**Homeworks Explanation**

**Question 1:**

With the use of image subtraction, we can enhance the differences between the two images. We can see that most of the pixels had a significant change in their values (>10) as most of the image pixels are white. There are some areas that the pixels did not differ that much or they were the same (<10) as we notice that there are some black areas.

We used absolute difference to handle negative values. After the subtraction, we threshold the pixels so that everything that did not change will become black(0) and everything that had difference will become white(255). (question\_1.png)

**Question 2:**

1. We notice that the intensity of the colors is low in all three images, as we see that the components of the histogram are concentrated on the low (dark) side of the intensity scale. (question\_2a.png)
2. After global histogram equalization, we stretch the histograms to improve the contrast of the images. We notice a big difference in both the images and their histograms. Now the histograms cover a wide range of the intensity scale and there is a concentration on the high (light) side. This results in lighter images with higher contrast. Objects like the buildings and the trees are now visible and detailed. Many areas like the sky and the shiny parts of the road though, are modified without the desirable result. We lost most of the information there due to over-brightness. (question\_2b.png)
3. After locally adaptive histogram equalization, we have better results compared with the (b). Here the images are still dark but with a clear view of the hidden objects(buildings, trees, signs) and we do not have noisy light areas like the shiny part of the street and the sky. The histograms are similar with the original ones, but with better distribution over the intensity scale. This means that images are divided into small blocks (tiles). Then each of these blocks are histogram equalized as usual. To avoid amplifying noise, contrast limiting is applied. If any histogram bin is above the specified contrast limit, those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. The best parameters were clipLimit=4.0, tileGridSize=(5,5). (question\_2c.png)

**Question 3:**

1. We notice that the median filter effectively reduced the noise without blurring the images. It is very effective in removing salt & pepper noise. With kernel 3x3 , we notice that some details are better preserved (cable lines and tree branches) than with kernel 5x5. With kernel 5x5 though we notice a better noise reduction. With median filter, the center pixel value is replaced by **a value that is present in the surrounding pixels**. This is why it preserves edges. The bigger the kernel, the more blurry the result because we take into consideration more neighbor pixels for replacing so we have more blending of the colors. (question\_3a.png)
2. The weighted median seems to outperform the "normal" median with a (3x3) kernel performed in part (a) in noise reduction consistently. The noise reduction seems to be at least as good as the median filter with a (5x5) kernel. What stands out though is that the weighted median preserves sharp edges and features much better because it weighs more the closest pixels, so the replaced pixel will have a value closer to the nearest neighbors. This can be seen especially well at the fence in the foreground or the stoop of the house. So weighted median filtering outperforms the "normal" variants in both areas, at the cost of being more computation-intensive. (hw3\_building\_weighted\_median.jpg and hw3\_train\_weighted\_median.jpg)

**Question 4:**

We notice that after the selection of appropriate dilation or erosion, the image is sharper without losing the proportions of the objects depicted. The shapes are more robust with sharper edge lines. There are some points around the two humans that the lines became pretty sharp but we can now see clearly the borders and the text. Overall, we improved the image sharpness without opening or closing gaps between the objects. (question\_4\_3.png is with kernel (3x3)), (question\_4\_5.png is with kernel (5x5))

**Question 5:**

Image processing algorithm:

1. Edge detection with canny algorithm
2. Line detection with Hough Transform technique
3. Draw the lines on the image (question\_5a.png)
4. Calculate possible rotation (with the help of the line position) and remove it (question\_5b.png)
5. Crop any undesired white space that occurred with the rotation
6. Repeat steps 1-3 to the new rotated image (question\_5c.png)

With the use of Hough transform, we can detect all the lines, even the broken ones and we can redraw them again on the image. For finding the orientation of the image, we just need calculate the minimal difference between the line angles and the directions of 0°, 1/2 pi, pi and 3/2 pi. Then we take the median difference and we rotate with this value. Lastly, we perform Hough transform again to the processed image and we draw the resulting lines on it. (final: question5\_insurance\_form\_restored.jpg)

**Question 6:**

Coin detection algorithm:

1. Edge detection with canny algorithm
2. Circle detection with Hough Transform technique
3. Draw the circles on the image (question\_6.png)
4. Count and print the number of detected circles/coins

For counting the coins in the image, we use similar technique with question5. This time we use Hough transform to detect circles that are our coins. We count the circles to find how many coins are in the picture. Then, we draw the circles to highlight the coins.

