EEE-6512: Image Processing and Computer Vision

August 30, 2017

Lecture #5: Point and Geometric Transformations

Damon L. Woodard, Ph.D.
Dept. of Electrical and Computer Engineering
dwoodard@ece.ufl.edu

Chapter Outline

- Different ways to transform an image into another image
- Simple Geometric Transformations
- Graylevel Transformations
- Graylevel Histograms
- Multispectral Transformations
- Multi-Image Transformations
- Change Detection
- Compositing
- Interpolation
- Warping

Change Detection

Change Detection - Frame Differencing

- **Difference image:** to compare successive image frames in the video sequence.

$$I'(x, y) = |I_t(x, y) - I_{t-1}(x, y)| > \tau$$

- **Double-difference image:** used to solve the double-image problem, that is, the difference image will contain foreground pixels not only where the foreground object is located in the current frame but also where it was in the previous frame.

$$I' = |I_t - I_{t-1}| > \tau \text{ AND } |I_{t+1} - I_t| > \tau$$

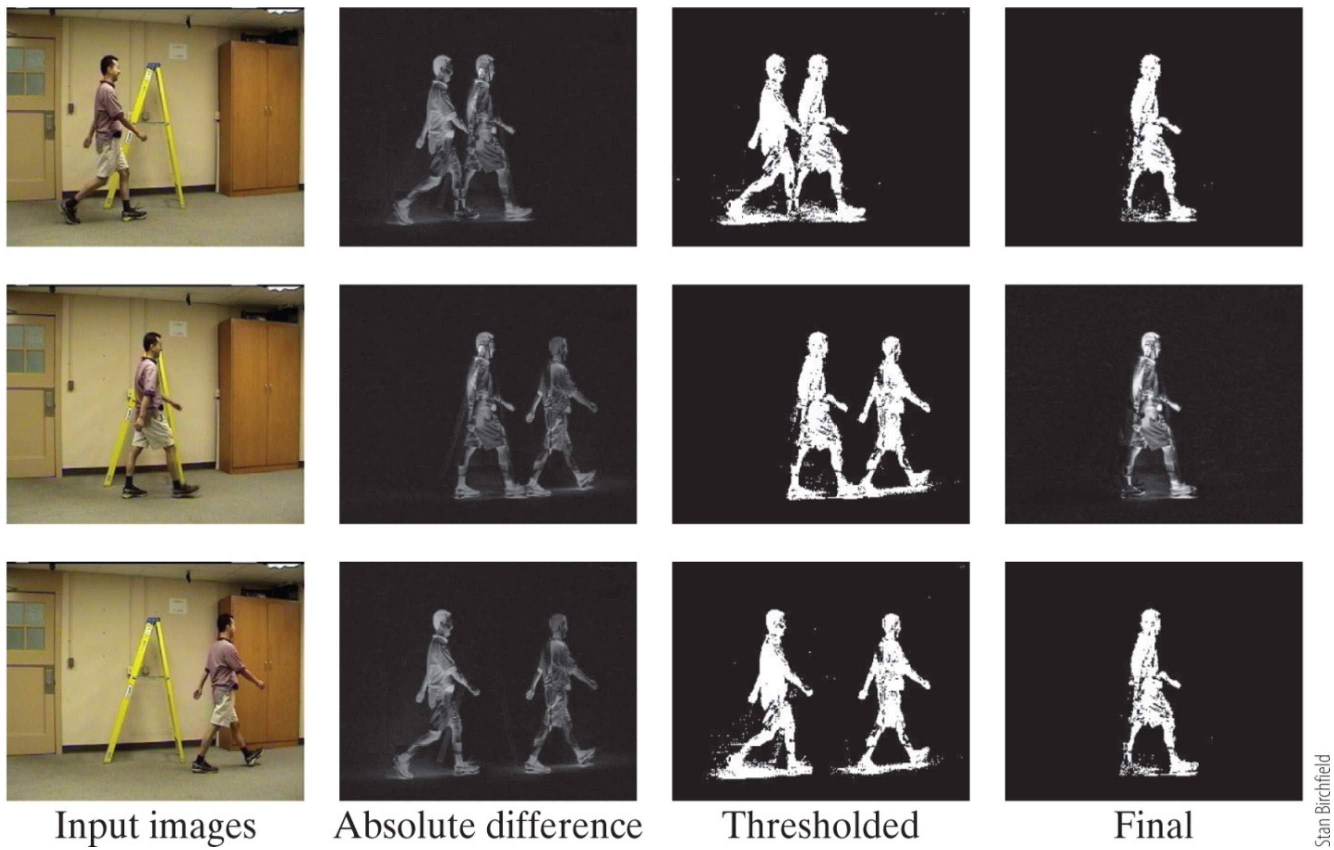
Change Detection - Frame Differencing (cont'd)

