Too Lazy to Count (TLtC): A coin detection and recognition project.

With this project, I aim to tackle the problem of detecting coins in images, recognizing their respective values, and calculating the total value of coins present utilizing classic computer vision and image processing methodologies. There are two main aspects to this problem. First, a coin detection algorithm is necessary to detect the coins that appear in an image. Secondly, the program should be able to recognize the type of each detected coin, and to do so, a coin descriptor and a matching method must be implemented. The project focuses on euro coins and three assumptions are made:

A. The coins are on top of a planar, fairly smooth background, particularly a white background - although the code can be adjusted to accommodate different kinds of smooth backgrounds-.
B. The camera is perpendicular to the plane
C. There are no occlusions between the coins.

Three different kinds of detection methods were tested, as well as two different kinds of descriptors. More on that later.

<u>Creating a database of coin descriptors from a set of ground truth images:</u>

The first kind of descriptor tested was a simple radius descriptor. Each circular coin has a distinct radius different from the rest of the coins, so with that in mind, we aim to identify each coin by calculating it's radius and comparing it to a set of known radius values – the database of radius descriptors. In more detail, the extracted radius of a coin whose value is unknown is compared to the radius values stored in the database. The predicted coin value is the label of the closest database radius value (Euclidean distance). To create this database, eight ground truth images, one for each type of coin, were used. Each one of these images contains a different coin on top of a fairly smooth white background. An example image is visible below.

In order to detect the coin in each image, a detection algorithm based on image segmentation is utilized (the detection techniques are analyzed in detail further on). From the output segmented – binary image (background pixels set to black, coin pixels set to white) we extract the external contours of the coin. A minimum enclosing circle is fitted to the largest area contour returning the center coordinates (x,y) and the radius of the fitted circle. If the circle is ideally fitted to the coin these circle parameters are the same for the circular coin. The process is repeated for all eight images and at the end of it we have obtained 8 different radius values, one for each type of coin. These values synthesize the database of descriptors (a JSON file) and are in essence the ground truth reference values. It is important to mention that all eight images are taken with the same phone (Redmi Note 12 Pro) with the same camera settings and more importantly from the same height, resulting in images of the same scale. It is also important to note that every coin lays on it's 'heads' side but because the radius descriptor is independent from which side of the coin faces upwards (the radius is the same for both cases) the coins could have also been flipped and nothing would have changed. Despite that, we deal only with images where coins lay on the 'heads' side because the second descriptor tested is sensitive to coin flips.

The second type of coin descriptors that I experimented with was the ones that the SIFT algorithm produces. The radius descriptors are very simple and efficient but are not robust to scaling. Even the slight change in scale results in false classifications of the coins. It was also proven that they are very sensitive to detections that are not ideal – when the circle is not fitted ideally to the perimeter of the coin- as opposed to SIFT descriptors which are far more robust to all kind of changes. The database of SIFT descriptors was constructed in the same way that the database of radiuses was. Each one of the eight ground truth images was segmented, and for the coin detected in each image, the SIFT descriptors were extracted from the corresponding region (a set of 128D vectors –

concatenated gradient direction histograms for each coin). This second database is also in the form of a JSON file. For an unknown coin detected in an image, in order to predict it's value we calculate the SIFT descriptors of this coin region of interest and for each descriptor vector found we find it's two closest neighbors (using the L2 norm) in the set of database descriptors belonging to the first type of coin. To determine whether a match is good we use Lowe's ratio test which states that the distance of the descriptor from it's closest neighbor should be far greater than it's distance from it's second closest neighbor. We count how many good matches result from the comparison to the set of database descriptors belonging to the first type of coin and then we repeat the process for the remaining 7 sets of database descriptors corresponding to the remaining 7 types of coins. The type of coin whose set of database descriptors resulted in the largest number of good matches when matched to the descriptors of the detected coin is predicted to be the type of the detected coin.

To sum up, in the first case, for each type of coin a single value is used as a descriptor, the radius of the coin and in the second case, for each type of coin a set of 128D vectors obtained through the SIFT algorithm is used to describe it.

Coin Detection Methods:

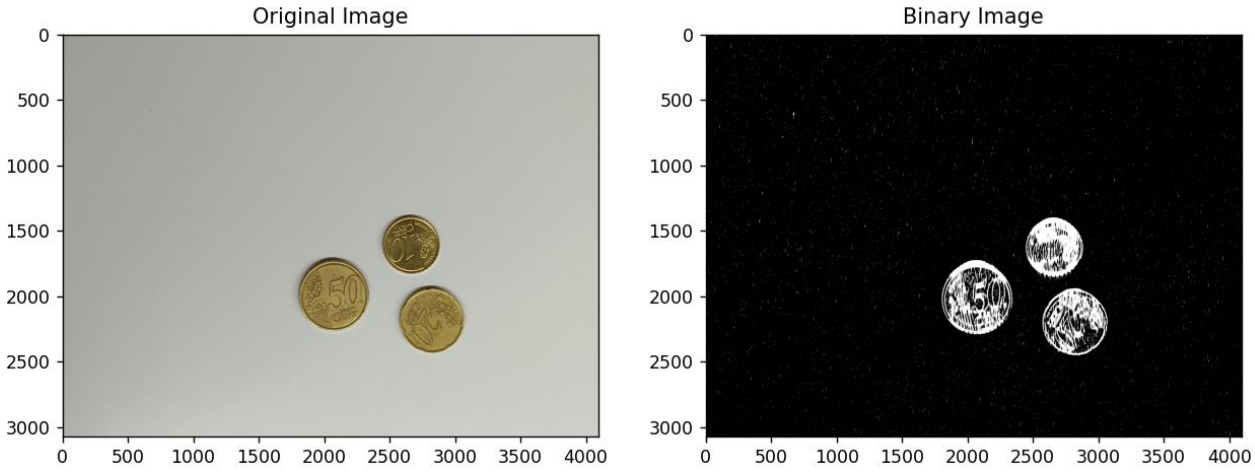
1. The first detection method that I tested was the following:
   a. The input image is converted to grayscale and is segmented by applying binary thresholding, setting pixel values that are above a specified threshold to 0 (remember that the background is white) and the rest to 255.
   b. External contours are extracted from the binary image that results from the previous step.
   c. A minimum enclosing circle is fitted to contours whose area is larger than a specified threshold value (this threshold is used to filter out contours that are not the perimeter contours of the coins that we are interested in).
   d. The circle parameters returned are used to draw the circles to the original RGB image, depicting the detection results.

   This method of detecting coins was proven unreliable because the fixed threshold value means that the slightest changes in illumination would result in a poorly segmented image, and consequently to poor coin detection.