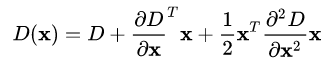
**Question 7:**

Puzzle solver algorithm:

1. Cropping of every piece and storing in a list
2. Template matching for identifying the correct position of every piece
3. Split the reference image size into a 4x5 grid (as our num of pieces)
4. Find where this piece belongs to on the reference image using template matching
5. Get the index using the found cell
6. Print the found index onto the output image
7. Save the resulted image (question7\_puzzle\_pieces\_numbered.jpg)

With this algorithm, many pieces were located correctly. However, some pieces were located in the same position as others, so the algorithm performed average.

**Question 8:**

1. Scale-space extrema detection produces too many key point candidates, some of which are unstable. Some of them lie along an edge, or they do not have enough contrast. In both cases, they are not as useful as features. The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures. This information allows points that have low contrast (sensitive to noise) or are poorly localized along an edge to get discarded. For low contrast features, we simply check their intensities. For each candidate keypoint, interpolation of nearby data is used to accurately determine its position. We use Taylor series expansion of the Difference-of-Gaussian scale space to get a more accurate location of extrema, and if the intensity at this extrema is less than a threshold value, it is rejected. The Taylor expansion is given by:  where D and its derivatives are evaluated at the candidate key point and x = ( x , y , σ ) T {\displaystyle {\textbf {x}}=\left(x,y,\sigma \right)^{T}} X = (X,Y,S)T is the offset from this point. The location of the extremum, x ^ {\displaystyle {\hat {\textbf {x}}}} X’ is determined by taking the derivative of this function with respect to x {\displaystyle {\textbf {x}}} X and setting it to zero. If the offset x ^ {\displaystyle {\hat {\textbf {x}}}} X’ is larger than 0.5 {\displaystyle 0.5} 0.5 in any dimension, then that is an indication that the extremum lies closer to another candidate key point. In this case, the candidate key point is changed and the interpolation performed instead about that point. Otherwise the offset is added to its candidate key point to get the interpolated estimate for the location of the extremum. To discard the key points with low contrast, the value of the second-order Taylor expansion D(X) is computed at the offset X’. If this value is less than 0.03 {\displaystyle 0.03} 0.03, the candidate key point is discarded. Otherwise it is kept, with final scale-space location y + x ^ {\displaystyle {\textbf {y}}+{\hat {\textbf {x}}}} Y+X’, where y {\displaystyle {\textbf {y}}} Y is the original location of the key point.

Difference of Gaussians has a higher response for edges, so edges also need to be removed. We use a 2x2 Hessian matrix (H) to compute the principal curvature. In the end, it eliminates any low-contrast key points and edge key points and what remains is strong interest points.

1. Compressed Sensing is a signal processing technique for efficiently acquiring and reconstructing a signal, by finding solutions to underdetermined linear systems. It asks if there is some minimal number set of observations that may be made. There are two conditions under which recovery is possible. The first one is sparsity and the second is incoherence, which is applied through the Restricted Isometric Property. RIP provides the necessary and sufficient requirements for the compressive sensing matrix. However, it is not robust enough for consideration under the noise. To introduce robustness in the recovery of all sparse signal vectors to noise, a necessary condition is required on the sensing matrix A. The definition of RIP says: matrix A of size n×N is said to satisfy the RIP with RIP constant R(k,n,N;A) if, for every x ∈ χN(k) :={x ∈ RN : ‖x‖0 ≤ k}, R(k,n,N;A) := minc ≥0 c subject to (1−c)‖x‖22 ≤ ‖Ax‖22 ≤ (1 +c)‖x‖22  For many CS decoders it has been shown that if the RIP constants for the encoder remain bounded as n and N increases with n/N→δ, then the decoder can be guaranteed to recover the sparsest x for k up to some constant multiple of n (ρ(δ)\*n). Compressed sensing is used in medical imaging, and seismic imaging, where the cost of measurement is high, but the data can usually be represented in a sparse format.
2. In image restoration, we try to restore a degraded image to its original form. To do so, we need to estimate the degradation model. One of the estimation techniques is the estimation by modeling. Environmental and physical conditions can cause degradations. We try to derive a mathematical **model** that can describe this physical degradation. For example, (Ex1)

a degradation model can be an atmospheric turbulence model with the form:

H(u,v) = e^-k(u2+v2)5/6  where k is a constant depending on the turbulence. With this model, we can create blur images depending on the turbulence we want to introduce. Low k indicates low turbulence so not much blurring and higher k indicates more blurry images (more turbulence).

Also, we can derive a mathematical model to describe motion blur. For example, (Ex2) using the fourier transform, we can compute an image that has been blurred by uniform linear motion. The degradation function is: H(u, v) = T/ π (ua + vb) \* sin[π(ua + vb)]e-jπ(ua+vb)



Ex1



Ex2