T(T \times R_{T,\pi}Rp_2) = 0$ too.

RODRIGUES FORMULA FOR ROTATIONS: (axis n, $\|n\| = 1$)

$$R_n(\theta) = I + \sin\theta\widehat{u} + (1 - \cos\theta)\widehat{u}^2$$

This useful formula comes from the formulation of a rotation as the exponential of an antisymmetric matrix:

See derivation in Jean Gallier's book.

$$R_n(\theta) = e^{\theta\widehat{u}} = I + \frac{1}{2!}\theta\widehat{u} + \frac{1}{3!}\theta^2\widehat{u}^2 + \dots$$

EXAMPLE: $u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \frac{1}{\sqrt{14}}, \quad \theta = \frac{\pi}{6}$

$$R_n\left(\tfrac{\pi}{6}\right) = I + \sin\tfrac{\pi}{6}\tfrac{1}{\sqrt{14}} \begin{bmatrix} 0 & -3 & 2 \\ 3 & 0 & -1 \\ -2 & 1 & 0 \end{bmatrix} + (1 - \cos^2\tfrac{\pi}{6}) \begin{bmatrix} 0 & -3 & 2 \\ 3 & 0 & -1 \\ -2 & 1 & 0 \end{bmatrix}^2$$

$$R_n(\theta)p = p + \sin\theta(n \times p) + (1 - \cos\theta(n \times (n \times p)))$$

We now apply this result to a rotation of π around T:

$$\begin{aligned} R_T(\pi)p &= p + (1 - \cos\pi)(T \times (T \times p)) \\ &= p + (T^T pT - T^T Tp) \end{aligned}$$

$a \times (b \times c) = a^T cb - a^T bc$

End of derivation in handout.

Obtaining a valid E-matrix

Recall that our estimation algorithm is the following:

- Estimate E by performing SVD on the following system, obtained by stacking 8 equations coming from 8 pairs of matching points:

$$\underbrace{\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}}_{8\times 9}\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}_{9\times 1} = 0,$$

  with $E = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} \in \mathbb{R}^{3\times 3}$

- Recover $(\widehat{T}, R)$ from E.

Our problem is that the **output E' of the SVD is not necessarily an acceptable essential matrix**: it does not necessarily verify $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$. For an arbitrary E' with $\sigma'_1 \neq \sigma'_2$ and $\sigma'_3 \neq 0$, how can we find the "closest" matrix E with $\sigma_1 = \sigma_2$ and $\sigma_3 = 0$?

In other words we want to project E' on the space of essential matrices, for the Frobenius norm.

We want to find $\text{argmin}_E \|E - E'\|_F$ with E essential. We can also formulate the cost as follows:

$$\left\| U \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T - U' \begin{bmatrix} \sigma'_1 & 0 & 0 \\ 0 & \sigma'_2 & 0 \\ 0 & 0 & \sigma'_3 \end{bmatrix} V'^T \right\|_F^2$$

We use the following algebra results:

1. $\|A - B\|_F^2 = \text{tr}\left((A-B)^T(A-B)\right) = \text{tr}\left(A^TA\right) + \text{tr}\left(B^TB\right) - \text{tr}\left(A^TB\right) - \text{tr}\left(B^TA\right)$
2. For Q orthogonal, $\|QAQ^T\|_F = \|A\|_F$

We have the following:

$$\left\| \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} - \underbrace{U^TU'}_{P \text{ orthogonal}} \begin{bmatrix} \sigma'_1 & 0 & 0 \\ 0 & \sigma'_2 & 0 \\ 0 & 0 & \sigma'_3 \end{bmatrix} \underbrace{V'^TV}_{Q \text{ orthogonal}} \right\|_F^2$$

$$= \text{tr}\left(\begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix}^2\right) + \text{tr}\left(\begin{bmatrix} \sigma'_1 & 0 & 0 \\ 0 & \sigma'_2 & 0 \\ 0 & 0 & \sigma'_3 \end{bmatrix}^2\right) \underbrace{- \text{tr}\left(\left(\begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} P \begin{bmatrix} \sigma'_1 & 0 & 0 \\ 0 & \sigma'_2 & 0 \\ 0 & 0 & \sigma'_3 \end{bmatrix} Q\right)\right)}_{(1)}$$

$$(1) = \sigma(\sigma'_1(p_{11}q_{11} + p_{12}q_{12}) + \sigma'_2(p_{21}q_{21} + p_{22}q_{22}))$$

because of the minus sign, (1) has to be maximized. P and Q are orthogonal.

$$\begin{aligned} p_{11}q_{11} + p_{12}q_{12} &\leq 1 \\ p_{21}q_{21} + p_{22}q_{22} &\leq 1 \\ \sigma, \sigma'_1, \sigma'_2 &> 0 \end{aligned}$$

