Image Processing and Computer Vision - Notes

Dom Hutchinson

October 14, 2019

Contents

1 Image Acquistion

Proposition 1.1 - *Common Challenges with Image Acquistion*
Below are some common challeges that are faced/produced by image acquistion

Viewpoint Variation	Several images may be taken of the same object but will vary the angle
Illumination	Images may be taken in low/high light
Occlusion	Object may be partly obscured
Scale	Objects may look vary different when placed next to other objects due to their relative scale
Deformation	Objects may have slight variations on the perfect form
Background Clutter	Lots happening behind an object may work to obscure it
Object Intra-Class Variation	Some objects in the same class can vary a lot in shape (*e.g.* chairs)
Local Ambiguity	Certain regions of an image can be missinturpred without the rest of the scene being accounted for
World Behind the Image	Depth may need to be accounted for to make sense of an image.

Definition 1.1 - *Dirac Delta-Function, δ*
The *Dirac Delta-Function* is used to map continuous distributions to discrete distributions by sampling at particular intervals. Intuitively

$$\delta(t)=\begin{cases}1 & ,t=0\\ 0 & ,t\neq 0\end{cases} \implies \delta(t-\alpha)=\begin{cases}1 & ,t=\alpha\\ 0 & ,t\neq \alpha\end{cases}$$

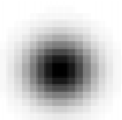
Definition 1.2 - *Sifting Property*
We can apply the *Dirac Delta-Function* to a function to sample a particular value

$$\int_{-\infty}^{\infty} f(t)\delta(t)dt=f(0) \implies \int_{-\infty}^{\infty} f(t)\delta(t-\alpha)dt=f(\alpha)$$

This can be applied to 2D objects (such as images) as

$$\int_{-\infty}^{\infty} f(a,b)\delta(a-x,b-y)dadb=f(x,y)$$

Definition 1.3 - *Point Spread-Function*
A *Point Spread-Function* is applied after sampling an image. It takes the value of a pixel & transforms pixels around it using this value in some way.

e.g.  (Should be a white dot on black background but ink).

2 Image Representation

Definition 2.1 - *Colour Space*
Colour Space are different techniques for representing colours. These are generally made up of 3D vectors.

Colour Space	Vector Description
RGB	(Red $\in[0,255]$,Green $\in[0,255]$,Blue $\in[0,255]$)
HSI	(Hue $\in[0,360)$,Saturation $\in[0,1]$,Intensity $\in[0,1]$) Hue gives the colour in degrees
YUV	(Brightness $\in[0,255]$,Blue Projection $\in[0,255]$,Red Projection $\in[0,255]$)
La*b*	(Luminance $\in[0,100]$, Red/Green $\in\{-a,+b\}$, Blue/Yellow $\in\{+b,-b\}$

Remark 2.1 - *Representing Video*
To represent video each fixel is given a third parameter, *time* so we now have

$$f(x, y, t) \mapsto (R, G, B)$$

or any other *Colour Space*.

Definition 2.2 - *Quantisation*
Quantisation is representing a continuous single channel function with discrete single channel function that groups the continuous values into a set number of levels.

Example 2.1 - *Quantisation*

16 levels 6 levels 2 levels

Definition 2.3 - *Aliasing*
Aliasing is the result of sparse sampling since single pixels represent to large an area to get any detail out of it.

Example 2.2 - *Aliasing*

256 x256 64x64 32x32

Definition 2.4 - *Anti-Aliasing*
Anti-Aliasing is the process for avoiding *Aliasing*. This can be achieved by using a sampling rate which is a critical limit defined by the *Shannon-Nyquist Theorem*.

Theorem 2.1 - *Shannon-Nyquist Theorem*
An analogue signal with maximum frequency xHz may be completley reconstructed if regular samples are taken with frequency $2x$Hz.

Definition 2.5 - *Convolution*
Convolution is an operation which takes two functions & produces a third which describes how the shape of one of the two functions is changed by the other.
For functions f & g

$$(f * g)(x) := \int_{-\infty}^{\infty} f(x - t)h(x)\partial t$$

N.B. $*$ is the symbol for convolution.