2. The second method that was taken into consideration was the following:
   a. Convert the input image to grayscale.
   b. Reduce noise by applying a Gaussian kernel.
   c. Produce a binary image using adaptive thresholding.
   d. Use a morphological opening filter to reduce noise artifacts.
   e. Use a morphological closing filter to fill in the segmented coin areas.

f. Extract the external contours.
g. Fit a minimum enclosing circle to contours whose area is larger than a specified threshold value (again, this threshold is used to filter out contours that are not the perimeter contours of the coins that we are interested in).
h. Draw the circles fitted to the original RGB image, depicting the detection results.

This method is far more reliable than the previous one. It can adapt to illumination changed due to the adaptive thresholding used. A segmentation example utilizing the above method is shown below.



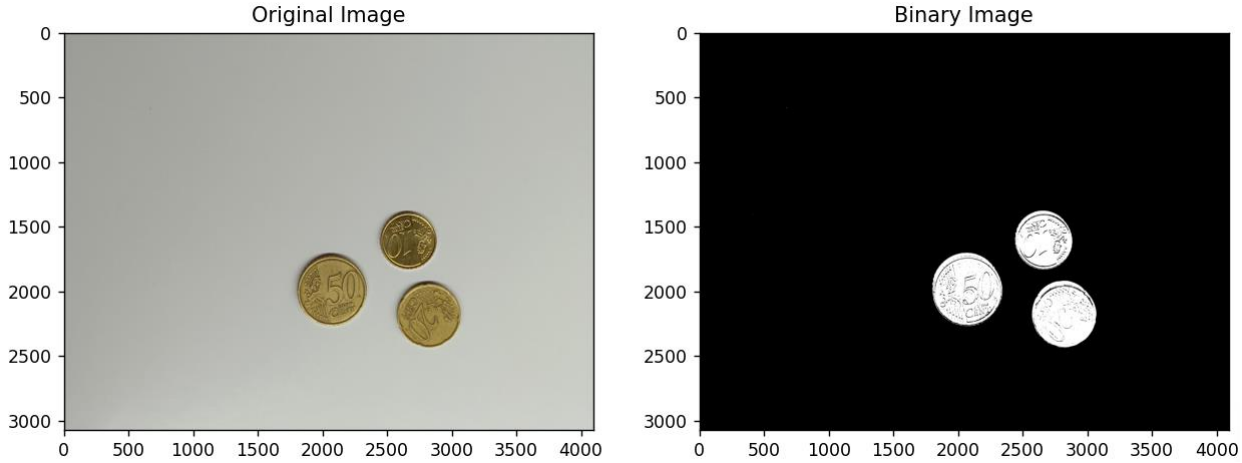
The detection results are visible in the following image.

We notice that the coins are indeed detected but the circles fitted on them and consequently the resulting detections are not ideal. This is happening because the segmentation is not ideal. In an ideal scenario every pixel of the background should be black and every pixel belonging to a coin should be white. This is not happening despite our best efforts and the use of morphological filters. Some background pixels are set to white, and some coin pixels are set to black. This leads to a problematic extraction of the external contours of the coins, and in the end to a not

ideal fitting of the minimum enclosing circles. Whether we should allow room for errors or whether we should only aim for ideal detections depends on how robust to this kind of errors is the descriptor used to recognize the coins. More on that later.

3. The third and final detection method that I tested is described in the following sentences:
   a. The RGB input image is reshaped into a 2D matrix where each row corresponds to an image pixel, and the three columns correspond to pixel intensity values in the red, green, blue channels.
   b. The KMeans algorithm is used to cluster the above dataset created by reshaping the image. The number of clusters is set to k=2 because we intend for a binary image output. The goal is to have coin pixels clustered in one cluster and the background pixels clustered to the second cluster.

   An example case is shown below.



The detection results are visible in the following image.

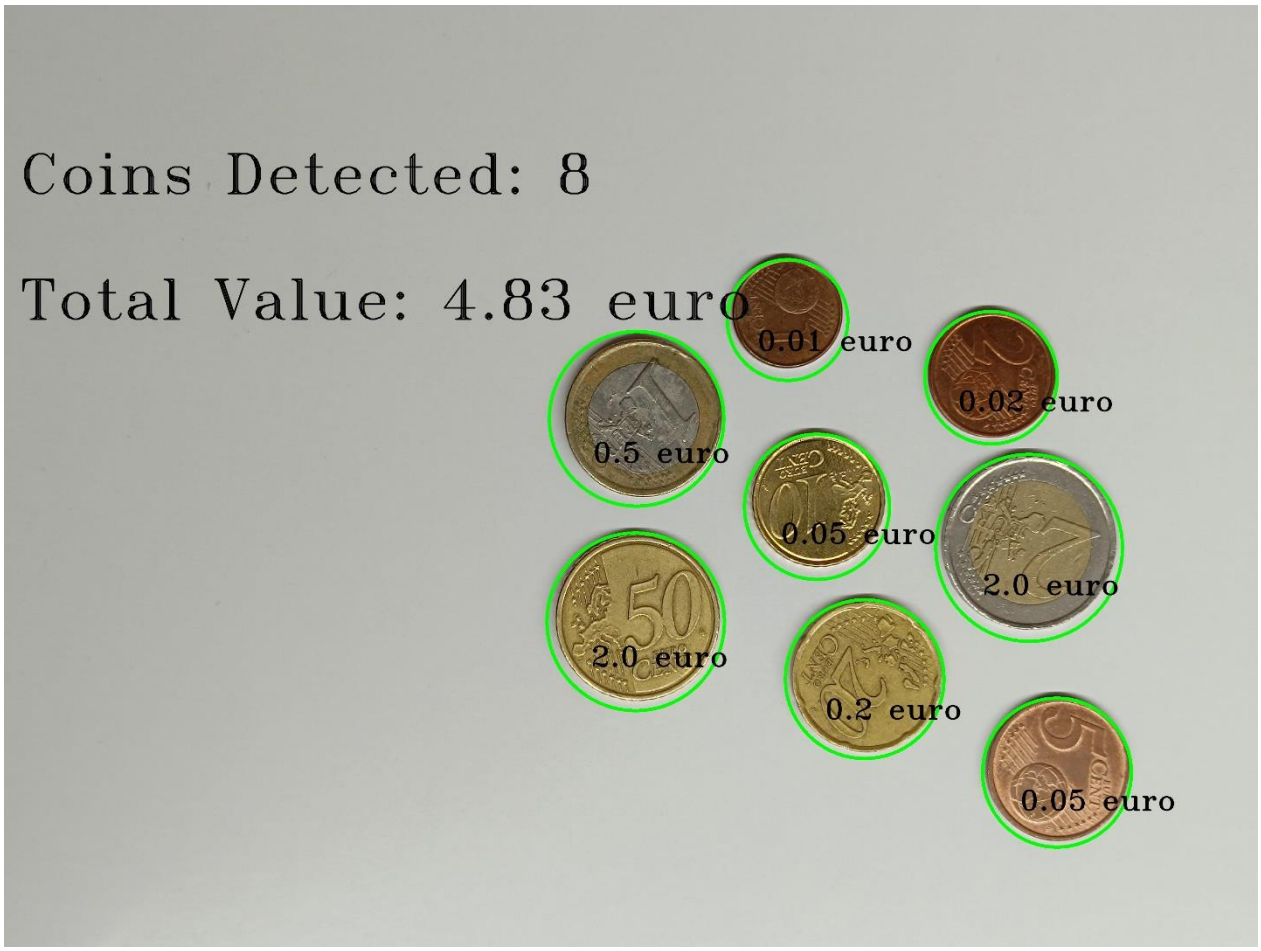
We observe that the segmentation result is better compared to the previous case, which is also true for the final detection result which could be characterized as close