Therefore we can rewrite the cost to minimize as follows:

$$\begin{aligned}&\text{tr}\begin{bmatrix}\sigma^2 & 0 & 0\\ 0 & \sigma^2 & 0\\ 0 & 0 & 0\end{bmatrix}+\text{tr}\begin{bmatrix}\sigma_1'^2 & 0 & 0\\ 0 & \sigma_2'^2 & 0\\ 0 & 0 & \sigma_3'^2\end{bmatrix}-2(\sigma\sigma_1'+\sigma\sigma_2')\\ =&2\sigma^2+\sigma_1'+\sigma_2'+\sigma_3'-2\sigma\sigma_1'-2\sigma\sigma_2'\\ =&(\sigma-\sigma_1')^2+(\sigma-\sigma_2')^2+\sigma_3'^2\end{aligned}$$

This expression is convex in σ and has to be minimized, therefore we obtain the solution by setting the derivative to 0:

$$\frac{\partial}{\partial\sigma}=0\Rightarrow\sigma=\frac{\sigma_1'+\sigma_2'}{2}$$

The 8-point algorithm: summary

We can now (at last!) write the complete algorithm for camera motion estimation from eight pairs of matching points:

1. Build the homogeneous linear system by stacking epipolar constraints $p_{1i}^T(T\times Rp_{2i})=0,\ i=1,\ldots,8$:

$$\underbrace{\begin{bmatrix}\vdots\\ (p_{1i}\otimes p_{2i})^T\\ \vdots\end{bmatrix}}_{A\ (8\times 9)}\begin{bmatrix}e_1'\\ e_2'\\ e_3'\end{bmatrix}$$

Recall the Hadamard product: $p_1\otimes p_2=\begin{bmatrix}p_{2x}p_1\\ p_{2y}p_1\\ p_{2z}p_1\end{bmatrix}\in\mathbb{R}^9$

2. Let $\begin{bmatrix}e_1'\\ e_2'\\ e_3'\end{bmatrix}$ be the nullspace of A (if $\sigma_8\approx 0$ give up)
3. $\begin{bmatrix}e_1' & e_2' & e_3'\end{bmatrix}=U\text{diag}\,(\sigma_1',s_2',\sigma_3')V^T$. Then use the following estimate of the essential matrix:

$$E=U\text{diag}\left(\frac{\sigma_1'+\sigma_2'}{2},\frac{\sigma_1'+\sigma_2'}{2},0\right)V^T$$

4. $T=\pm u_3\rightarrow R=UR_{Z,\pi/2}V^T$ or $R=R_{T,\pi}R$
5. Try all pairs (T,R) to check if reconstructed points are **in front** of the cameras $\boxed{\lambda_1 p_1 Rp_2+T}$ (three equations, two unknowns).

We are then left with a triangulation problem: $\lambda_1 p_1=\lambda_2 Rp_2+T\Rightarrow\lambda_1,\lambda_2=?$

Why are there three equations but two unknowns? This is overconstrained because in general two lines in space do not intersect. In our case, we know that they should intersect if the epipolar constraint is satisfied (which means p_1,T,p_2 are coplanar).

But the constraint $p_1^T(T \times Rp_2) = 0$ is subject to noise: the input points of the algorithm are not perfect!

The problem of finding λ_1, λ_2 can be formulated as the following simple linear system:

$$\underset{\substack{3\times 2,\text{ rank 2 because } p_1 \nparallel Rp_2}}{\begin{bmatrix} p_1 & Rp_2 \end{bmatrix}} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = T$$

We can solve the system by hand:

$$\begin{aligned} \lambda_1 p_1 \times Rp_2 &= T \times Rp_2 \\ \lambda_1 (p_1 \times Rp_2)^T (p_1 \times Rp_2) &= (T \times Rp_2)^T (p_1 \times Rp_2) \end{aligned}$$

$$\lambda_1 = \frac{(T \times Rp_2)^T (p_1 \times Rp_2)}{\|p_1 \times Rp_2\|^2} > 0$$

Minimal problems in computer vision

Using the 8-point is a bad idea for RANSAC, because 8 points are not the minimal number of correspondences needed to solve for 5 unknowns. Technically the following system obtained by stacking only **five** epipolar constraints should be enough:

$$\begin{aligned} p_{11}^T (T \times RP_{21}) &= 0 \\ p_{12}^T (T \times RP_{22}) &= 0 \\ &\vdots \\ p_{15}^T (T \times RP_{25}) &= 0 \end{aligned}$$

This defines five equations with five unknowns, but we have **no way to find five variables that we could set as unknowns**.

In the 8-point algorithm, we defined the coefficients of E as the unknowns and therefore needed more equations, but those 8 independent coefficients of E contained redundant information about the real five unknowns that we were interested in: the coefficients of R and T.

