Advanced Computer Vision

Chapter 3

Image processing

Presenter: 林保羅 (Paulo L.)

指導教授: 傅楸善 博士

E-mail: d12922028@ntu.edu.tw

Outline

1 Point Operators

2 Linear Filtering

3 More Neighborhood Operators

4 Fourier Transforms

5 Pyramids and Wavelets

6 Geometric Transformations

Image processing operators

- Image processing operators h generate an image g from another image f.
- Where $\vec{\mathbf{x}} \in \mathbb{N}^2$ is the domain of the image, and $f(\vec{\mathbf{x}})$ is the range of the image

Intensity operator

The operator h aims to modify the pixels' intensities of an image f or a group of images f_k.

$$g(\vec{\mathbf{x}}) = h\left(\{f_k(\vec{\mathbf{x}})\}_{k=0}^{K-1}\right) \quad \text{Processing multiple images}$$

$$g(\vec{\mathbf{x}}) = h(f(\vec{\mathbf{x}})) \quad \text{Processing a single image}$$

Geometric operator

The operator h aims to modify the pixels' locations within the image.

$$g(\vec{\mathbf{x}}) = f(h(\vec{\mathbf{x}}))$$

Point Operators

"Each pixel is modified independently within an image"

Point Operators

Manipulate the intensity of each pixel independently of its neighbors.

- **Pixel transforms:** A predefined function is applied to the intensity of each pixel.
- **Color transforms:** Change the color space to process chromacity and luminosity independently. (RGB $\rightarrow$ YUV)
- **Compositing and matting:** Combining Foreground and Background images regarding their opacity .
- **Histogram equalization:** Generate a pixel transformation function from the image histogram.

Point Operators (cont.)

Pixel transforms

- **Problem:** The objects are poorly visible within the image or their appearance (brightness, contrast) are not good enough.
- **Solution:** Define an intensity mapping function and use it to the modify pixels' intensities in a straighforward manner.

Intensity mapping function

Assuming intensity values in the range $[0, 1]$:

$$m : [0, 1] \rightarrow [0, 1]$$

$$s = m(k)$$

- k: Original intensity
- s: New intensity

Point Operators (cont.)

Gain and bias (Linear transformation)

$$\text{Constant gain, bias: } g(\vec{\mathbf{x}}) = af(\vec{\mathbf{x}}) + b$$

$$\text{Spatially varying gain, bias: } g(\vec{\mathbf{x}}) = a(\vec{\mathbf{x}})f(\vec{\mathbf{x}}) + b(\vec{\mathbf{x}})$$



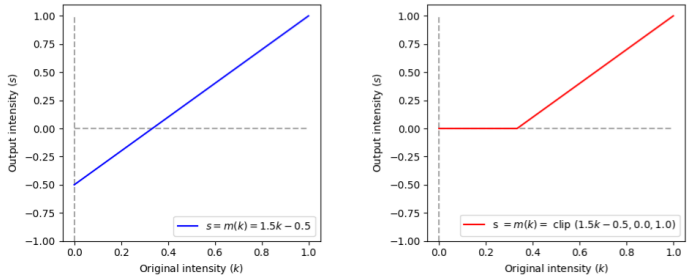
(a) **(b)** **(c)**

Figure 1: *Example of a gain and bias transformation: a) Original image; b) Results of applying constant gain and bias; c) Applying spatially dependent gain and bias (see the lantern).*

Point Operators (cont.)

Constant gain, bias: $g(\vec{\mathbf{x}}) = af(\vec{\mathbf{x}}) + b$

Spatially varying gain, bias: $g(\vec{\mathbf{x}}) = a(\vec{\mathbf{x}})f(\vec{\mathbf{x}}) + b(\vec{\mathbf{x}})$



(a) **(b)**

Figure 2: *Example of a gain and bias transformation: a) Original image; b) Results of applying constant gain and bias; c) Applying spatially dependent gain and bias*

Point Operators (cont.)

Gamma correction (Power transformation)

$$g(\vec{\mathbf{x}}) = f(\vec{\mathbf{x}})^{\gamma}$$

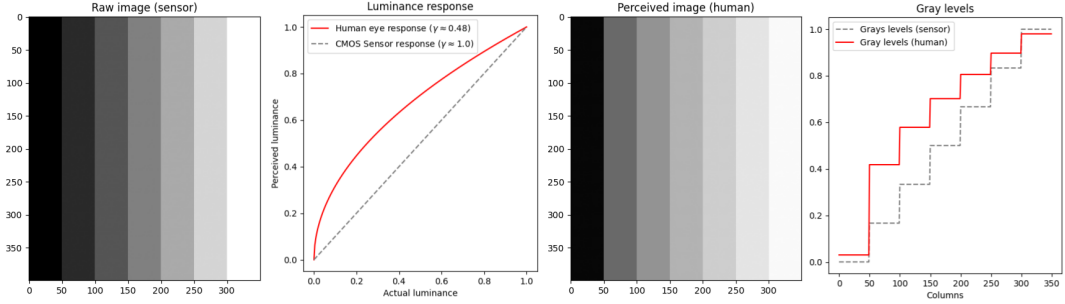


Figure 3: *Image sensor and human eye luminance response.*

Point Operators (cont.)

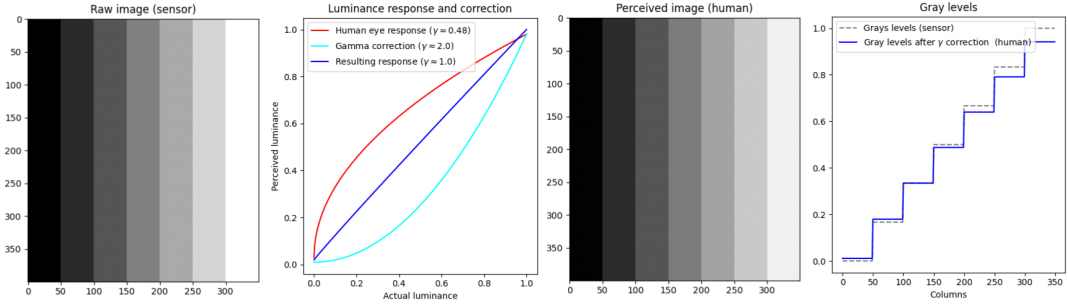
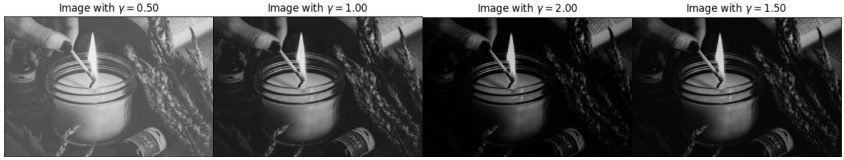


Figure 4: *Application of gamma correction to meet the human eye response.*

Point Operators (cont.)



