For this assignment, I used my work from the past assignment (classification) to determine if I needed to search an image for circles or arrows around which to put boxes. If detect\_arrow( ) returned true, then I called box\_arrow( ), which attempts to draw a bounding box around all arrows in the image. It does this by using the probabilistic Hough Line Transform, which, as implemented in openCV, returns starting and ending points for each line it detects. I use the minimum x-value from that list as the leftmost coordinate of my box, and so on. For circles, I use the information I already had from the Hough Circle Transform to draw a box around each circle I detect. I do bloom out the bounding box by a few extra pixels to make it less cramped seeming.

As you will see, this approach works fairly well on most images, but struggles on a couple as well. Image 9, which is particularly busy, throws my arrow box for a loop. In addition, a couple of the images with multiple circles occasionally had one circle that was not boxed. A note about the arrow boxer: it draws a single box around all arrows in the images (instead of a box around *each* arrow in the image) because of how it is implemented. You will find that my approach produces a very tight bounding box when the image does not have several objects in it.





























