Video Examples
Example 1 – QR codes

• A “QR” code (Quick Response Code) is a two-dimensional barcode that is machine-readable.

• It encodes information about the item to which it is attached.

• It is designed to be quickly and reliably detected … simple thresholding can segment the black from white regions.
clear all
close all

movieObj = VideoReader('qr1.wmv'); % open file

images = read(movieObj); % get all images
nImg = size(images,4); % Number of images read
fprintf('Read in %d images from video files\n', nImg);

H = size(images,1); % Height of image
W = size(images,2); % Width of image

for iImg=1:nImg
    Irgb = images(:, :, :, iImg); % Get next image
    I = rgb2gray(Irgb);

    subplot(1,2,1), imshow(I, []);
    % Display image number
    text(20, H-20, sprintf('%d', iImg), 'Color', 'k', 'BackgroundColor', 'w');

    % Do Otsu global thresholding
    BW = im2bw(I, graythresh(I));

    subplot(1,2,2), imshow(BW);
    drawnow;
end
Example 1 - continued

• If lighting across the image is uneven, a global threshold may not work … a local, or adaptive threshold may be needed.

• A simple way to do local thresholding is to subtract the local mean in each neighborhood from the image.

• If the two classes are of (about) equal population, this should make the threshold separating them equal to 0.
Example 1 - continued

• Replace the Otsu threshold by local thresholding:
  – Compute the local mean at each pixel by filtering the image with a Gaussian, using a large value for sigma
  – Subtract the local mean image from the original image
  – Threshold the result at zero
Example 2 – Finding circles

• The Hough transform can be used to find circles \((x_0, y_0, r)\)
• In this case the parameter space is 3 dimensional

\[(x - x_0)^2 + (y - y_0)^2 = r^2\]
clear all
close all

movie = VideoReader('sphero1.wmv');
images = read(movie); % get all images
nImg = size(images,4); % Number of images read
fprintf('Read in %d images from video file\n', nImg);

% Suppress warnings about low accuracy because of small radii.
warning('off', 'all');

for i=1:2:nImg
    I = images(:,:,:,i); % Get next image
    G = rgb2gray(I);

    [centers, radii] = imfindcircles(G, [5, 20]);

    imshow(I, []), title(sprintf('%d', i));
    for j=1:size(centers,1)
        r = radii(j);
        x = centers(j,1);
        y = centers(j,2);
        rectangle('Position', [x-r y-r 2*r 2*r], 'EdgeColor', 'r', ...
           'Curvature', [1 1], 'LineWidth', 2);
    end
drawnow;
end

Matlab code and images
from Grant Latham
Example 3

• To detect moving objects in a scene, it helps to have a reference, or background image of the scene with no objects in it.
• Then, moving objects can be detected by subtracting the background image from each new image.
• To estimate the reference image, ideally you would take an image of the scene when it contained no moving objects.
• However, that isn’t always possible. An alternative is take the average over many images … if moving objects are fairly infrequent, their effect will be small and will cancel out. Even better is to take the median at each pixel.
Example 3 - continued

• On the course website is a sequence of 300 images (taken at 30 frames per second) of a street scene.

• Find the background image for the scene.
Example 4

• Using the background image you computed, detect moving objects in the sequence of images:
  – Take the difference between each image and the background
  – Threshold the difference image and label connected components
  – You may have to choose a size threshold so that you eliminate small “noise” components.
  – Also, the use of morphological operators may be helpful.
  – Draw a box around the segmented moving objects