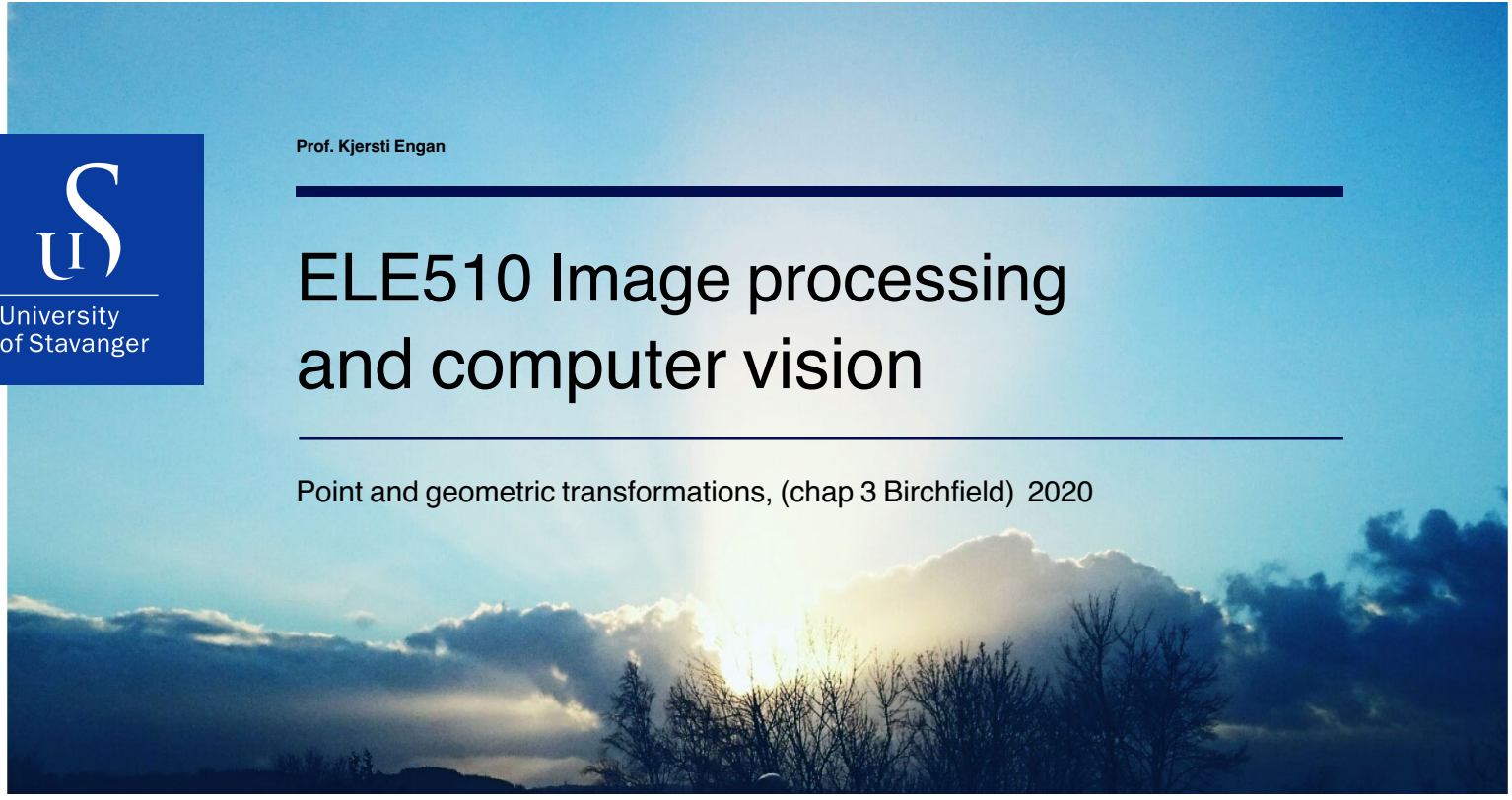
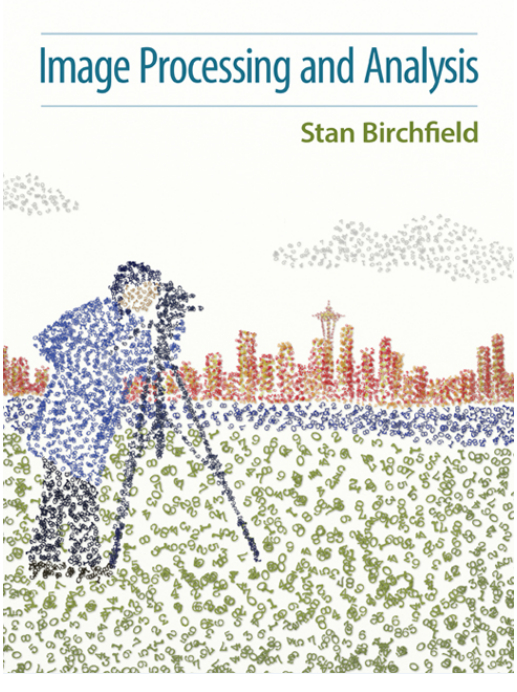