The trick is to define a polynomial system. Let's go back to the E-matrix system:

$$\underset{5\times 9}{A} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = 0$$

9×1 solution

$E = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} \in \mathbb{R}3 \times 3$. If we perform an SVD of $A_{5\times 9} = U\Sigma V^T$, we expect by definition that $\sigma_6 = \sigma_7 = \sigma_8 = \sigma_9 = 0$.

Therefore, solutions are in the null-space, of which $\{v_6, v_7, v_8, v_9\}$ is an orthonormal basis. This means that we can write the following decomposition of the vector of essential matrix coefficients:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = x v_6 + y v_7 + z v_8 + w v_9, \qquad x, y, z, w \text{ unknowns } (w = 1)$$

Our goal is now to find x, y, z such that $\begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}$ is an E-matrix.

THEOREM: A real 3×3 matrix E is essential (i.e. $E = \widehat{T}R$,antisym-orthogonal decomposition) **iff**:

$$\boxed{det(E) = 0, \qquad 2EE^T E - \text{tr}\,(EE^T)E = 0}$$

These are 10 equations where we can replace $\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = xv_6 + yv_7 + zv_8 + v_9$, and we obtain 10 equations with three unknowns x, y, z. The equations are **cubic** in x, y, z.

We use a linear algebra trick, the "hidden variable elimination trick":

$$\begin{aligned} & a_0(z)x^3 + a_1(z)x^2y + a_2(z)xy^2 + a_3(z)y^3 \\ +& a_4(z)x^2 + a_5(z)xy + a_6(z)y^2 \\ +& a_7(z)x + a_8(z)y + a_9(z) \\ =& 0 \end{aligned}$$

$$\underset{10\times 10}{\begin{bmatrix} a_0(z) & \dots & \dots & a_9(z) \end{bmatrix}} \begin{bmatrix} x^3 \\ x^2y \\ xy^2 \\ x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{bmatrix}$$

The technique we just used, which consists in precomputing monomials of z and placing them in linear forms, is called **lifting**. It enables them to solve linear equations instead of polynomial ones.

(to be continued)

CIS 580 Spring 2012 - Lecture 25

April 18, 2012

Notes and figures by Matthieu Lecce.

Continuous epipolar constraint and the motion field

Consider P_1 measured at t_1, P_2 at t_2, with $t_2 - t_1$ small.

$$P_1 = RP_2 + T$$

The camera undergoes a motion with linear velocity V and angular velocity Ω.

The velocity of a 3D point P is given by the following equation:

$$\dot{P} = -\Omega \times P - V$$

The 2D projection is given by the following equations:

- Uncalibrated: $p' \sim \begin{bmatrix} K & 0 \end{bmatrix} \begin{bmatrix} P \\ 1 \end{bmatrix}$
- Calibrated: $p \sim \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} P \\ 1 \end{bmatrix}$, i.e. :

$$p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad x = \frac{X}{Z},\ y = \frac{Y}{Z},\ \text{where } P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Our goal is to find $\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$ the motion field . Since $p = \frac{1}{Z}P$, we have the following expression for the derivative of p:

$$\begin{aligned}
\dot{p} =& \frac{Z\dot{P} - \dot{Z}P}{Z^2} \\
=& \frac{1}{Z}\dot{P} - \frac{\dot{Z}}{Z}p \\
=& \frac{1}{Z}(-V - \Omega \times P) - frac\dot{Z}Zp \\
=& -\frac{V}{Z} - \Omega \times p - \frac{\dot{Z}}{Z}p
\end{aligned}$$

Instead we write:

$$\begin{aligned}
\dot{p} =& \frac{1}{Z}\dot{p} - \frac{1}{Z}\dot{Z}\frac{P}{Z} \\
=& \frac{1}{Z}z_0^T p\dot{P} - \frac{1}{Z}z_0^T \dot{P}p \\
=& \frac{1}{Z}z_0 \times (\dot{P} \times p) \\
=& \frac{1}{Z}z_0 \times ((-V - \Omega \times P) \times p) \\
=& \frac{1}{Z}z_0 \times (p \times V) + z_0 \times (p \times (p \times \Omega))
\end{aligned}$$

Parenthesis on visual servoing

- State x, $\dot{x} = f(x,u)$, where u is the control input
- Measurement $y = h(x)$ (image points in our case)

The goal is to *design* a control

$$u = g(y, y_{\text{desired}})$$

so that $y - y_{\text{desired}}$ is minimized. The variation of y is given by the following equation:

$$\dot{y} = \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x}f(x,u).$$

Note that the motion field is different from the optical flow, that we defined by $I_x u + I_y v = 0$

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\dot{p} = \underbrace{\frac{1}{Z}\begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix}}_{\text{translational flow}} + \underbrace{\begin{bmatrix} xy & -(1+x^2) & y \\ (1+y)^2 & -xy & -x \end{bmatrix}\Omega}_{\text{rotational flow independernt of depth!!}}$$

$$\dot{p}_{\text{trans}} = \frac{V_z}{Z}\begin{bmatrix} x - \frac{V_x}{V_z} \\ y - \frac{V_y}{V_z} \end{bmatrix}$$

If Z is known, $\dot{p}$ is linear in V and Ω!