- **Triple-difference image:** combines the absolute differences from all three image pairs using addition and subtraction prior to thresholding:

$$I'(x, y) = (|I_{t-1} - I_t| + |I_{t+1} - I_t| - |I_{t-1} - I_{t+1}|) > \tau$$

Frame Differencing (cont.)

Figure 3.27 Detecting a moving object by frame differencing. LEFT COLUMN: Three image frames from a video sequence. SECOND COLUMN: The absolute difference between pairs of frames. THIRD COLUMN: Thresholded absolute difference. RIGHT COLUMN: Final result using double difference (top), triple difference (middle), and thresholded triple difference (bottom) methods.

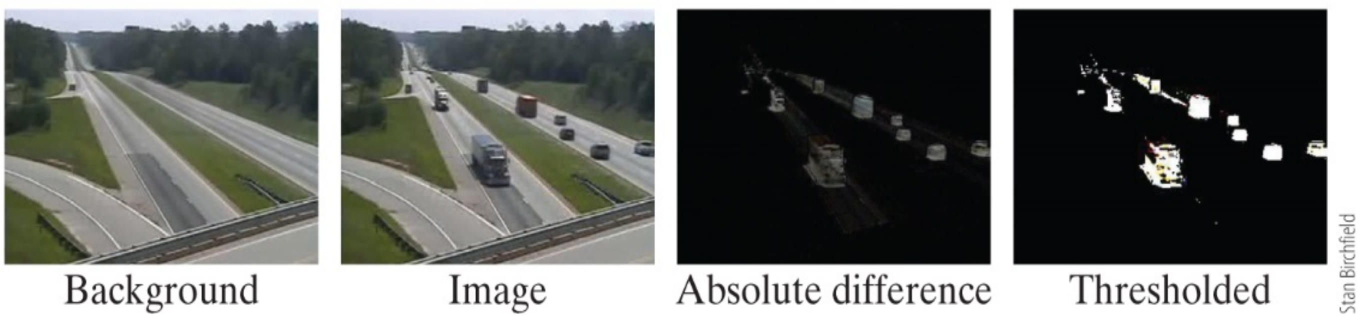


Background Subtraction

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

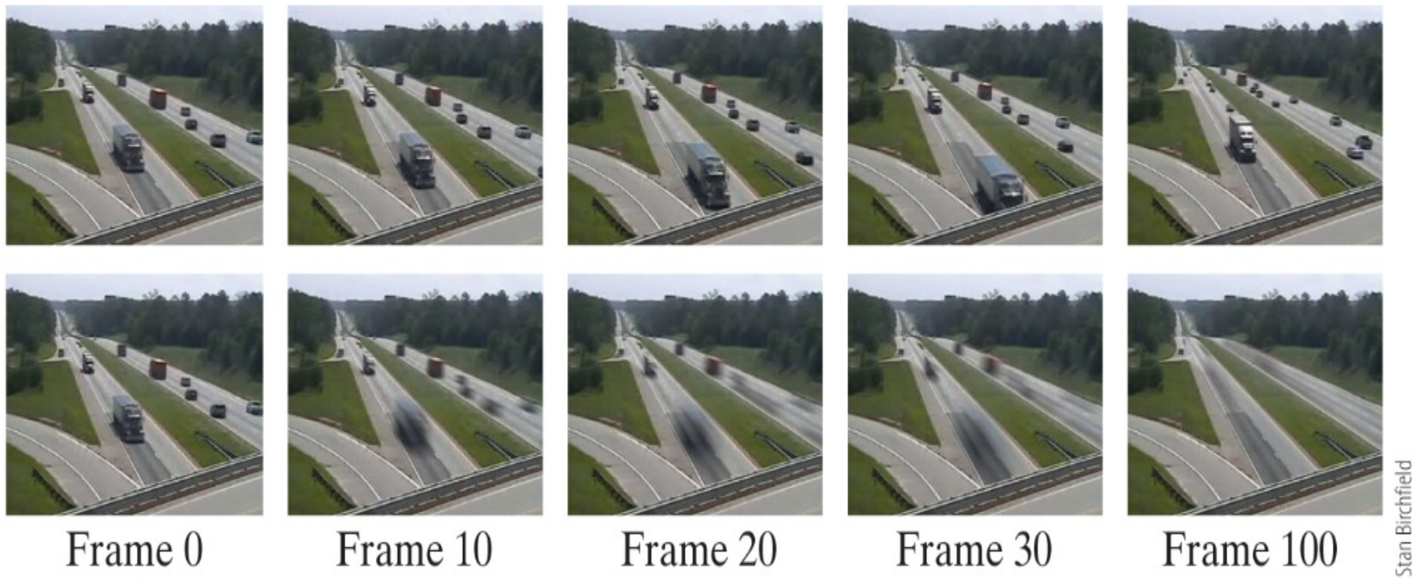
$$I'(x, y) = |I(x, y) - B(x, y)|$$

Figure 3.29 Background subtraction. From left to right: the background image, the current image, the absolute difference between the image and the background, and the thresholded absolute difference.



Background Subtraction (cont.)

Figure 3.28 Top: Five images from a video sequence. Bottom: Each column shows the mean image obtained using all the images up to and including the one above it. As time progresses the moving objects disappear, leaving only the background.



Background Subtraction (cont.)

Advantage: It separates the foreground objects even when they cease moving for a period of time.

Drawback: Objects that remain stationary for a long time prevent the detection of objects that may pass in front.