University of Stavanger

Prof. Kjersti Engan

# ELE510 Image processing and computer vision

Point and geometric transformations, (chap 3 Birchfield) 2020




Parts in course presentations is material from Cengage learning. It can be used for teaching, and it can be share it with students on access controlled web-sites (Canvas) for use in THIS course. **But it should not be copied or distributed further in any way** (by you or anybody).

# Geometric and point transformation (chap 3)

- Geometric transformations:
  - changing a pixels location without changing its value

- Point transformations:
  - change a pixels value without changing the location
  - Independent of pixel coordinates and values of other pixels

# (3.1) Geometric transformations

- Simple geometric transformations: Output pixel only dependent on a single input pixel
- Changes a pixels location without changing its value

- (x,y) : coordinates of input pixel.  Input image :  I(x,y)
- (x´,y´) : coordinates of output pixel. Output image :I´(x´,y´)

- Forward mapping: compute destination coordinates from source
- Invers mapping; compute source coordinates from destination – often preferred to be sure there are no ambiguities ( not multiple answers to a destination coodrdinate).
- For some mappings they are equivalent.

# Flipping and flopping

- The simplest geometric transformation is to reflect the image about a horizontal or vertical axis passing through the center of the image.
  - If the axis is horizontal, the transformation **flips** the image upside down.
  - If the axis is vertical, the transformation **flops** the image to produce a right-to-left mirror image.

$$\begin{bmatrix} 128 & 78 & 174 \\ 181 & 48 & 77 \\ 109 & 49 & 138 \end{bmatrix} \xrightarrow{\text{FLOP}} \begin{bmatrix} 174 & 78 & 128 \\ 77 & 48 & 181 \\ 138 & 49 & 109 \end{bmatrix}$$

$$\downarrow \text{FLIP}$$

$$\begin{bmatrix} 109 & 49 & 138 \\ 181 & 48 & 77 \\ 128 & 78 & 174 \end{bmatrix}$$