1. By intersecting the lines spanned by $\dot{p}_{\text{trans}}$, we can obtain the epipole[1]
2. What is the length of $\dot{p}$?

$$\|\dot{p}_{\text{trans}}\| = \frac{V_z}{Z}\left\| p - \begin{bmatrix} V_x/V_z \\ V_y/V_z \end{bmatrix} \right\|$$

3. The time to collision (sometimes studied in biology) is given by $\frac{Z}{V_z}$:

$$\frac{V_z}{Z} = \frac{\|\dot{p}_{\text{trans}}\|}{\left\| p - \begin{bmatrix} V_x/V_z \\ V_y/V_z \end{bmatrix} \right\|}$$

[1] F.O.E: focus of expansion

Recovering the motion field from image measurements

Let's say we measure the image velocities of five points. The unknowns can be:

- V, Ω and the depths of four of the points (one depth is fixed)
- V, Ω with $\|V\| = 1$, and the depths of all five points

We solve the following equations for $V, \Omega, Z_{i=1,\ldots,n}$:

$$\begin{aligned} \dot{p} &= \frac{1}{Z}\underset{2\times 1}{B(V)} + \underset{2\times 3}{A}\Omega \\ &= \begin{bmatrix} \underset{2\times 4}{B(V)} & A \end{bmatrix}\begin{bmatrix} 1/Z \\ \Omega \end{bmatrix} \end{aligned}$$

For n points, we obtain the following system:

$$\underset{2n\times 1}{\begin{bmatrix} \vdots \\ \dot{p}_i \\ \vdots \end{bmatrix}} = \underset{2n\times(n+3)}{\begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \ldots & B_i(V) & \ldots & A_1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}} \underset{(n+3)\times 1}{\begin{bmatrix} \vdots \\ \vdots \\ 1/Z_i \\ \vdots \\ \Omega \end{bmatrix}}$$

This is an overconstrained linear system $d = C(V)m$ solved by minimizing $\|d - Cm\|$ with the solution $m = C^{\dagger}d$. But C is still a function of V! Therefore we need to add the following minimization:

$$\min_V \|d - C(V)C^{\dagger}(V)d\|$$

D. Heeger and A. Jepson, 1991

Continuous epipolar constraint

How does our epipolar constraint $p_1^T(T \times Rp_2) = 0$ translate here? We obtain an equation involving $p, \dot{p}, V, \Omega$ without Z:

$$\boxed{(V \times p)^T \quad \underbrace{(\dot{p} + \Omega \times p)}_{\text{translational flow in spherical}} \quad = 0}$$

Spherical eye $\dot{p} = -\frac{1}{\lambda}(p \times (V \times p)) - \Omega \times p$

$$(p \times \dot{p})^T V + p^T \Big(\underbrace{\frac{V\Omega^T + \Omega V^T}{2}}_{3\times 3 \text{ symmetric}} - \Omega^T V I \Big) p = 0$$

$$\begin{bmatrix} -\dot{y} & \dot{x} & x\dot{y} - y\dot{x} & 1 & x & y & x^2 & xy & y^2 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \\ -V_x\Omega_x - V_y\Omega_y \\ V_x\Omega_z + V_z\Omega_x \\ V_z\Omega_y + V_y\Omega_z \\ -V_z\Omega_z - V_y\Omega_y \\ V_x\Omega_y + V_y\Omega_x \\ -V_z\Omega_z - V_x\Omega_x \end{bmatrix} = 0$$

This 9×1 vector is the "*continuous essential parameter*" vector:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = V$$

$$\begin{bmatrix} e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{bmatrix} = \begin{bmatrix} -e_1 & -e_2 & 0 \\ e_3 & 0 & e_1 \\ 0 & e_3 & e_2 \\ 0 & -e_2 & e_3 \\ e_2 & e_1 & 0 \\ -e_1 & 0 & -e_3 \end{bmatrix}$$

This linear system has to be solvable w.r.t. Ω: given the system $Ax = b$, the necessary and sufficient condition for solvability is for $\begin{bmatrix} A & b \end{bmatrix}$ to be linearly dependent.

(Condition on determinant)

Note that, we assumed $\Delta P = \dot{P}$, and this assumption does not work for high velocities or low framerate.

$$\dot{P} = -\Omega \times P - V$$