(a) **(b)** **(c)** **(d)**

Figure 5: *Gamma correction applied to a sample image for different γ values.*

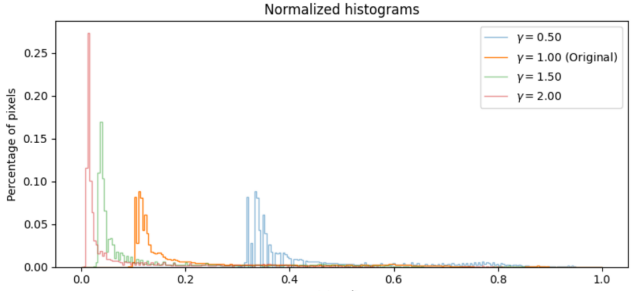


Figure 6: *Normalized histograms of the image after gamma correction.*

Point Operators (cont.)

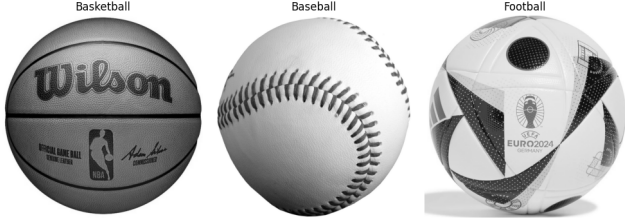
Linear blend (Convex combination of multiple images)

Multiple images

$$g(\vec{\mathbf{x}}) = \sum_k \alpha_k f_k(\vec{\mathbf{x}}) \quad s.t. \quad \sum_k \alpha_k = 1$$

Two images

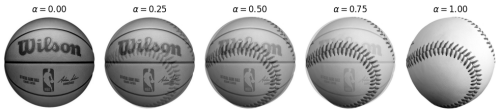
$$g(\vec{\mathbf{x}}) = \alpha f_1(\vec{\mathbf{x}}) + (1-\alpha) f_2(\vec{\mathbf{x}})$$



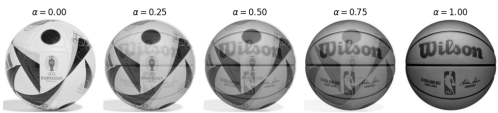
Point Operators (cont.)



(a)



(b)



(c)

Figure 7: *Examples of linear blending between two images.*



Figure 8: *Examples of linear blending between three images.*

Point Operators (cont.)

Color transforms

- In digital images, color information is represented in 3D color space (cartesian and non-cartesian).
- Even though RGB is the mostly used color space, all its channels conveys luminance and chromaticity information. If we want to process luminance and chromaticity separately we should process the image in other color spaces (YUV, CIELAB, HSV).
- Color transforms can be expressed as linear or non-linear transformations.

Point Operators (cont.)

- **Problem:** Applying pixel transforms to an RGB image changes the appearance of the scene in the image.
- **Solution:** Convert the RGB image to another color space, apply pixel transformations, convert back to RGB.



Point Operators (cont.)

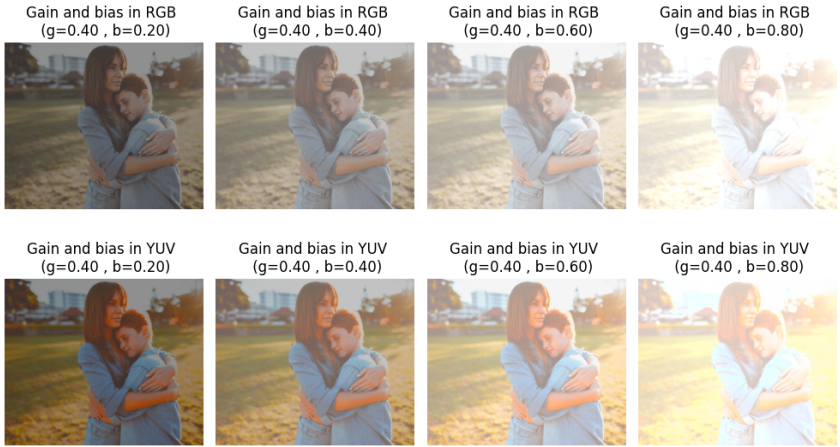


Figure 9: *Gain and bias transformation applied to a color image. Top: Using RGB; Bottom: Using YUV.*

Point Operators (cont.)

Compositing and matting

- **Opacity** (α): Additional channel that is multiplied to an image to attenuate objects in the background. A region with $\alpha = 1.0$ is fully "visible", while $\alpha = 0.0$ makes the region "invisible".
- **Compositing** : Process of inserting foreground regions from a image F into a background image B without visible artifacts
- **Matting** : The process of extracting the α channel from the original image.

$$C = \alpha F + (1 - \alpha)B$$

Point Operators (cont.)

Compositing when the alpha channel is available (RGBA image).



Figure 10: *Example of compositing when opacity information is available: (a) Foreground image; (b) Foreground opacity channel; (c) Background image; (d) Final composed image.*

Point Operators (cont.)

Straightforward matting using a green screen.