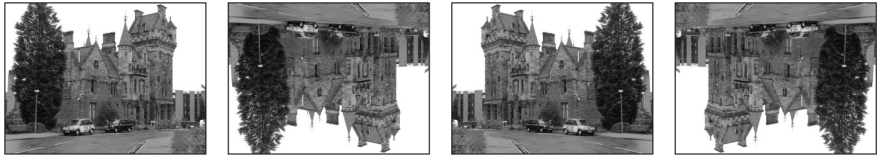


Image Flip Flop Flip-flop

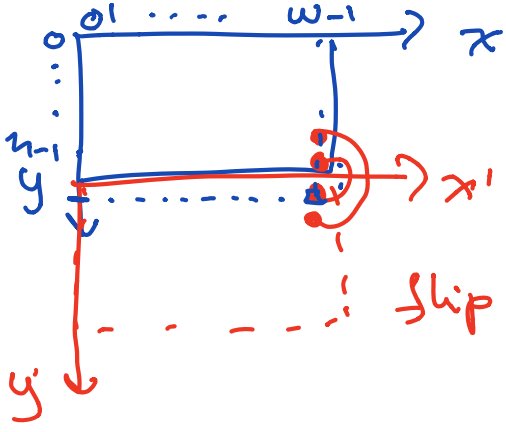
Stan Birchfield

forward mapping, flip

$$x' = x$$

$$y' = h-1-y$$

$$I'(x, h-1-y) = I(x, y)$$




Inverse mapping flip

$$I'(x', y') = I(x', h-1-y')$$

# Rotating by a multiple of 90 degrees

$$I'(height - 1 - y, x) = I(x, y) \qquad I'(x', y') = I(y', height - 1 - x')$$

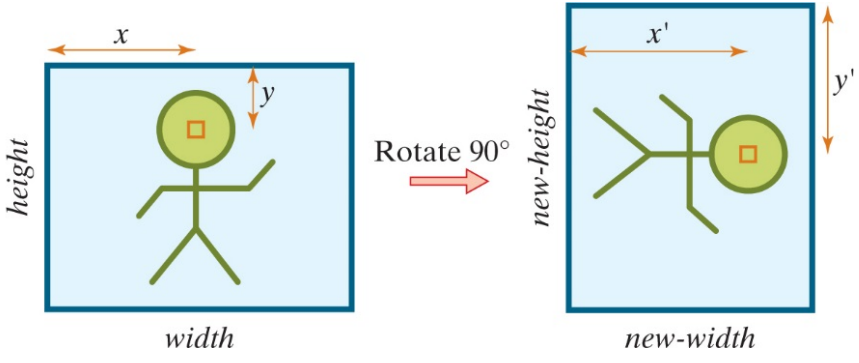
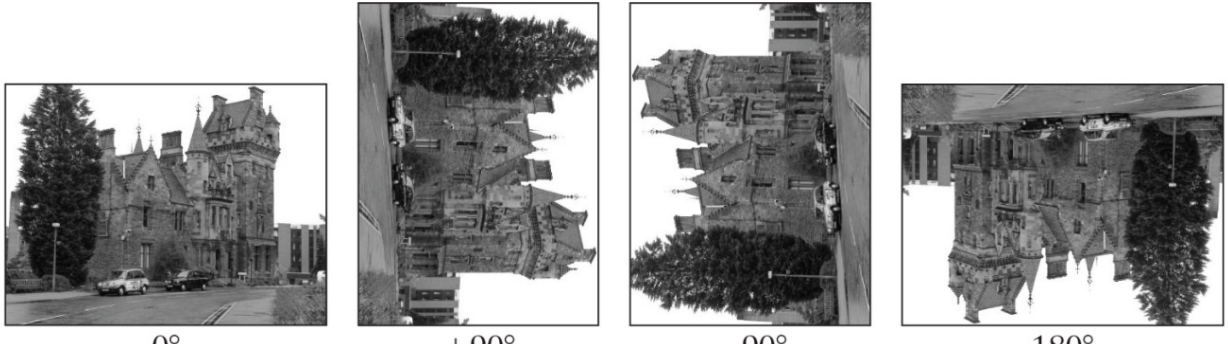



**Figure 3.4** To rotate an image clockwise by 90 degrees, the pixel $(x, y)$ in the input image is mapped to $(x', y')$ in the output image. From the drawing, it is easy to see that $x' = new\text{-}width - 1 - y = height - 1 - y$, and $y' = x$.

# Rotating by a multiple of 90 degrees

**Figure 3.3** An image rotated by 0, +90, −90, and 180 degrees.



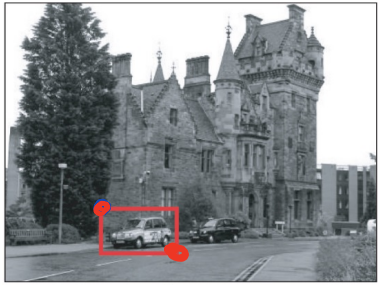

Stan Birchfield

# Cropping an Image

$I(x, y)$

**Figure 3.5** An image and an automobile cropped out of the region of the image indicated by the red rectangle.

$x = left$
$y = top$



Image

$(0,0)$ $I'(x', y')$

Cropped region

Stan Birchfield

**ALGORITHM 3.4** Crop an image

CROPIMAGE(*I*, *left*, *top*, *right*, *bottom*)

