Computer Vision Systems Programming VO

A Recap of Image Processing

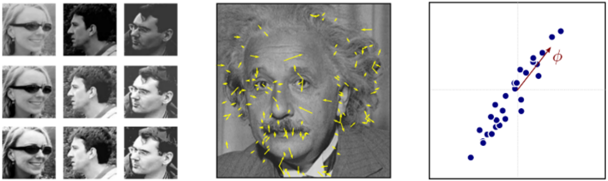
Christopher Pramerdorfer

Computer Vision Lab, Vienna University of Technology

Topics

A brief recap of Image Processing (IP)

- Assuming you are already familiar with IP
- Focus on methods that are widely used in practice



Images from Prince 2012

Relation of IP and CV

IP encompasses operations that

- Take images as input
- Produce images or representations (e.g. descriptors)

We regard IP as preprocessing for CV

- IP has great influence on CV performance

Relation of IP and CV

CV is all about

- Infering some world state $\mathbf{w}$
- From measurements $\mathbf{x}$

We use IP to obtain a suitable $\mathbf{x}$ from images

Suitable often means

- Distinctive features
- That are robust and covariant

Contrast Normalization

Reduce variation due to contrast and intensity changes

We cover two techniques

- Whitening
- Histogram equalization

Images from Prince 2012

Contrast Normalization
Whitening

Transform pixel values so that

- Their mean is zero
- Their variance is one

Images from Prince 2012

Contrast Normalization

Whitening – Matlab Implementation

```matlab
img = single(rgb2gray(imread('image.png'))); % load
m = mean(img(:)); % compute mean
s = std(img(:)); % compute standard deviation
whitened = (img - m) / s; % normalize
```

Contrast Normalization
Histogram equalization

Transform pixel values so that distribution is “flat”

- Cumulative histogram linear over value range

Images from Prince 2012

Contrast Normalization

Histogram Equalization – C++ Implementation

```cpp
// cv = OpenCV namespace
cv::Mat img, equalized; // storage
img = cv::imread("image.png", cv::IMREAD_GRAYSCALE); // load
cv::equalizeHist(img, equalized); // normalize
```

Noise Reduction and Change Detection

Reduce image noise

Locate intensity changes

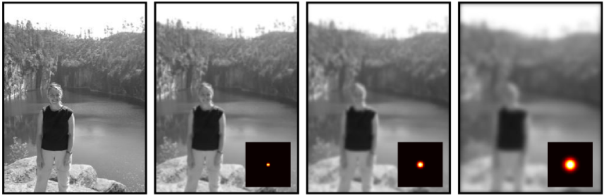
Often accomplished via **linear filtering**

- Pixel values linear combination of neighbor values
- Computed via convolution (or correlation)

$$f'(x, y) = \sum_{i,j} f(x - i, y - j)h(i, j)$$

Use a 2D Gaussian as kernel h:

$$h(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right)$$



Images from Prince 2012

Noise Reduction and Change Detection
Noise Reduction via Blurring – Matlab Implementation

```
img = rgb2gray(imread('image.png')); % load
h = fspecial('gaussian', [3 3], 0.5); % create Gaussian kernel
filtered = imfilter(img, h); % filter
```

Use a Laplacian of Gaussian (LoG) filter as kernel h

- Gaussian for noise reduction
- Laplacian approximates $\nabla^2 = f_{xx} + f_{yy}$

LoG filters respond to intensity changes

- Regardless of direction
- At a frequency defined by σ of Gaussian

Substrate for SIFT interest points


Images from Prince 2012

```
img = rgb2gray(imread('image.png')); % load
h = fspecial('log', [3 3], 0.5); % create LoG kernel
filtered = imfilter(img, h); % filter
```