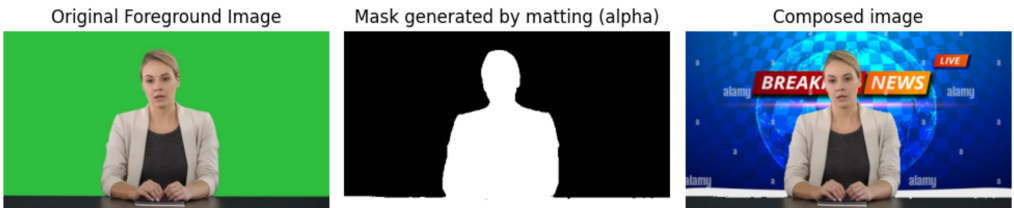


Figure 11: *Example of using a green screen to perform matting from a regular image.*

Point Operators (cont.)

Histogram equalization

Aims to find an intensity mapping function $s = m(k)$ such that the histogram of the resulting image is flat.

- **Problem**: We have a lot of images and don't know which intensity mapping function may be suitable for all of them.
- **Solution**: Generate a intensity mapping function $m(k)$ by processing the image intensity statistics (histogram).

Point Operators (cont.)

Histogram equalization (Global)

1. Source Image $f(\vec{\mathbf{x}})$ with a shape of $W \times H \rightarrow$ Histogram $h(k)$
2. Normalize the histogram to calculate the PDF of each gray value.
   $\tilde{h}(k) = h(k)/HW$
3. PDF $\rightarrow$ CDF: $c(k) = \sum_{u=0}^{k} \tilde{h}(k)$
4. Apply quantization to the CDF to generate an integer look-up-table (LUT)
   $m(k) = c(k)$.
5. Store the LUT.
6. Tranform the pixels' intesities using the LUT

Apply blending (optional): $m(k) = \alpha c(k) + (1 - \alpha)k$

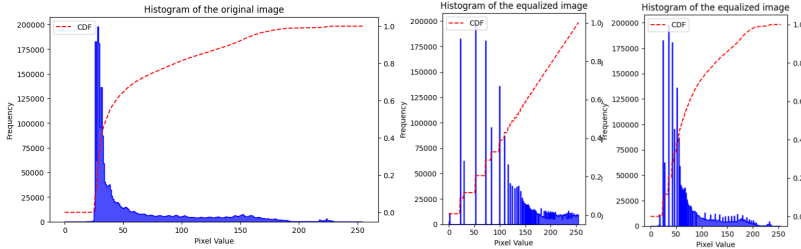
k: Original intensity. $c(k)$: Mapping function obtained from the CDF of the image.
$m(k)$: Final mapping function.

Point Operators (cont.)



(a) (b) (c)

Figure 12: *Application of histogram equalization: (a) Original image; (b) Using conventional histogram equalization; (c) Using histogram equalization and mapping blending.*



(a) (b) (c)

Figure 13: *Histograms of the images in Fig. 12.*

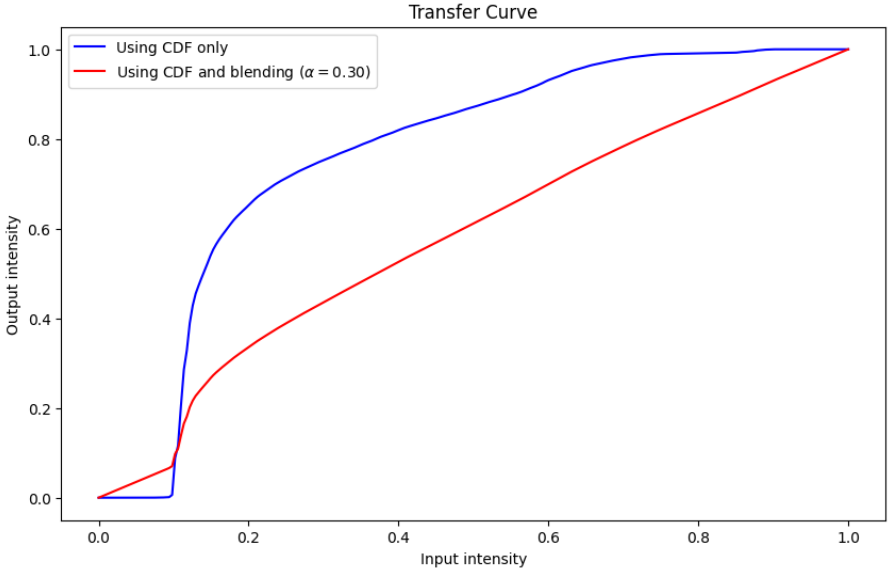


Figure 14: *Intensity mappings functions generated with histogram equalization.*

Point Operators (cont.)

Histogram equalization

- **Problem**: The background of the image is too dark, and after applying histogram equalization the objects in the foreground look quite bright.
- **Solution** : Adopt a local approach to perform histogram equalization.

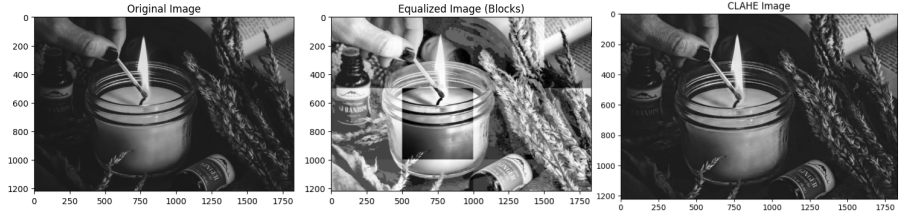


(a) **(b)**

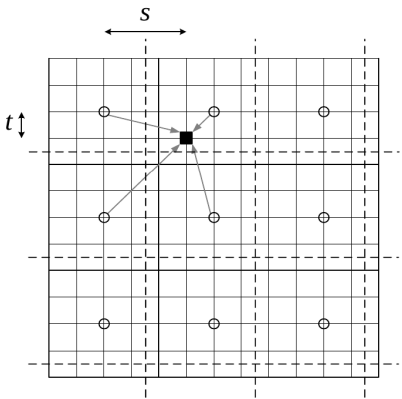
Point Operators (cont.)

Histogram equalization (Local)

- **Naive solution**: Divide the image into blocks and apply histogram equalization within each block
- **Results**: The resulting image exhibits a lot of block-shaped artifacts.
- **Better approach**: Divide the image into blocks, compute histograms and mapping functions for each block, combine the mapping functions of the blocks in a linear way (e.g. AHE, CLAHE).