to ideal as possible. The KMeans method is also expected to be robust to illumination changes because the segmentation is based on clustering pixels in the RGB space. We can confidently declare that the third detection method is superior to the second with the only negative being that the time needed to detect the coins is significantly greater in the case of KMeans. For the above picture the second detection method takes 0.08 sec to detect the coins whereas the KMeans method takes around 4.2 sec so, it might not be suitable for real time applications where time is of crucial importance.

The second method of detection was chosen to detect the coins in the 8 ground truth images used for extracting both the radius and the SIFT descriptors of each coin type, after being tuned to achieve very good detection results in these 8 images. The KMeans method could also be used without having to worry about it's time-consuming nature due to the fact that the database creation is always done offline.

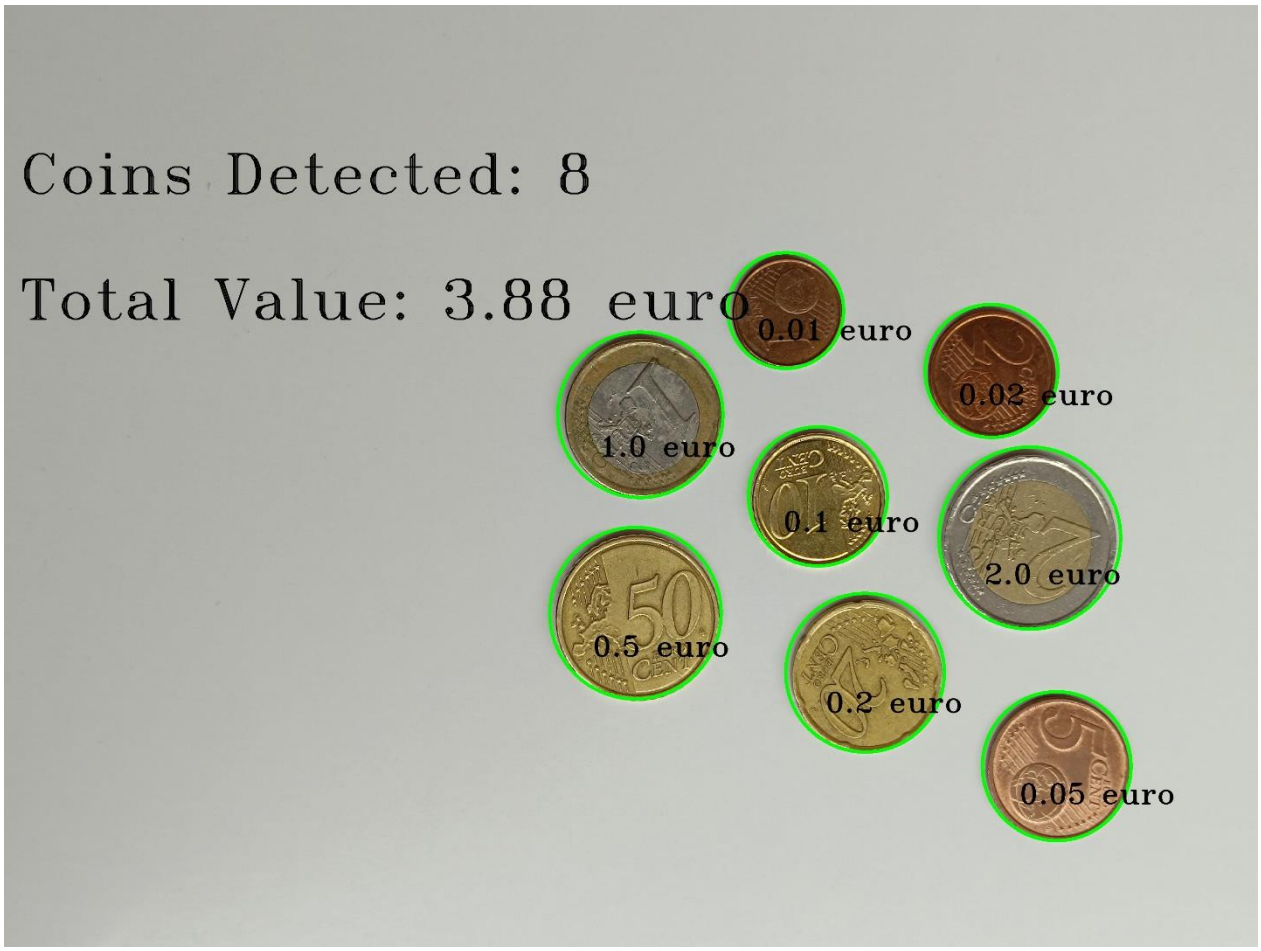
Let's focus on the first method of recognition utilizing the radius descriptors. The example input image is shown below. This image is of the same scale as the 8 ground truth images and taken under similar brightness conditions.

The results we get when the second method of detection is used (adaptive thresholding and morphological filtering) is:



All eight coins are detected but not all coins are recognized correctly – three of them are misclassified -. The 0.50 euro coin is recognized as a 2 euro coin, the 1 euro coin is recognized as a 0.50 euro coin and the 0.10 euro coin is recognized as a 0.05 euro coin. The above errors are not directly connected to the radius descriptor but are due to the segmentation method used. We have already discussed that the adaptive thresholding method detects the coins but the detections are not ideal. In other words, in some cases the circle is not fitted perfectly to the perimeter of the coin leading to a wrong calculation of its radius. For example, in the above image we see that the circle fitted to the 0.50 cent coin is bigger than the coin which results in a larger radius value than the actual value, which is then matched to the 2 euro coin radius in the database and that's why the 0.50 euro coin is falsely predicted to be a 2 euro coin.