Use a Gabor filter as kernel h, which consists of

- A Gaussian for noise reduction
- A Sinusoid for change detection

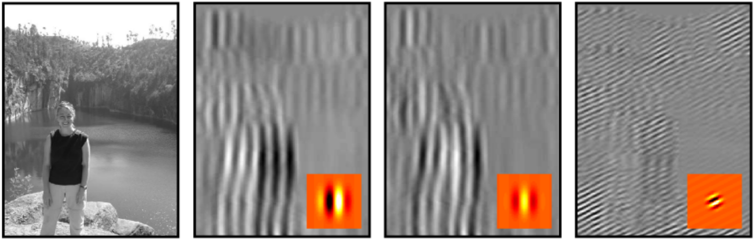
Gabor filters respond to intensity changes at a

- Phase and orientation defined by the Sinusoid
- Frequency defined by the Gaussian and Sinusoid

Substrate for object recognition and scene understanding

Noise Reduction and Change Detection

Change Detection via Gabor Filtering



Images from Prince 2012

```
cv::Mat img, gabor; // storage
img = cv::imread("image.png", cv::IMREAD_GRAYSCALE); // load
h = cv::getGaborKernel(...); // create Gabor kernel
cv::filter2D(img, gabor, CV_32F, h); // filter
```

Interest Point Detection

Interest points are image features that

- Are local (precise location, little spatial extent)
- Can be detected reliably in multiple images of same object

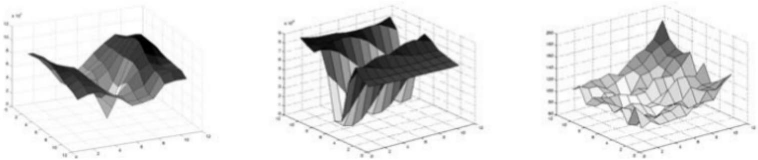
Which implies that they are

- Invariant to image transformations
- Robust with respect to noise

Corners characterized by intensity change in multiple directions

Harris corner detector exploits this by

- Checking gradient distribution in local neighborhood
- Corner: gradient distribution has two large eigenvalues

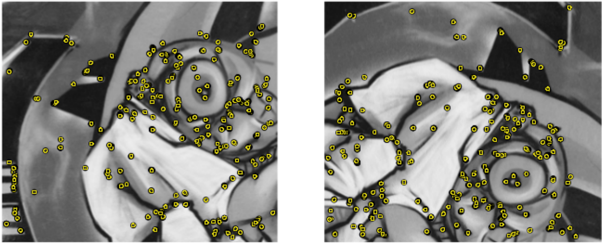


Images from Szeliski 2010

Interest Point Detection
Harris Corner Detector

Harris interest points

- Are invariant to translation and rotation
- Stable under varying lighting conditions



Images from Tuytelaars and Mikolajczyk 2008

```
img = rgb2gray(imread('image.png')); % load
corners = corner(img, 'Harris', maxNum); % detect corners
```

Scale invariant blob detector

- A blob is an image region with similar intensity

Blob detection accomplished via LoG filtering

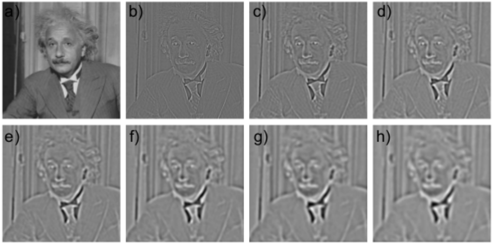
- LoG filter responds to blobs of size that depends on σ

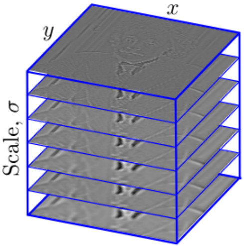
Scale invariance is achieved by

- Applying LoG filter with multiple σ
- Finding local maxima in resulting scale-space

Repeated LoG approximated by Differences of Gaussians (DoGs)

Interest Point Detection
SIFT Detector – Scale Space





Images from Prince 2012

Local maxima are

- Localized to sub-voxel accuracy
- Discarded unless on corners
- Assigned an orientation via gradient histograms

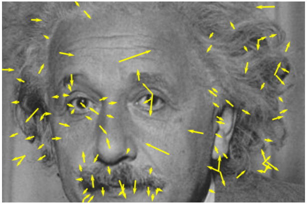


Image from Prince 2012

```
cv::Mat img = cv::imread("image.png", cv::IMREAD_GRAYSCALE);
cv::SiftFeatureDetector det(20, 10); // create detector
std::vector<cv::KeyPoint> kps; // keypoint storage
det.detect(img, kps); // detect keypoints
```