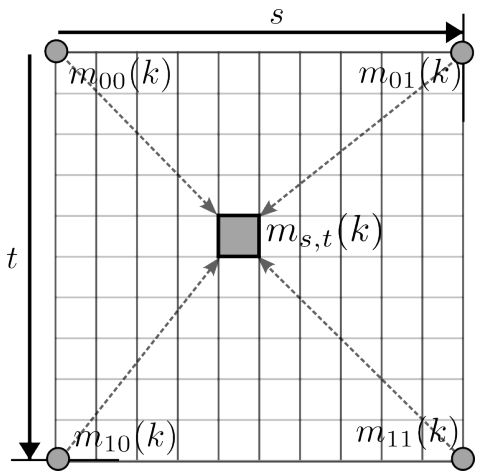
Point Operators (cont.)



One-way approach (block based)

1. Divide the image into blocks and compute $m(k)$ for each block as done for global histogram equalization.
2. Apply bilinear blending to the mapping functions and modify pixel intensities.

Point Operators (cont.)

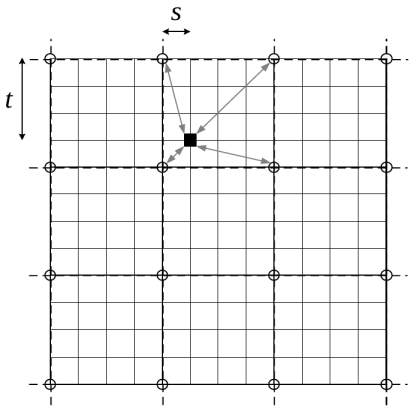


Mapping bilinear blending

$$m_{s,t}(k) = (1-s)(1-t)m_{00}(k) + s(1-t)m_{10}(k) + (1-s)tm_{01}(k) + stm_{11}(k)$$

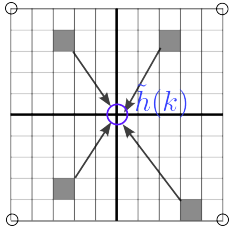
$s, t \in [0,1]$ are relative coordinates within each block.

Point Operators (cont.)



Two-way approach (corner based)

1. Divide the image into blocks and compute normalized histograms $\tilde{h}(k)$ as follows:



2. Compute $m(k)$ for each corner based on $\tilde{h}(k)$.
3. Apply bilinear blending to the mapping functions and modify pixel intensities.

Linear Filtering

"Convolution and correlation: A pixel's intensity is determined by its neighbors"

Linear Filtering

Linear Shift Invariant (LSI) operators

- **Linear operator**: $h\left(\alpha f_a(\vec{x}) + \beta f_b(\vec{x})\right) = \alpha h\left(f_a(\vec{x})\right) + \beta h\left(f_b(\vec{x})\right)$
- **Shift invariant operator**:
  - Shift operation: $g(\vec{x}) = f(\vec{x} + \vec{x_0})$
  - Shift invariant condition: $(h \circ g)(\vec{x}) = (h \circ f)(\vec{x} + \vec{x_0})$
  - An operator h is shift invariant if applying the operator to the shifted image $g(\vec{x})$ gives the same results as applying h to the original image $f(\vec{x})$ and then shifting the resulting image.

Linear Filtering (cont.)

Convolution and Correlation

Convolution and correlation are LSI operators

Correlation ($g = f \otimes h$)

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$$

Convolution ($g = f * h$)

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l)$$

Linear Filtering (cont.)

Other representations of convolution

Shifted impulse responses

$$g(i,j) = \sum_m \sum_n f(m,n)h(i-m, j-n)$$

Sparse matrix multiplication

$$\boxed{72}\boxed{88}\boxed{62}\boxed{52}\boxed{37} \; * \; \frac{1}{6}\boxed{1}\boxed{2}\boxed{3} \quad \Leftrightarrow \quad \frac{1}{6}\begin{bmatrix} 2 & 3 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Linear Filtering (cont.)

Image Padding

- **Problem:** The convolution and correlation operations yield invalid values near the borders. To avoid invalid values, the size of the resulting image is shrinked.
- **Solution:** Apply padding (extend the image over the borders) to the original image before applying convolution.

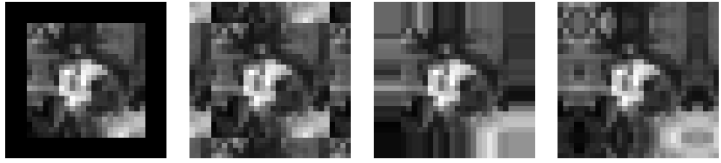


Figure 15: *Image padding techniques.*

Linear Filtering (cont.)

Separable filters

- **Problem:** Applying 2D convolution with a large kernel ($N = 100$) to the frames of a 4K video (3840 x 2160 pixels) takes too much time. Performing a 2D convolution requires N^2 (multiply-add) operations per pixel.
- **Solution:** Decompose the 2D filter kernel into 2 1D kernels. Performing a 1D convolution requires N operations per pixel (total of $2N$ operations/pixel).

Linear Filtering (cont.)

- The two-dimensional kernel K corresponding to successive convolution with a horizontal kernel $\vec{\mathbf{h}}$ and a vertical kernel $\vec{\mathbf{v}}$ is the outer product of the two kernels.

$$K = \vec{\mathbf{v}}\vec{\mathbf{h}}^{\mathsf{T}}$$

- For kernel with shape $N \times N$ the number of multiply-add operations per pixel is reduced from N^2 to $2N$.

Linear Filtering (cont.)