But what if the KMeans method of detection is used instead?



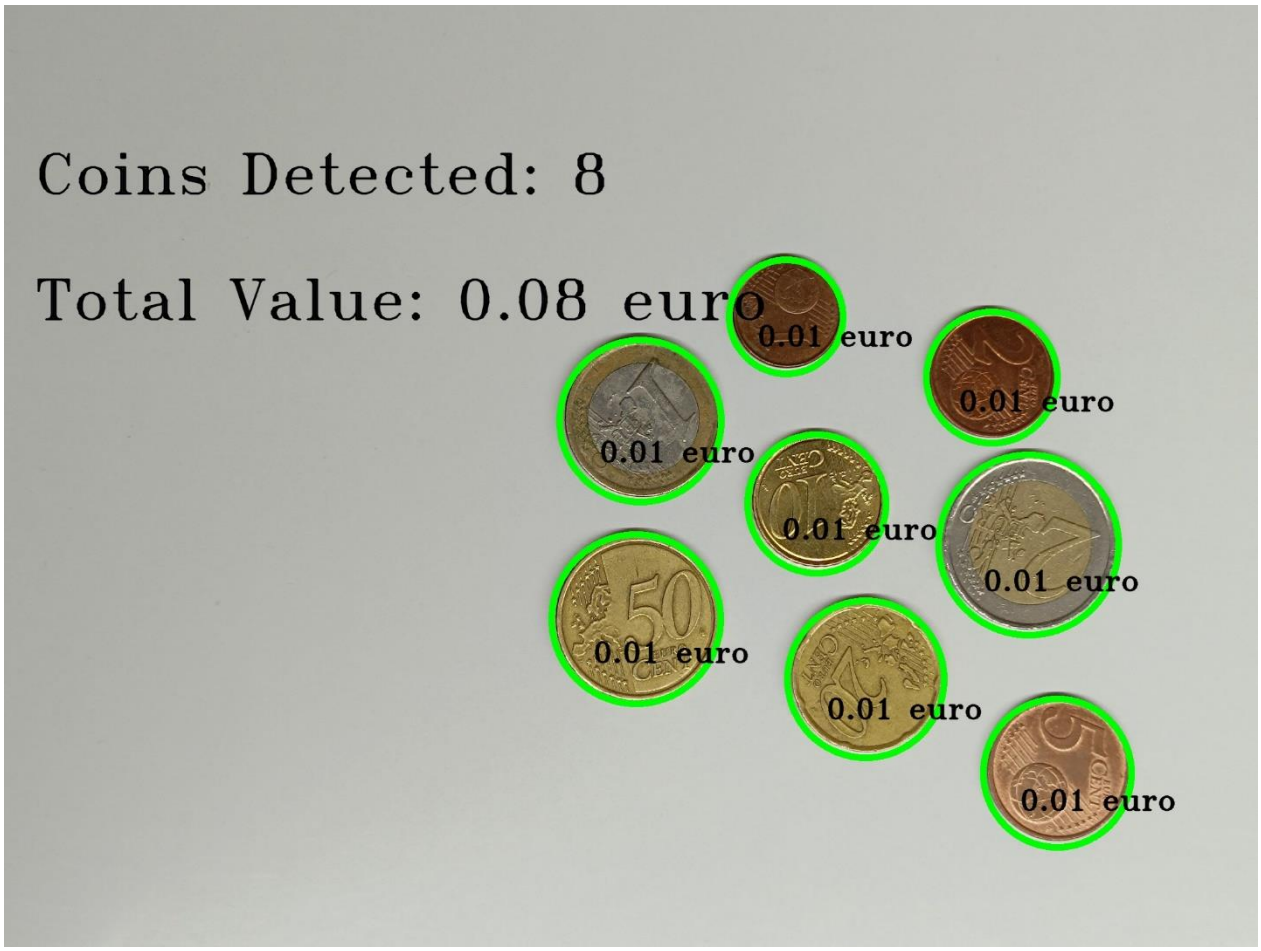
In this case we see that the each circle is fitted perfectly to each coin, resulting in a correct calculation of the coin radiuses and thus correct matches to the database values. Every coin in the image above is now correctly classified.

The sensitive nature of the radius descriptor requires perfect detection to yield correct results. That being said, we can conclude that when the goal is to recognize the coins through the use of the radius descriptors the optimal detection method is the one that applies the KMeans clustering algorithm.

How robust is the radius descriptor to illumination and scale changes?