Compositing

Compositing

Widely used in the movie industry to blend live action with computer graphics or to blend different areas of a computer graphic scene rendered by different pieces of software.

- Dissolving
- Compositing with Binary Masks
- Compositing with Alpha Channels

Dissolving

Figure 3.30 Two examples of dissolving one image into another. Source: Screenshots by WETA Digital Ltd. – © 2011 Paramount Pictures. 'The Adventures of Tintin'

Compositing With Binary Masks

$M_A(x, y)$	$M_B(x, y)$	$M'(x, y)$	$I'(x, y)$
0	0	0	$\cdot$
0	1	0,1	$\cdot, I_B(x, y)$
1	0	0,1	$\cdot, I_A(x, y)$
1	1	0,1,1	$\cdot, I_A(x, y), I_B(x, y)$

TABLE 3.2 The four cases for any given pixel in compositing images with binary masks. For each case, the choices available for the output pixel are given in the last two columns, with a dot $(\cdot)$ meaning that the RGB value is irrelevant since the mask is zero. Each entry in the M' column is paired with an entry in the I' column.

Compositing With Binary Masks (cont.)

$M_A(x, y)$	$M_B(x, y)$	$M'(x, y)$	$I'(x, y)$
0	0	0	$\cdot$
0	1	0,1	$\cdot, I_B(x, y)$
1	0	0,1	$\cdot, I_A(x, y)$
1	1	0,1,1	$\cdot, I_A(x, y), I_B(x, y)$

TABLE 3.2 The four cases for any given pixel in compositing images with binary masks. For each case, the choices available for the output pixel are given in the last two columns, with a dot ($\cdot$) meaning that the RGB value is irrelevant since the mask is zero. Each entry in the M' column is paired with an entry in the I' column.