Examples of separable filters

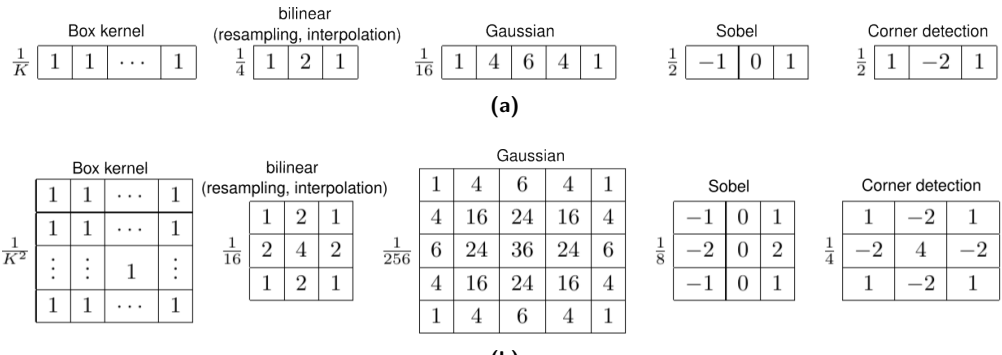


Figure 16: *Examples of separable filters: (a) 1D kernels; (b) 2D kernels.*

Linear Filtering (cont.)

Separability criterion

- Use SVD to decompose the kernel $K = \sum_i \sigma_i \vec{\mathbf{u}}_i \vec{\mathbf{v}}_i^\top$. Where σ_i are the singular values, $\vec{\mathbf{u}}_i$ are the left singular vectors, $\vec{\mathbf{v}}_i$ are the right singular vectors.
- The kernel is separable if it has only one non-zero singular value σ_0.

Separable filter approximation (non-separable $\rightarrow$ separable)

1. Decompose the kernel using SVD $K = \sum_{i=0}^{I} \sigma_i \vec{\mathbf{u}}_i \vec{\mathbf{v}}_i^\top$.
2. Take the t highest non-zero singular values (e.g. 1, 2).
3. Approximation of the original as $K' = \sum_{i=0}^{t} \sigma_i \vec{\mathbf{u}}_i \vec{\mathbf{v}}_i^\top \quad s.t. \quad t < I$
4. Convolution is linear: Convolve the image with each component $\sigma_i \vec{\mathbf{u}}_i \vec{\mathbf{v}}_i^\top$ and sum the results.

Linear Filtering (cont.)

Summed area table

- If an image is going to be repeatedly convolved with different box filters (different sizes N at different locations), we might need N^2 or $2N$ adding operations.
- We can reduce the number of operations to just 4.
- Summed area table $s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$

$$S(i_0 : i_1, j_0 : j_1) = \sum_{k=i_0}^{i_1} \sum_{l=j_0}^{j_1} f(k,l) = s(i_1,j_1) - s(i_1,j_0-1) - s(i_0-1,j_1) + s(i_0-1,j_0-1)$$

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

Figure 17: *Summed area tables: (a) original image; (b) summed area table; (c) computation of area sum*

Linear Filtering (cont.)

Band-pass steerable filters

- The Sobel and corner operators are simple examples of band-pass and oriented filters.
- More sophisticated kernels can be created by first smoothing the image with a Gaussian filter, and then taking the first or second derivatives.
- Such filters are known collectively as band-pass filters, since they filter out both low and high frequencies.

Linear Filtering (cont.)

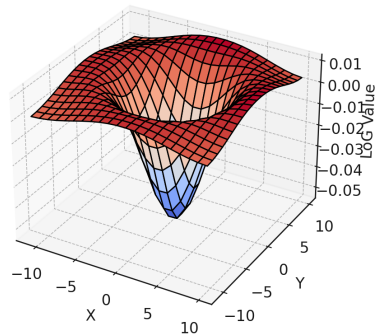
Laplacian of Gaussian (isotropic band-pass filter)

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right) G(x, y; \sigma)$$

Laplacian of Gaussian (LoG) Kernel



Linear Filtering (cont.)

Steerable band-pass filter

Steerable filtering (Filtering the image and steering the response) :

$$\hat{\mathbf{u}} \cdot \nabla(G * f), \quad \hat{\mathbf{u}} = (\cos\theta, \sin\theta)$$

Steerable kernel (Steering the kernel) :

$$G_{\hat{u}} = uG_x + vG_y = u\frac{\partial G}{\partial x} + v\frac{\partial G}{\partial y}$$

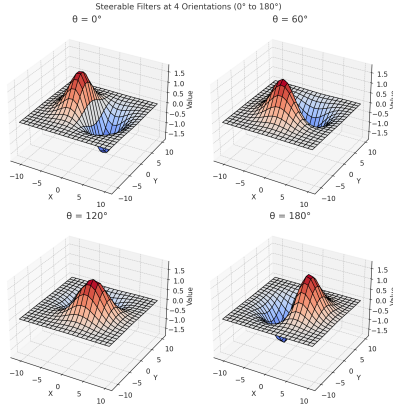


Figure 18: *Example of steerable kernels for 4 different orientations* $(0°, 60°, 120°, 180°)$

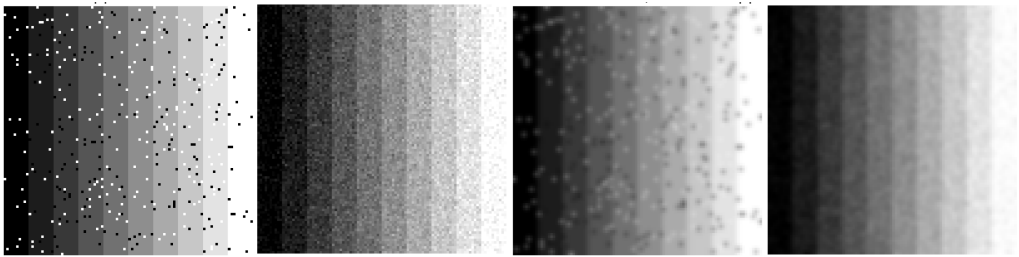
More Neighborhood Operators

"Non-linear filters, Morphological operations over binary images"

More Neighborhood Operators

Image Noise

Unwanted random variations in pixel intensity that degrade image quality. It is typically introduced during image acquisition, transmission, or processing.