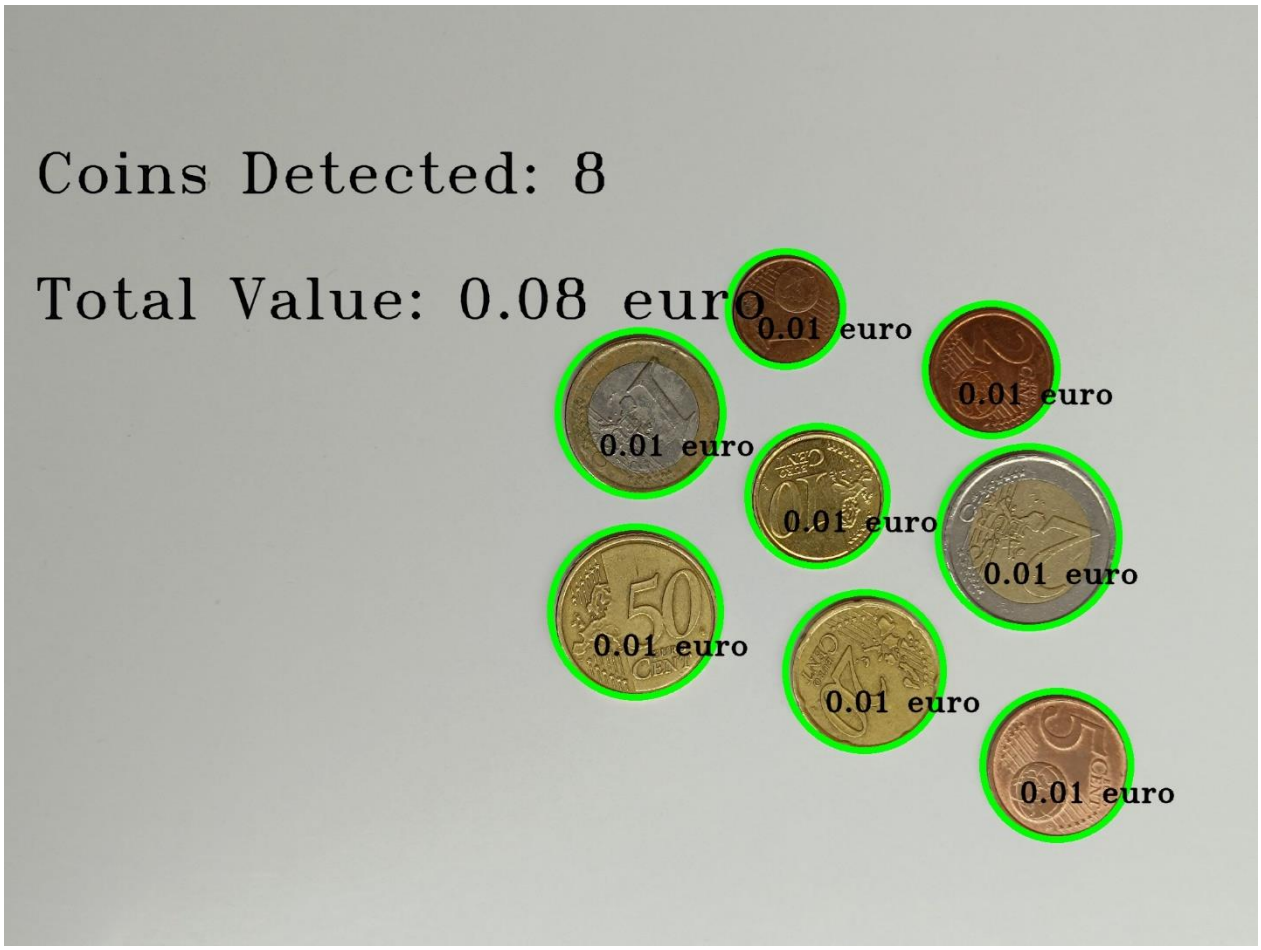
The following image is the result of scaling the original test image by a scaling factor equal to 0.50. In other words, the image below is a zoomed out version of the original image. It is obvious that the radius descriptor should not be used in such cases because the ground truth database was created by extracting coin radiuses in a particular scale, meaning that these values should only be considered ground truth when working in this exact scale. In all other cases – scales it is guaranteed that it will fail to recognize the coins.

The results that we obtain from using the adaptive thresholding method of detection and the radius descriptor method for recognition are visible in the next page. Again, all coins are detected. Please note that although it seems like the detection results are ideal, this is only because the width of the green circles is larger than before, some of the circles are still not fitted correctly but it is not visually observed. Having said that, that happens because both the width of the circles drawn and the text size is dynamically adjusted based on the input image dimensions. Although the recognition has completely failed as expected, the detection part of the program works (even though some circles are not fitted again ideally, which in this case does not even matter, even if all circles were fitted ideally the recognition would still fail due to scaling).



In order for the adaptive thresholding method of detection to be fairly robust in scale differences, a few parameters are set to be adjusted dynamically when changes in input image dimensions are noticed. To be more exact, the first parameter is the size of the kernel used for morphological closing. This value is calculated by multiplying a base size that was found to work well on the original scale with the scaling factor computed. The second parameter is the number of times the morphological closing filter is applied to the image which is similarly computed by multiplying a base number of iterations that was found to work well in the original scale, with the scaling factor. Finally, the third parameter is the threshold value that is used to filter out external contours that are not of sufficient area. This value is calculated as the multiplication of a base value by the scaling factor squared, as we are interested in area values. This dynamic adjustment works well when we are dealing with scaled versions of the original image, but it fails when we are dealing with images of different scale whose dimensions are similar to the dimensions of the original image. That is because the scaling factor is calculated as s = image_height / original_image_height (assuming uniform scaling in both dimensions - we can work either with the height or the width). So, if an image is of

a different scale but has similar dimension values to the original image, the scaling won't be detected, and the parameters will not be adjusted accordingly. Also as mentioned above, the scaling is thought to be uniform in both dimensions, so even if the image is a scaled version of the original image whose corresponding base value parameters we are dynamically adjusting, the detection will not be robust if this assumption is not true. That being said, we have observed another limitation of the second method of detection. It is only robust to scale changes under a set of very precise assumptions. On the contrary, the Kmeans detection method (although time consuming) is very robust to scaling. The following image shows the results that the radius descriptor yields for the scaled (s=0.5) version of the original image when coupled with the Kmeans detection method. Once again the every coin is misclassified due to the nature of this particular descriptor not being robust to scaling. The differences between the two detection methods are not visible because of the big line width.