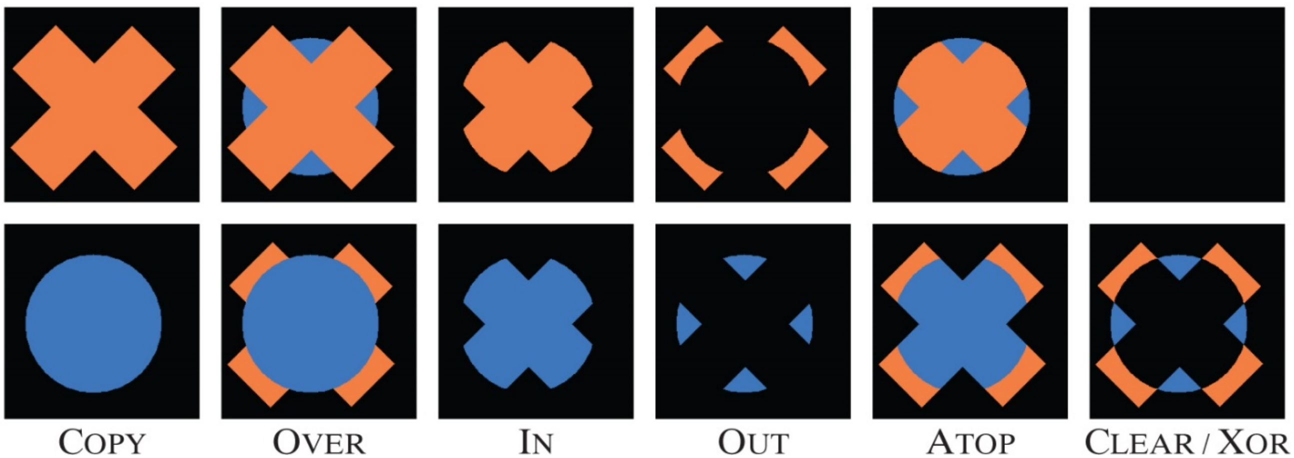
Compositing With Binary Masks (cont.)

Operation	I'	M'
CLEAR	0	0
COPY I_A	I_A	M_A
I_A OVER I_B	$I_A \wedge M_A + I_B \wedge M_B \wedge \neg M_A$	$M_A + M_B$
I_A IN I_B	I_A	$M_A \wedge M_B$
I_A OUT I_B	I_A	$M_A \wedge \neg M_B$
I_A ATOP I_B	$I_A \wedge M_A + I_B \wedge \neg M_A$	M_B
I_A XOR I_B	$I_A \wedge M_A \wedge \neg M_B + I_B \wedge \neg M_B \wedge \neg M_A$	$M_A \wedge \neg M_B + M_B \wedge \neg M_A$

TABLE 3.3 Formulas for compositing two images with binary masks. The formulas for the reverse versions of the non-commutative operations are easily obtained by swapping the operands. The caret $(\wedge)$ symbol refers to logical AND, the angle $(\neg)$ refers to logical NOT, and the plus $(+)$ symbol indicates logical OR (which is equivalent to addition in these formulas due to mutual exclusion between the terms).

Compositing With Binary Masks (cont.)

Figure 3.31 The twelve binary compositing operations. The first column shows the original two images. Columns 2 through 5 show the noncommutative operations, with the order of operands reversed in the two rows. The final column shows the CLEAR (top) and XOR (bottom) operations. In all cases the display shows the RGB image after applying the mask, i.e., I' AND M', with black pixels indicating a mask value of 0.



12 Compositing operations are called the Porter-Duff operators.

Compositing With Binary Masks (cont.)

Figure 3.32 Common binary compositing operations applied to a pair of masked images. The top two rows show, from left to right: Original image I_A and mask M_A, original image I_B and mask M_B, and image I' and mask M' resulting from the four operations OVER, IN, OUT, and ATOP, respectively. The bottom row shows the result of ANDing each image with each mask.

Compositing with Alpha Channels

Compositing with Alpha Channels

- One of the problems with binary masks is that they produce harsh, unnatural edges around the boundaries of objects when compositing two images.

- **Solution:** alpha channel

  It is the same size as the image, and each pixel in the alpha channel is (conceptually at least) a floating-point value, typically between 0 and 1, with 1 meaning that the associated RGB pixel is opaque and 0 meaning that it is transparent (or, equivalently, invisible or invalid).

$$I'(x, y) = \eta(x, y) I_A(x, y) + (1 - \eta(x, y)) I_B(x, y)$$

$$\alpha'(x, y) = \phi_A(x, y)\alpha_A(x, y) + \phi_B(x, y)\alpha_B(x, y)$$

Compositing with Alpha Channels (cont.)

Operation	ϕ_A	ϕ_B
I_A OVER I_B	1	$1 - \alpha_A$
I_B OVER I_A	$1 - \alpha_B$	1
I_A IN I_B	α_B	0
I_B IN I_A	0	α_A
I_A OUT I_B	$1 - \alpha_B$	0
I_B OUT I_A	0	$1 - \alpha_A$
I_A ATOP I_B	α_B	$1 - \alpha_A$
I_B ATOP I_A	$1 - \alpha_B$	α_A
I_A XOR I_B	$1 - \alpha_B$	$1 - \alpha_A$

TABLE 3.4 Coefficients for the different compositing operations.

Questions?