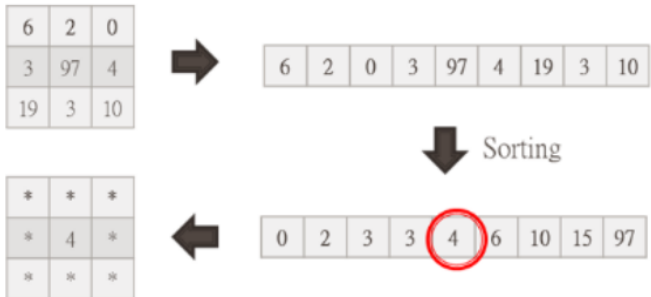
(a) **(b)** **(c)** **(d)**

Figure 19: *Example of the application of the median filter: (a) Image with salt-pepper noise; (b) Image with additive gaussian noise; (c) Box filter applied to Fig. 19a; (c) Box filter applied to Fig. 19b*

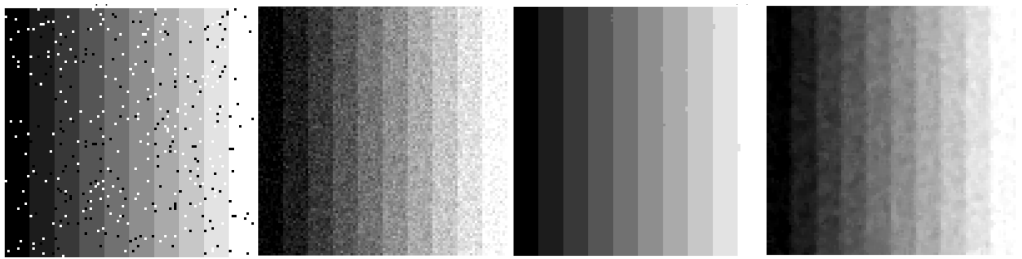
More Neighborhood Operators (cont.)

Median filtering

Non-linear edge-preserving filter. It selects the median value from each pixel's neighborhood.



More Neighborhood Operators (cont.)



(a) (b) (c) (d)

Figure 20: *Example of the application of the median filter: (a) Image with salt-pepper noise; (b) Image with additive gaussian noise; (c) Median filter applied to Fig. 20a; (c) Median filter applied to Fig. 20b*

More Neighborhood Operators (cont.)

α-trimmed mean

Sort the entries of a rectangular region. Average the region's entries disregarding the lowest and highest values according to the α fraction.

$$g(i,j) = \frac{1}{|S_{ij}| - 2\lfloor \alpha |S_{ij}| \rfloor} \sum_{(s,t) \in S_{ij}} f(s,t)$$

13	28	33
15	76	32
17	30	42

$=\{13,15,17,28,30,32,33,42,76\}$

if $\alpha = 0.25 \rightarrow \lfloor \alpha |S_{ij}| \rfloor = 2$

Mean($\{13,15,17,28,30,32,33,42,76\}$)=28

More Neighborhood Operators (cont.)

Bilateral filtering (leveraging spatial and intensity data)

Assign lower importance to pixels whose intensity values differ too much from the central pixel value and are distant.

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,j,l)}{\sum_{k,l} w(i,j,k,j,l)}$$

Bilateral weight function

$$w(i,j,k,j,l) = d(i,j,k,l) r(i,j,k,l)$$

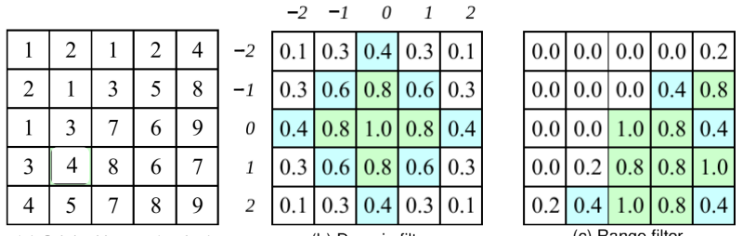
Domain kernel (spatial information)

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right)$$

Range kernel (intensity information)

$$r(i,j,k,l) = \exp\left(-\frac{(f(i,j) - f(k,l))^2}{2\sigma_r^2}\right)$$

More Neighborhood Operators (cont.)



(a) Original image (region)

	−2	−1	0	1	2
−2	0.1	0.3	0.4	0.3	0.1
−1	0.3	0.6	0.8	0.6	0.3
0	0.4	0.8	1.0	0.8	0.4
1	0.3	0.6	0.8	0.6	0.3
2	0.1	0.3	0.4	0.3	0.1

(b) Domain filter

0.0	0.0	0.0	0.0	0.2
0.0	0.0	0.0	0.4	0.8
0.0	0.0	1.0	0.8	0.4
0.0	0.2	0.8	0.8	1.0
0.2	0.4	1.0	0.8	0.4

(c) Range filter

Figure 21: *Domain and range filters used for bilateral filtering: (a) Original image; (b) Domain filter; (c) Range filter.*

More Neighborhood Operators (cont.)

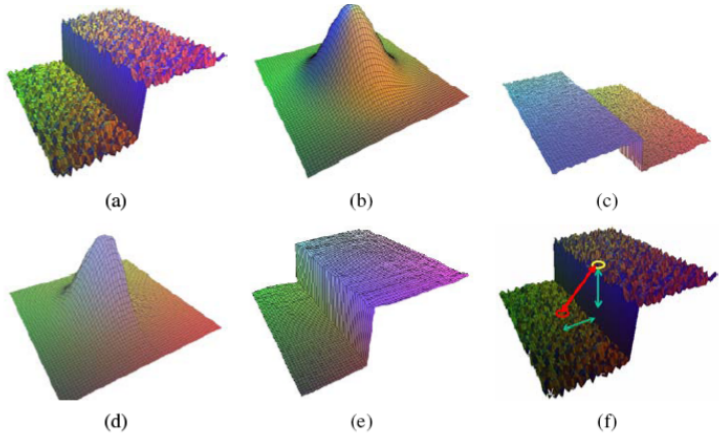


Figure 22: *Bilateral filtering: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.*

More Neighborhood Operators (cont.)

Binary images

- Images whose entries are 1 or 0.
- Such images often occur after a thresholding operation

$$\theta(f,t) = \begin{cases} 1, & \text{if } f \geq t, \\ 0, & \text{otherwise.} \end{cases}$$

More Neighborhood Operators (cont.)