Local Descriptors

Compact representations of contents of an image region

Usually computed at interest point locations

Invariant in conjunction with suitable interest points

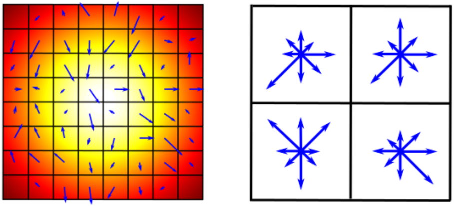
Pool information locally to achieve robustness

- Often accomplished via histograms

Computed from gradient histograms

Usually used together with SIFT interest points

- Compensate for scale, rotation



Images from Prince 2012

SIFT descriptors are

- Invariant to scale and rotation (interest points)
- Invariant to global intensity changes (gradients)
- Robust to small affine transformations (pooling)

SIFT Descriptor – C++ Implementation

```
// img and kps from previous example
cv::Mat descriptors; // storage (n by 128)
cv::SiftDescriptorExtractor ex; // create extractor
ex.compute(img, kps, descriptors); // compute descriptors
```

Dimensionality Reduction

Sometimes desirable to reduce the dimensionality of $\mathbf{x}$

- Makes learning and inference more efficient
- Can improve generalization performance
- Facilitates data visualization

Goal is to find transformation from $\mathbf{x}$ to $\mathbf{v}$

- With $v = \dim(\mathbf{v}) < x = \dim(\mathbf{x})$
- That minimizes the information loss

Principal Component Analysis (PCA) yields

- The orthogonal transformation ϕ to a v-dimensional subspace
- That minimizes $\sum_i \|\mathbf{x}_i - \phi^{-1}\mathbf{v}_i\|_2$ (assuming zero mean)

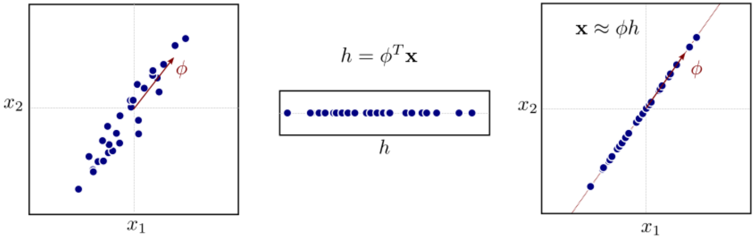
Accomplished if $\phi = [\phi_1, \ldots, \phi_v]^T_{v \times x}$

- With ϕ_k being the kth largest eigenvectors of $\mathbf{X}\mathbf{X}^T$
- Where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_I]$

Projects $\mathbf{x}_i$ onto hyperplane

- Spanned by v largest axes of covariance ellipsoid

In this case $h = v$ and $\phi = \phi^{-1} = \phi^T$



Images from Prince 2012

PCA is dependent on the units of x_i

- So standardize (whiten) your data

PCA is unsupervised and linear

- Better methods for supervised case (e.g. LDA)
- More powerful non-linear methods (e.g. autoencoders)
- But generally superfluous with modern learning methods

```matlab
pc = princomp(X'); % compute all principal components (sorted)
phi = c(:,1:v)'; % keep only v "largest" ones and transpose
V = phi * X; % project to lower dimension
Xe = phi' * V % reconstruct X
```

Summary

We utilize IP to obtain $\mathbf{x}$ from an image

- That is for feature extraction

We want $\mathbf{x}$ to be distinctive, invariant, robust, concise

- And we have seen methods to achieve this

We have only scratched the surface of IP

- See literature

Bibliography

Prince, S.J.D. (2012). **Computer Vision: Models Learning and Inference**. Cambridge University Press.

Szeliski, Richard (2010). **Computer vision: algorithms and applications**. Springer.

Tuytelaars, Tinne and Krystian Mikolajczyk (2008). **Local invariant feature detectors: a survey**. Foundations and Trends in Computer Graphics and Vision.