**Input:** image $I$, rectangle with corners (*left*, *top*) and (*right*$-1$, *bottom*$-1$)
**Output:** cropped image $I'$ of size *new-width* $\times$ *new-height*

```
1  new-width ← right − left
2  new-height ← bottom − top
3  I′ ← ALLOCATEIMAGE(new-width, new-height)
4  for (x′, y′) ∈ I′ do
5     I′(x′, y′) ← I(x′ + left, y′ + top)
6  return I′
```

# Downsampling and upsampling

- **Downsample** an image to produce a smaller image than the original (should be smoothed first):

$$I'(x, y) = I(2x, 2y) \qquad \text{(downsample by two)}$$

- **Upsample** an image to produce a larger image than the original (interpolation should be done):

$$I'(x, y) = I\left(\left\lfloor \frac{x}{2} \right\rfloor, \left\lfloor \frac{y}{2} \right\rfloor\right) \qquad \text{(upsample by two)}$$

$\lfloor \frac{x}{2} \rfloor$ floor $x=1$ $\lfloor \frac{1}{2} \rfloor = 0$

$I'(1,1) = I(0,0)$

$I'(0,0) = I(0,0)$

**Figure 3.6** LEFT: An image and the result of downsampling by a factor of 2 and 4, respectively, in each direction. RIGHT: A cropped region and the result of upsampling by a factor of 2 and 4, respectively, in each direction.

# (3.2) Point transformations

- changes a pixel's value without changing its location.
- Independent of pixels coordinate and coordinates and values of other pixels
  - ✓ Special case of spatial domain filtering where the constrain on values of neighbor pixels is removed.

- Gray level transformations
- Gray level histograms

$$I'(x,y) = f(I(x,y))$$

8 bpp

$I(x,y)$

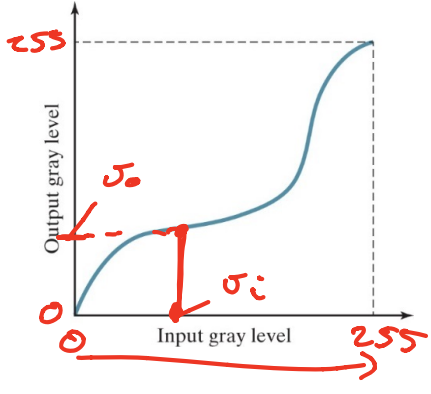


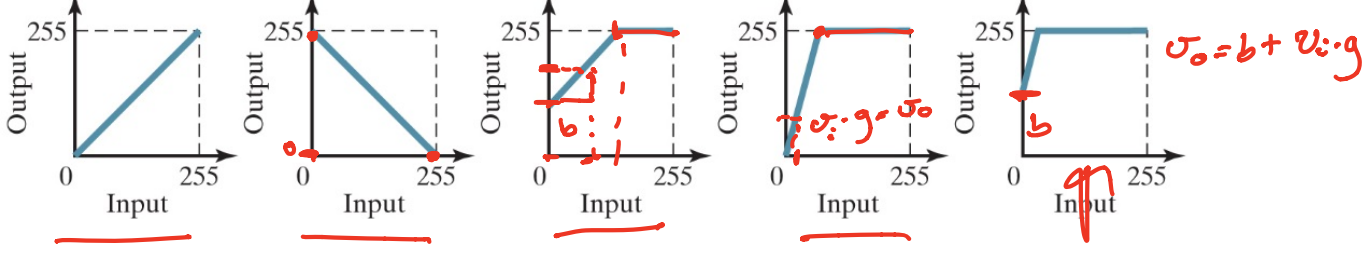

Figure 3.7 A graylevel transformation maps input gray levels to output gray levels. Based on http://www.unit.eu/cours/videocommunication/Point_Transformation_histogram.pdf

# Arithmetic operations

- A useful class of gray level transformations is the set of arithmetic operations; inversion, addition (bias) , multiplication (gain)
- If we want a grayscale image out with we need to use saturation arithmetic:

  example, addition (bias=b)

**Figure 3.8** Arithmetic graylevel transformations. From left to right: identity, inversion, addition (bias), multiplication (gain), and gain-bias transformation, where saturation arithmetic prevents the output from exceeding the valid range. Note that the slope remains 1 under addition, while the mapping passes through the origin under multiplication.