Morphological operations

- The most common binary image operations are called morphological operations, because they change the shape of the underlying binary objects.
- Morphological operations are regarded as a combination of binary convolutions (binary image, binary kernel) and thresholding. However this is not the standard definition (use of set operations).

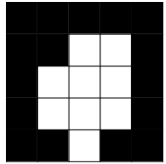
Binary convolution

$$c = f \otimes s$$

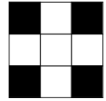
Dilation:	$\text{dilate}(f, s) = \theta(c, 1);$
Erosion:	$\text{erode}(f, s) = \theta(c, S);$
Majority:	$\text{maj}(f, s) = \theta(c, S/2);$
Opening:	$\text{open}(f, s) = \text{dilate}(\text{erode}(f, s), s);$
Closing:	$\text{close}(f, s) = \text{erode}(\text{dilate}(f, s), s).$

- f: input binary image.
- s: binary kernel.
- c: Output image (non binary). c(i,j) counts how many 1-entries in the image f match the 1-entries in the kernel s.
- S: The number of 1-entries in the structuring element (kernel).

More Neighborhood Operators (cont.)



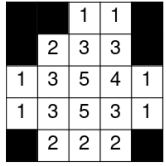
f



s

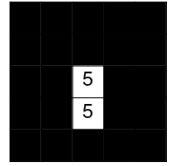
0	0	1	1	0
0	2	3	3	0
1	3	5	4	1
1	3	5	3	1
0	2	2	2	0

c



Dilation

$\theta(c, 1)$



Erosion

$\theta(c, 5)$

More Neighborhood Operators (cont.)

Distance transforms (minimum distance to a zero pixel)

The distance transform $D(i,j)$ of a binary image $b(i,j)$ is defined as follows. Let $d(k,l)$ be some distance metric between pixel offsets. Two commonly used metrics include the Manhattan distance and Euclidean distance.

$$D(i,j) = \min_{k,l\,:\,b(k,l)=0} d(i-k, j-l)$$

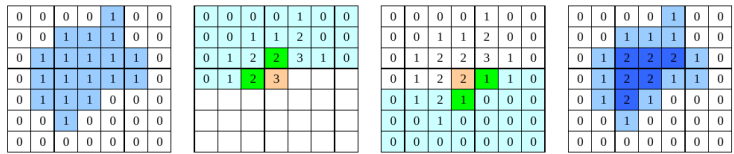
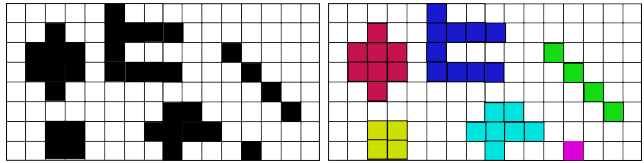


Figure 23: *City block distance transform: (a) original binary image; (b) top to bottom (forward) raster sweep: green values are used to compute the orange value; (c) bottom to top (backward) raster sweep: green values are merged with old orange value; (d) final distance transform.*

More Neighborhood Operators (cont.)

Connected components and component properties

Regions of adjacent pixels that have the same input value or label.



- The area: the number of pixels in the region.
- The perimeter: the number of boundary pixels.
- The centroid: the average x and y values,

$$\bar{x} = \frac{1}{N} \sum_{(x,y) \in R} x, \quad \bar{y} = \frac{1}{N} \sum_{(x,y) \in R} y.$$

- The second moments:

$$M = \sum_{(x,y) \in R} \begin{bmatrix} x - \bar{x} \\ y - \bar{y} \end{bmatrix} \begin{bmatrix} x - \bar{x} & y - \bar{y} \end{bmatrix}$$

Fourier Transforms

"Processing an image in the frequency domain and its applications"

Fourier Transforms

Fourier transform and 2D Fourier transform

- According to the Fourier analysis, any image (signal) can be represented as series of weighted periodic components (sine, cosine) with specific frequency values. The weights are determined by the Fourier transform.
- We use Fourier Transforms to analyze what a given filter in the spatial domain does to high, medium, and low frequency components (analysis in the frequency domain).
- Fourier transform is reversible without information loss. FTT $\leftrightarrow$ IFFT

Fourier Transforms (cont.)

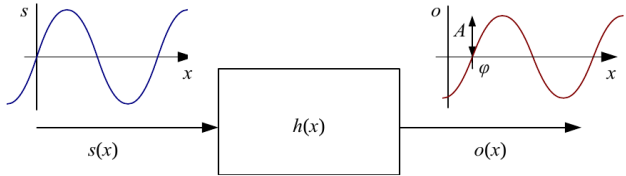


Figure 24: *The Fourier Transform as the response of a filter h(x) to an input sinusoid* $s(x) = e^{j\omega x}$ *yielding an output sinusoid* $o(x) = h(x) * s(x) = Ae^{j(\omega x+\phi)}$

1D Fourier transform

$$H(k) = \sum_{x=0}^{N-1} h(x) e^{-j\frac{2\pi kx}{N}}$$

2D Fourier transform (Image with shape $N \times M$)

$$H(k_x, k_y) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-j2\pi\left(\frac{k_x x}{M} + \frac{k_y y}{N}\right)}$$

Fourier Transforms (cont.)

Discrete Cosine Transform (DCT)

1D Discrete Cosine Transform

$$F(k) = \sum_{i=0}^{N-1} \cos\left(\frac{\pi}{N}\left(i + \frac{1}{2}\right)k\right) f(i)$$

2D Discrete Cosine Transform (Image with shape $W \times H$)

$$F(k,l) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \cos\left(\frac{\pi}{N}\left(i + \frac{1}{2}\right)k\right) \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)l\right) f(i,j)$$

The DCT is widely used in today's image and video compression algorithms (e.g. JPEG2000)

Pyramids and Wavelets

"Multiresolution processing, Interpolation, Decimation"

Image pyramid

- Image pyramids are extremely useful for performing multi-scale editing operations such as blending images while maintaining details.
- For example, the task of finding a face in an image. Since we do not know the scale at which the face will appear, we need to generate a whole pyramid of differently sized images and scan each one for possible faces.
- Changing image resolution to construct pyramids requires: upsampling (interpolation) and downsampling (decimation).