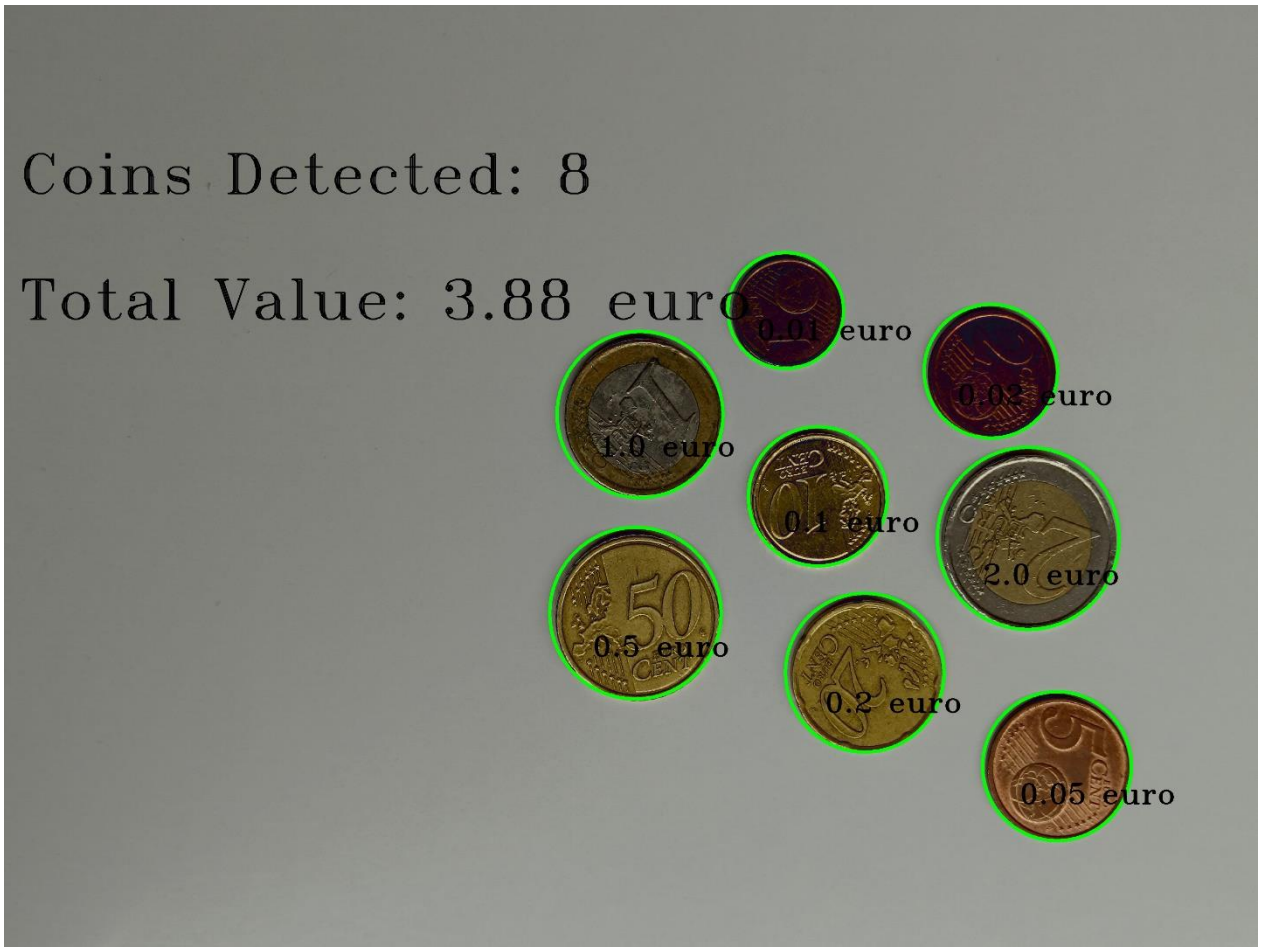


Let’s proceed with analyzing the radius descriptor performance under illumination changes. By applying a linear transform to the original image we reduce the brightness and we receive the following image.

The detection and recognition results that are produce by the adaptive threshold detection method and the use of the radius descriptors are visible below in the next page

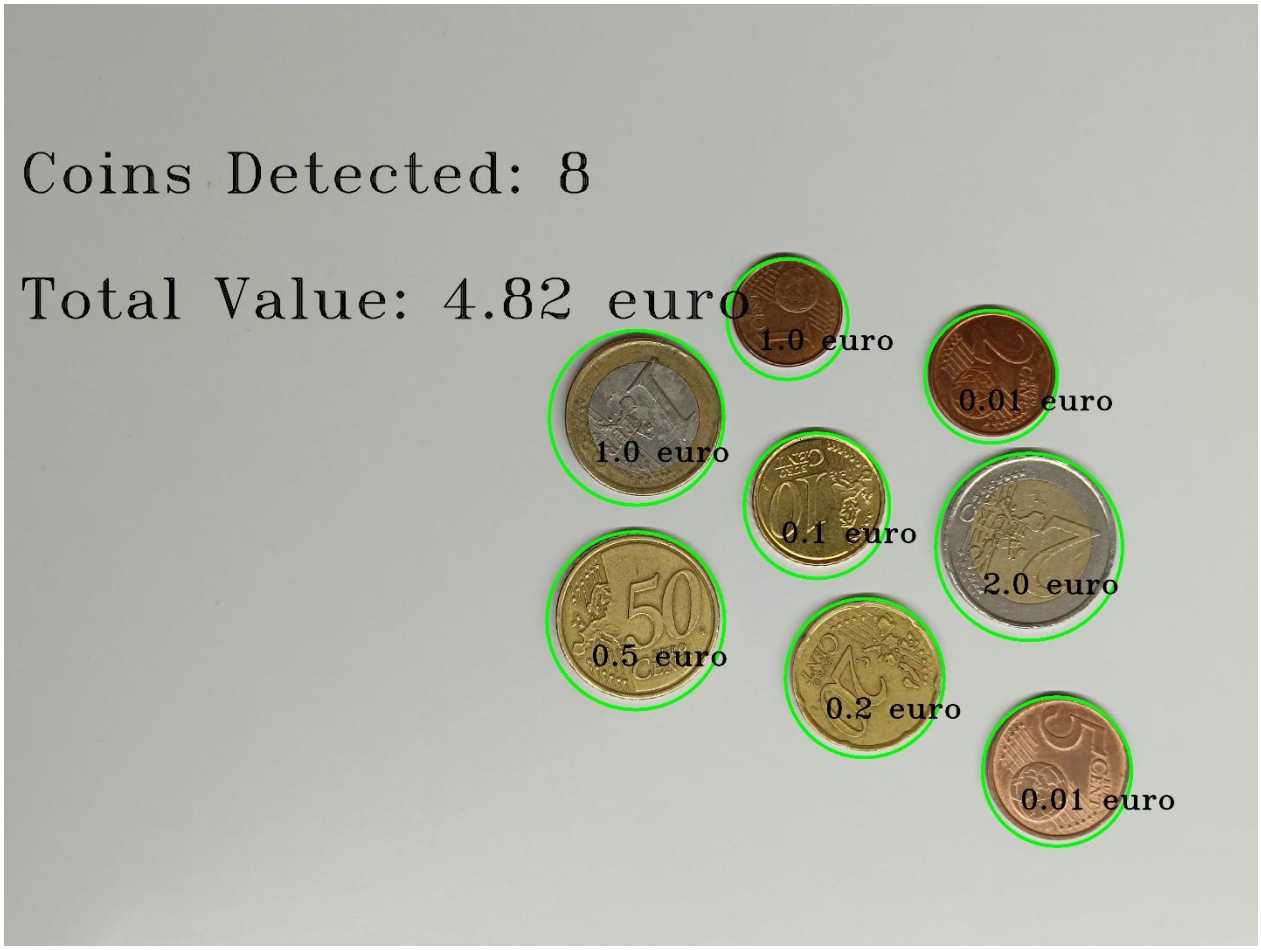
They are equal to the results that we received when tested on the original test image. This proves that both the detection method that used adaptive thresholding and the radius descriptors are invariant to illumination changes. To be more accurate, the radius descriptor is invariant to illumination changes and the adaptive thresholding method of detection is invariant to this type of linear transformations of intensity. The adaptive thresholding method might not work well with other kinds of brightness adjustments so we cannot conclude that it is generally invariant to brightness changes.

In the next page we can view the outcome for the KMeans method used with the radius descriptor as a mean to recognize coins. Once again, as in the original image, all the coins are precisely detected and correctly classified. It was expected that the Kmeans algorithm would be invariant to brightness changes due to the fact that it clusters pixels based on their color.