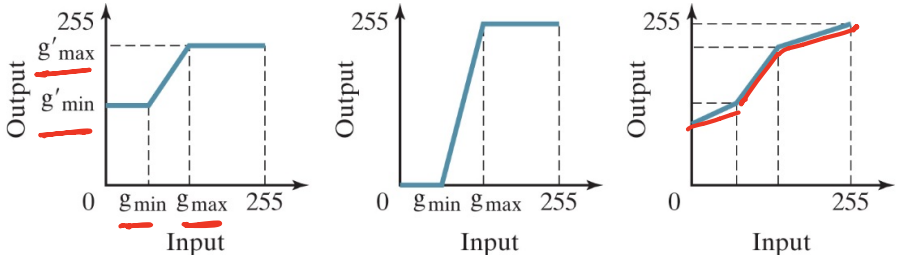


# Linear contrast stretching

- **Linear contrast stretch:** A transformation that specifies a line segment that maps gray levels between $g_{min}$ and $g_{max}$ in the input image to the gray levels $g'_{min}$ and $g'_{max}$ in the output image according to a linear function:

$$I'(x,y) = \frac{g'_{max} - g'_{min}}{g_{max} - g_{min}}(I(x,y) - g_{min}) + g'_{min}$$

- Piecewise linear contrast stretch can model any gray level transformation given enough line segments



# Analytic transformations

- Graylevel transformations can be specified using analytic functions such as the logarithm, exponential, or power functions:

$$I'(x,y) = \log(I(x,y))$$

$$I'(x,y) = \exp(I(x,y))$$

$$I'(x,y) = (I(x,y))^{\gamma}$$

$\gamma > 1$ gamma expansion

$\gamma > 0$ $0 < \gamma < 1$ gamma compression

- Rounding and clamping to [0,255] is needed to get a 8 bpp image. This is always the case but often left out for clarity. Might use images as matrices with real numbers, and perform multiple operations / filtering etc. Rounding/clamping is done on *final* image

# Thresholding

- Takes a grayscale image and sets every output pixel to 1 if its input gray level is above a certain threshold, or to 0 otherwise:

$$I'(x,y) = \begin{cases} 1 & \text{if } I(x,y) > \tau \\ 0 & \text{otherwise} \end{cases}$$

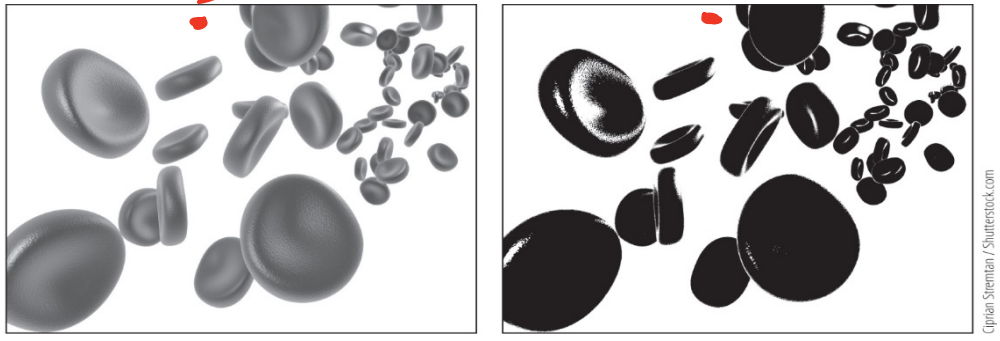
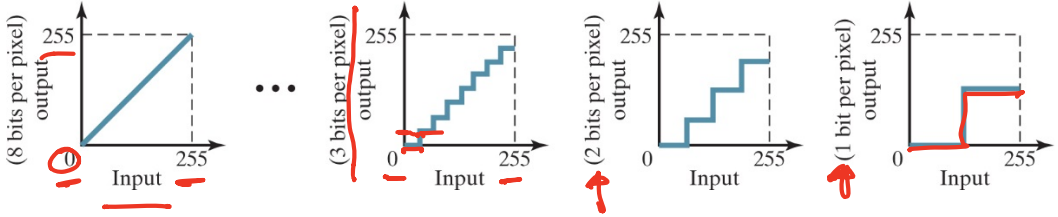


**Figure 3.14** An 8-bit grayscale image (left), and the binarized result obtained by thresholding with $\tau = 150$ (right).

