Computer Vision

Mario Dohr

October 20, 2022

1 Introduction

The goal of computer vision is to extract information from images, in contrast to computer graphics, where one wants to create images from a description. To extract information from an image the first step is to recognize the shapes in the image.

1.1 Shape from shading

Try to construct the texture of an object from a set of pictures of the object under different illuminations (Beleuchtungen). The camera is fix relative to the object.

Lambertian Shading Model This model states, that the shading c of a surface point is proportional to the cosine of the angel between the normal n and the direction of the light source l.

$$c \propto \cos(\theta) = |n \cdot l|$$

The Lambertian shading model assumes that the surface of the object is **perfectly diffused**, that is, that the light that comes in is reflected equally in each direction.

We now assume, we have n pictures in the set. For each of the pictures we have the following equation:

$$I(x, y) = k \cdot \rho(x, y) \cdot N(x, y) \cdot L(x, y)$$

where

- $I(x, y) = I_k(x, y)$ is the brightness of a pixel in picture $k = \{1, \ldots, n\}$
- k depends on the camera, can be thought as a bias or distortion. $\rho(x, y)$ is the texture of a pixel (the real brightness). So $k \cdot \rho(x, y)$ is the brightness of the pixel produces by the camera.
- the normal at (x, y)
- the position of the light source at (x, y)

The unknown in this equations are ρ and N, so we group unknowns and known things.

- $g(x,y) = \rho(x,y) \cdot N(x,y)$. $g(x,y)$ is a $3-d$ vector and does not depend on a specific image.
- $V(x,y) = k \cdot L(x,y)$ is a $3-d$ vector and different for each image.
- $I(x,y) = g(x,y) \cdot V(x,y)$ is a scalar and different for each image.

insert detailed description

2 Spatial and frequency domain processing

2.1 Image Analysis

Goal of image analysis is to identify distinct image features. Features should describe the main parts of an image. But features also should be robust, that is, invariant to translation, scaling, rotation and lightning.

2.1.1 Linear Filtering and convolution

One technique to extract features is linear filtering. When a filter is applied to an image F the result is a new image R. A pixel in R is the weighted sum of its neighbourhood in F. A filter is represented as a matrix H, referred as kernel. The process of applying the H to a image is known as **convolution**. The entries of the matrix H are referred as weights of the filter. The weights of the kernel determines the kind of operation.

Examples

- Identity $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
- local average, blurring $\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$
- sharpening $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

2.1.2 Gaussian Kernels and smoothing

For blurring or smoothing you use a Gaussian kernel and not the local average, because the spread of the light of a defocused point is Gaussian.

$$G_\sigma = \frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2})$$

$$H_{i,j} = \frac{1}{2\pi\sigma}\exp(-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2})$$

Properties of a Gaussian

- Removes hight frequency components from the image
- Convolution with itself is another Gaussian. Convolution two times with a small kernel is the same as convolution one time with larger kernel. (Check)
- Separable. It factorizes into product of two Gaussian kernels.

Convolution rules for shift invariant linear systems (SILS)

- Superimposition: the sum of two filtered image is equal to the filtering of the sum of the two kernels. $H_1 \otimes R + H_2 \otimes R = (H_1 + H_2) \otimes R$
- Scaling: $(kH) \otimes R = k(H \otimes R)$
- Symmetric: $H_1 \otimes (H_2 \otimes R)) = H_2 \otimes (H_1 \otimes R)$
- Associative: $H_1 \otimes (H_2 \otimes R) = (H_1 \otimes H_2) \otimes R$

Example: Unsharp masking Apply the rules above to construct a filter for removing of blurring. We have a image F and a Gaussiang filter g:

$$F + \alpha(F - F * g) =$$

$$F * e + \alpha(F * e - F * g) =$$
$$F * e + \alpha F * e - \alpha F * g =$$
$$(1 + \alpha)F * e - \alpha F * g =$$
$$F * ((1 + \alpha)e - \alpha g)$$

The result is called a Laplacian filter.

Example: partial derivatives How to detect edges or borders of objects. At the border of two objects the brightness of the pixel changes. One can measure this by calculating the gradient.

$$\frac{\partial f}{\partial x} = f(x + 1, y) - f(x - 1, y)$$

A kernel for computing partial derivatives in x-direction is

$$H = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

This kernel detects vertical edges.