Pyramids and Wavelets (cont.)

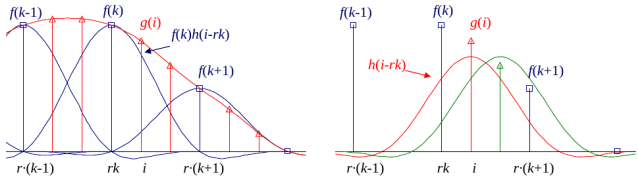


Figure 25: *Signal interpolation $g(i) = \sum_k f(k)h(i - rk)$: (a) weighted summation of input values; (b) polyphase interpretation*

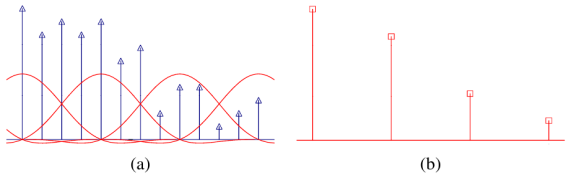


Figure 26: *Signal decimation: (a) the original samples are (b) convolved with a low-pass filter before being downsampled.*

Pyramids and Wavelets (cont.)

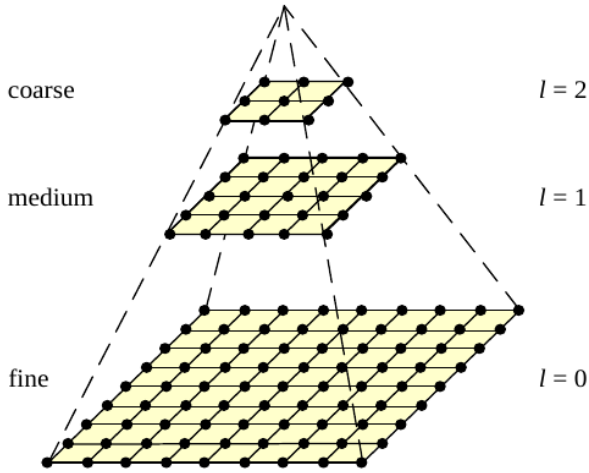


Figure 27: *A traditional image pyramid: each level has half the resolution (width and height), and hence a quarter of the pixels, of its parent level*

Pyramids and Wavelets (cont.)

Wavelets vs. Pyramids

- Wavelets are filters that localize a signal in both space and frequency (like the Gabor filter) and are defined over a hierarchy of scales.
- Wavelets provide a smooth way to decompose a signal into frequency components without blocking and are closely related to pyramids.
- Traditional pyramids are overcomplete (i.e. they use more pixels than the original image to represent the decomposition), whereas wavelets provide a tight frame (i.e., they keep the size of the decomposition the same as the image)

Pyramids and Wavelets (cont.)



(a) (b)

Figure 28: *Example of a wavelet decomposition: (a) Original image, (b) Wavelet decomposition.*

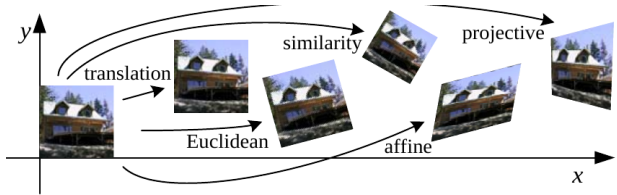
Geometric Transformations

"Aims to modify a pixel's position (image's domain) rather than its intensity (image's range)"

Geometric Transformations

Parametric transformations (affines and projection)

Parametric transformations apply a global deformation to an image, where the behavior of the transformation is controlled by a small number of parameters.



Geometric operator

The operator h aims to modify the pixels' locations within the image.

$$g(\vec{\mathbf{x}}) = f(h(\vec{\mathbf{x}}))$$

Geometric Transformations (cont.)

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix}\mathbf{I} & \mathbf{t}\end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix}s\mathbf{R} & \mathbf{t}\end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix}\mathbf{A}\end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix}\tilde{\mathbf{H}}\end{bmatrix}_{3\times 3}$	8	straight lines	

Figure 29: *Hierarchy of 2D coordinate transformations*

Geometric Transformations (cont.)

Image warping algorithms

procedure *forwardWarp*$(f, \mathbf{h}, \textbf{out}\ g)$:

For every pixel $\mathbf{x}$ in $f(\mathbf{x})$

1. Compute the destination location $\mathbf{x}' = \mathbf{h}(\mathbf{x})$.
2. Copy the pixel $f(\mathbf{x})$ to $g(\mathbf{x}')$.

Figure 30: *Forward warping*

procedure *inverseWarp*$(f, \mathbf{h}, \textbf{out}\ g)$:

For every pixel $\mathbf{x}'$ in $g(\mathbf{x}')$

1. Compute the source location $\mathbf{x} = \hat{\mathbf{h}}(\mathbf{x}')$
2. Resample $f(\mathbf{x})$ at location $\mathbf{x}$ and copy to $g(\mathbf{x}')$

Figure 31: *Inverse warping*

Geometric Transformations (cont.)



Figure 32: *Forward warping*



Figure 33: *Inverse warping*

Geometric Transformations (cont.)

Mesh based warping

- While parametric transforms specified by a small number of global parameters have many uses, local deformations with more degrees of freedom are often required.
- Different amounts of transformations are required in different parts of the image.

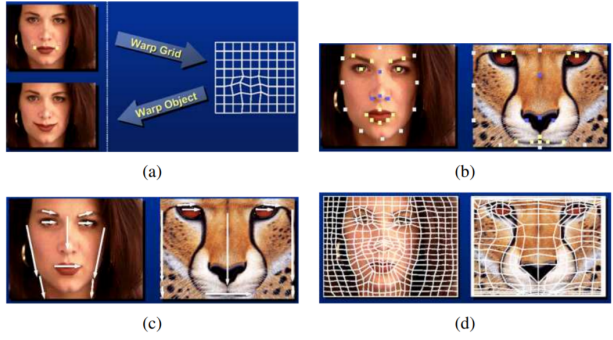


Figure 34: *Mesh based warping*