- **Density slicing:** assigns all gray levels within a certain range to a certain value. Others can be unaltered, set to zero etc.

- **Quantization:** discards one or more of the lower-order bits. How many bits per pixel (bpp)?

**Figure 3.16** Quantization discards the lower-order bits via a staircase function, with the number of stairs determined by the number of bits retained. From right to left: Only 1 bit is retained, so the gray levels in the dark half (less than 128) map to 0, while the gray levels in the bright half (above 127) map to 128 (binary: 10000000); 2 bits are retained, so gray levels are mapped to either 0, 64, 128, or 192; 3 bits are retained, so all gray levels are mapped to either 0, 32, 64, 96, 128, 160, 192, or 224; all 8 bits are retained (no quantization, the identity function).



**Bit-plane slicing:**

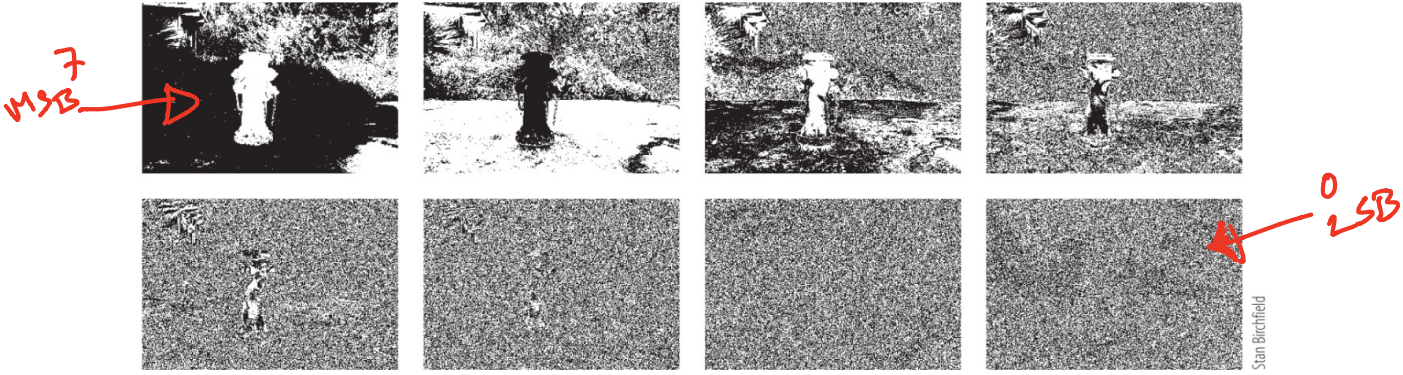
A *bit plane* is a binary image whose value at each pixel is the same as the appropriate bit in the corresponding pixel of the original image.

Bit plane 0: least significan bit (LSB)

Bit plane b-1: most significant bit (MSB) ( b is no of bpp)

Bit-slicing at bit plane b-1: quantization with 1 bpp

**Figure 3.19** From left-to-right and top-to-bottom: Bit planes 7, 6, 5, 4, 3, 2, 1, and 0 of the hydrant image. Note that the higher-order bit planes bear some resemblance to the original image, while the lower-order bit planes appear as noise. Bit plane 7 is identical, apart from scaling, to the quantization with 1 bit per pixel. The image reconstructed from the highest four bit planes (that is, 4 through 7) is shown in the bottom-left of Figure 3.17.



# Background subtraction

- **Background image:** a reference image that does not contain any foreground objects. If available;

$$I'(x, y) = |I(x, y) - B(x, y)|$$

- Can capture foreground objects even if they do not move for a while ( not possible with frame difference)

- **Digital subtraction angiography (DSA)** : a reference image is captured of a blood vessel before injecting it with dye to increase contrast.

- Alternative; find **average image** over successive video frames