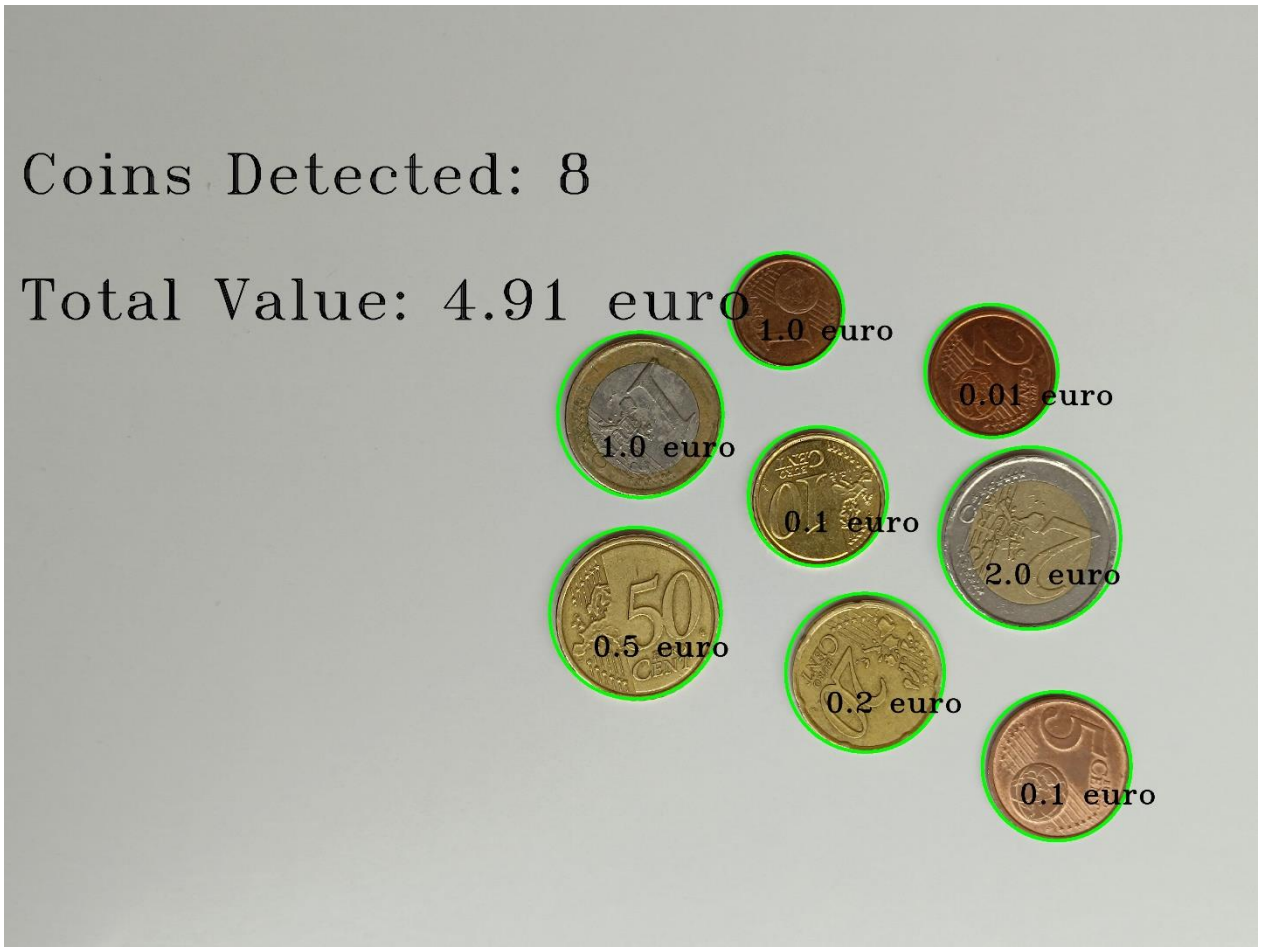

Coins Detected: 8
Total Value: 3.88 euro
0.01 euro
0.02 euro
1.0 euro
0.1 euro
2.0 euro
0.5 euro
0.2 euro
0.05 euro

SIFT descriptors

The image below that depicts the results of the SIFT descriptors matching coupled with the adaptive thresholding method when used on the original test image where there are no scale or brightness changes when compared to the images used to extract the SIFT descriptors database (small brightness variations might be present due to the reflective nature of the background but these are considered negligible).



All coins are detected (again, the detection is not the best with this method of detection) and all but three coins are also correctly classified. The 0.01, 0.02 and 0.05 cent coins are not recognized correctly and are predicted as 1euro, 0.01 euro, and 0.01 euro respectively. The results that we get when using KMeans for detection and not the adaptive thresholding method are the following.

