Window Detection

Overview of the procedure:

- Blur the input image using a gaussian filter.
- Get edges from the image using a canny edge detector.
- Apply an edge thresholding algorithm specifically made for this application.
- Join up disconnected edges with small gaps between them since they need to be connected but get disconnected due to the thresholding.
- Consider connected components removing some using constraints on the perimeter.
- Fill in remaining components.
- Use morphological opening to reduce the noise around the boundary.
- Apply constraints on area and perimeter to generalize for various shapes and keep rectangles/circles removing uneven shapes.
- Convert the window regions into boundaries and add these to the original image in red color to get the final image with players detected.

Steps in detail:

1. Apply a gaussian filter to the input image to get the blurred image.
2. Apply a canny edge detector to the input image to get an edge image (this is done from the non blurred image so it would have a lot of edges)
3. The edge thresholding algorithm basically keeps edges if they are near windows and removes them if they are elsewhere.
    a. Consider 4 regions for every pixel of a certain size, left, right, top and bottom.
    b. Calculate the average color intensities for all these 4 regions in the blurred image.
    c. If these averages are similar and they are also similar to the value of the current pixel then remove this pixel from the edge image since this is not one that is near a window, so it's not of our interest.
4. To connect edges with small gaps between them
    a. Consider 2 pixels above and below the current pixel, if one of the 2 above pixels are present and one of the 2 below pixels are present in the edge image then set this pixel to present as well.
    b. Similarly do it for the 2 pixels on the left and 2 pixels on the right of the selected pixel.
5. Remove connected components (edges basically) which don’t follow certain conditions
    a. If perimeter p < meanPerimeter/4.0 then discard it.
    b. If perimeter p > (1.5*meanPerimeter + maxPerimeter)/2.5 then discard it.
6. Apply a fill algorithm to fill in the various components that are connected and closed.
7. Apply a morphological opening to reduce the boundary noise on the filled components so they get smoothened out a bit.
8. Apply constraints on the area and perimeter in order to select objects that are rectangles, squares and circles (or similar to these shapes) and not long lines etc.

    a. Reject a region if area < meanArea/3.0
    b. Reject a region if area > meanArea*3.0
    c. Reject a region if area/perimeter < 3.0
9. Apply a morphological dilation and subtract the morphological erosion from this to get a boundary which I add to the original image in red color. This would give us the final image with red boundaries around the windows.