Remark 2.2 - *Convolution in Image Representation*
Suppose you have a system, represented by kernel $g(x)$, & an input signal, represented by $f(x)$. THen $f * g(x)$ describes the effect of the system on the input signal. The resulting image is called the *Response of f to the kernel h.*

Proposition 2.1 - *2D Discrete Convolution*
Since images are represnted by discrete 2D functions $f : \mathbb{N} \times \mathbb{N} \to (\mathbb{N} \times \mathbb{N} \times \mathbb{N})$ it is pertinent to understand *2D Discrete Convolution.*

$$h(x, y) = \sum_{i \in I} \sum_{j \in J} f(x - i, y - j) g(i, j)$$

Often the kernel, $g(x, y)$, has negative indices so the pixel being acted upto is equivalent to the middle pixel in the matrix representation of $g(x, y)$.
N.B. A convolution whose kernel is symmetric on 180 degree rotation is called a *Correlation.*

Example 2.3 - *2D Discrete Convolution*
Below is a representation of a grayscale image, $f(x, y)$, on the left & a kernel $g(x, y)$ on the right.

		y − 1	*y*	*y+1*	
x − 1		**43**	**12**	**61**	
x		**44**	**45**	**60**	
x+1		**43**	**50**	**61**	

	-1	*0*	*1*
-1	**-1**	**0**	**1**
0	**-2**	**0**	**2**
1	**-1**	**0**	**1**

$(f * g)(x, y) = f(x + 1, y + 1)g(-1, -1) + f(x + 1, y)g(0 - 1, 0) + \cdots + f(x - 1, y - 1)g(1, 1) = -68.$

Example 2.4 - *Kernels*
Kernels an be defined with specific outcomes in mind.

Operation	**Matrix**
Identity	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
Edge Detection	$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix}$ $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ $\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$
Sharpen	$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$
Box Blur	$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$
Gaussian Blur 3×3	$\frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

Gaussian Blur 5×5	$\frac{1}{256}\begin{pmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 5 & 4 & 1\end{pmatrix}$
Unsharp Masking 5×5	$\frac{-1}{256}\begin{pmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & -476 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{pmatrix}$

3 Frequency Domains & Image Transforms

Definition 3.1 - *Image Transform*
An *Image Transform* is deriving a new representation of the input data by encoding the image using another parameter space (*e.g.* Fourier, DCT, Wavelet, etc.).

Remark 3.1 - *Purpose of Image Transforms*
Image Transforms can be used in

i) Image Filtering;

ii) Image Compression;

iii) Feature Extraction;

iv) etc.

Definition 3.2 - *Properties of a Signal*
A *Signal* is a sinusoidal function over continuous time. They have the following properties

i) Frequency - Number of cycles per second, Hz;

ii) Period - Length of a cycle, s;

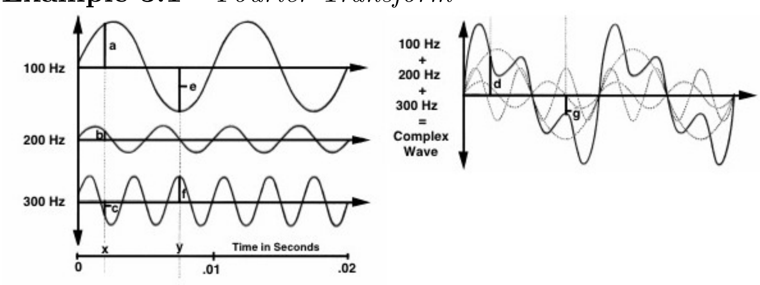
iii) Amplitude - Peak intensity of the signal;

iv) Phase - The shift of the trig wave from its default position, π.

Theorem 3.1 - *Fourier's Theorem*
All periodic functions over continous time can be expressed as a sum of sin &cos terms, each with their own amplitude & shift.

$$f(t) = a_0 \sin(t + \theta_0) + a_1 \cos(t + \theta_1) + \dots$$

Example 3.1 - *Fourier Transform*



Proposition 3.1 - *Frequency in Images*
Frequency in Images is measured as the rate of change in intensity along a given line on the image.

Remark 3.2 - *Fourier Transform on Frequency in Images*
If we read the intensity values along a single row or column we can produce a sinusoidal wave which generalises the distribution & then perform a *Fourier Transform.*

Definition 3.3 - *2D Discrete-Space Fourier Transform*
Images can be considered as 2-Dimensional discrete space. Let $f(x,y)$ be the intensity of the pixel at position (x,y). 2D Discrete-Space Fourier Transforms have two variables: $u \in [-\pi,\pi)$ for the vertical frequency; and, $v \in [-\pi,\pi)$ for the horizontal frequency.

$$\begin{aligned}\underbrace{F(u,v)}_{\text{Fourier Space}} &= \sum_{y=0}^{m-1}\sum_{x=0}^{n-1} f(x,y)e^{i(ux+vy)} \\ &= \sum_{y=0}^{m-1}\sum_{x=0}^{n-1} f(x,y)\left[\cos(ux+vy)+\sin(ux+vy)\right]\end{aligned}$$

N.B. $F(u,v)$ is a complex number.

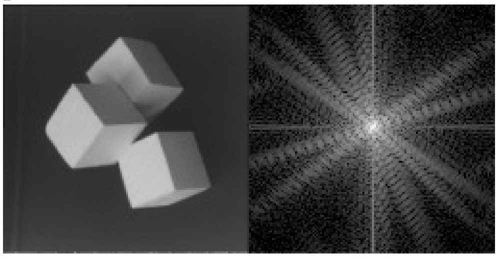
Proposition 3.2 - *Interpretations of 2D Fourier Transform*
Since $F(u,v)$ is a complex number we cannot plot it exactly. Thus we consider

i) Magnitude, $|F(u,v)| := \sqrt{F_\text{r}(u,v)^2 + F_\text{i}(u,v)^2}$, and
ii) Phase Angles, $\theta(u,v) := \tan^{-1}\left(\frac{F_\text{i}(u,v)}{F_\text{r}(u,v)}\right)$

Remark 3.3 - *Expressing $F(u,v)$ in Polar Cordinates*

$$F(u,v) = |F(u,v)|e^{i\theta(u,v)}$$

Remark 3.4 - *Plotting Magnitude, $|F(u,v)|$*
On the left we have a gray scale image & on the right we have the magnitude of a fourier transform on this image. On the right hand image the y-axis is $u \in [-\pi,\pi)$ and the x-axis is $v \in [-\pi,\pi)$. We see lots of straight lines since a linear transformation on $F(u,v) = F(au,av)\ \forall\ a \in \mathbb{R}$. Each line can be interpreted as the frequency of intensity for lines in the left hand image which are parallel to it.



Theorem 3.2 - *Convolution Theorem*
Let f be an image, g be a kernel, F be the result of a fourier transform on f and G be a kernel. Then

$$h = f * g \iff H = FG$$

Proof 3.1 - *Convolution Theorem*

$$\begin{array}{rcl} h(x) & = & f(x) * g(x) \\ & = & \sum_y f(x-y)g(y) \\ H(u) & = & \sum_x \left(\sum_y f(x-y)g(y)\right) e^{iux} \\ & = & \sum_x g(y) \sum_x f(x-y)e^{iux} \\ & = & \sum_y g(y) \left(F(u)e^{iux}\right) \\ & = & \sum_y g(y)e^{iuy}F(u) \\ & = & G(u) \cdot F(u) \\ & = & F(u) \cdot G(u) \end{array}$$

Definition 3.4 - *Butterworth's Low Pass Filter*
Butterworth's Low Pass Filter is a *Signal Processing Filter* designed to have a frequency response which is as flat as possible. It appears to soften an image

$$H(u,v) = \frac{1}{1 + \left(\frac{r(u,v)}{r_0}\right)^{2n}} \text{ of order } n$$

Definition 3.5 - *Butterworth's High Pass Filter*
Butterworth's High Pass Filter is a *Signal Processing Filter* designed to have a frequency response which is as flat as possible. It appears to sharpen an image

$$H(u,v) = \frac{1}{1 + \left(\frac{r_0}{r(u,v)}\right)^{2n}} \text{ of order } n$$

0 Reference

0.1 Definitions

Definition 0.1 - *Kernel*
A *Kernel* is a small matrix used in convolution. Typically 3×3 or 5×5. *Kernels* can be defined for blurring, sharpening, embossing, edge detection & more *N.B.* This definition only applies to image processing & is different from the definition in linear algebra.