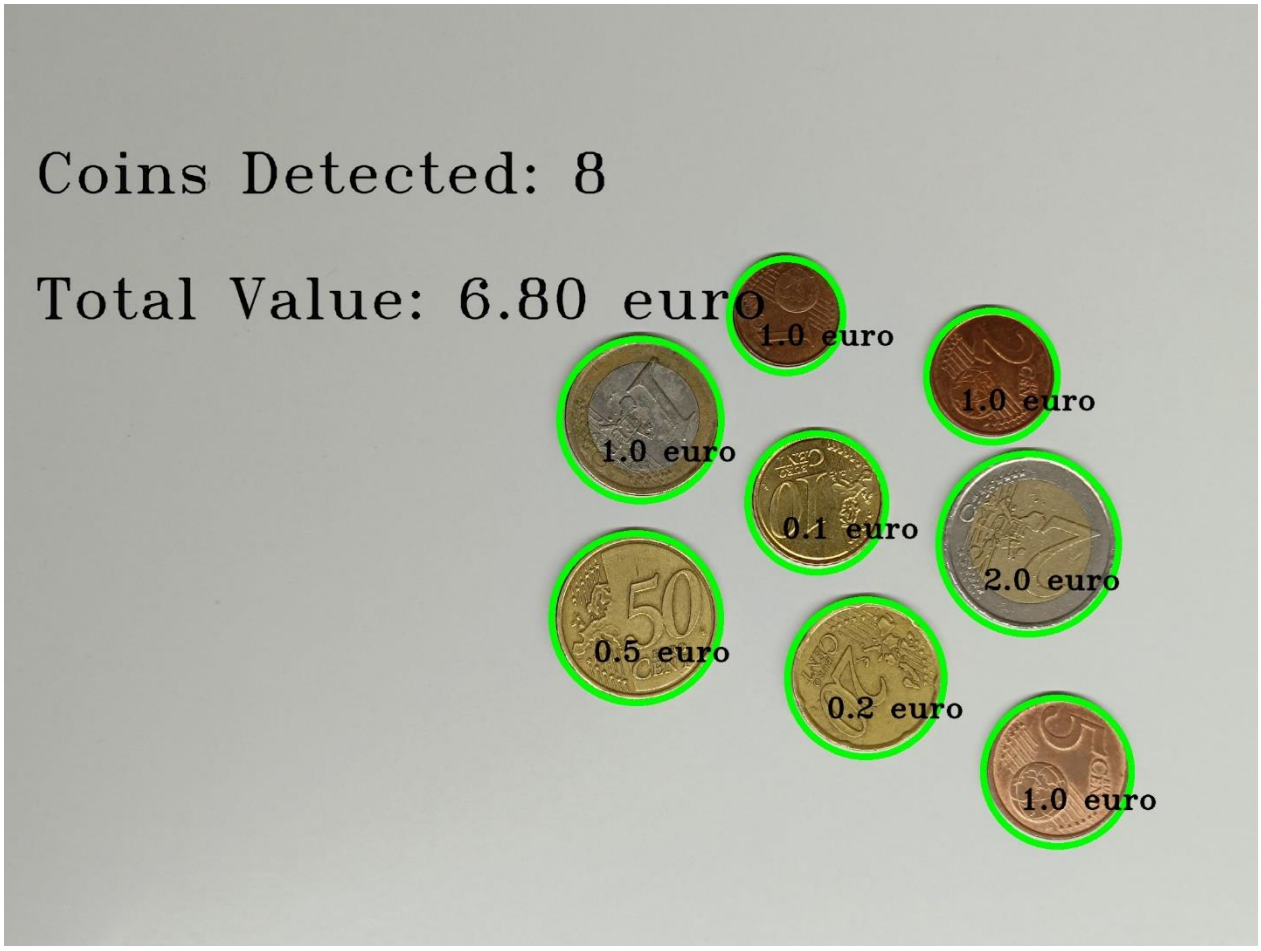


The only difference between this image and the previous one, is that the 0.05 cent coin is now falsely classified as 0.1 and not 0.01. In both cases the same three coins are misclassified. I expected SIFT to return better results and not to classify correctly only 62.5% of the coins. We notice a trend where the low value coins are being misclassified across all our tests. This highlights either the fact that the matching method (Lowe's ratio) used for SIFT descriptors fails when it comes to these low value coins meaning that another method should be considered, or that the SIFT descriptors in general is not the best option for describing these coins.

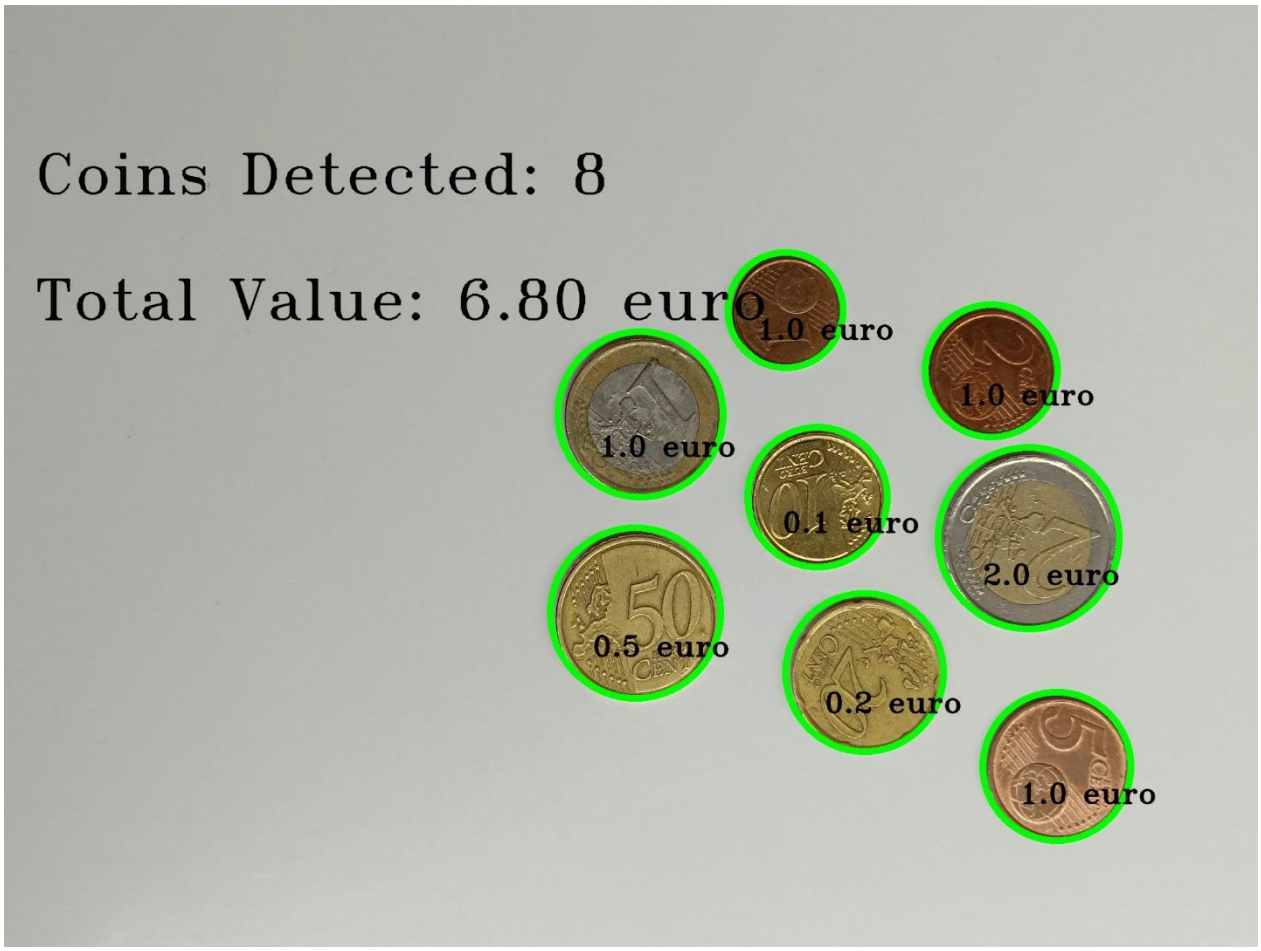
The reason the 0.1 cent coin is recognized as a 1 euro coin might have to do with the number 1 appearing on both coins thus having a similar subset of descriptors. The low value 0.01, 0.02, 0.05 coins are of similar appearance with the only difference between them being the number depicted on their surface. This might explain why the 0.02 cent coin is recognized as a 0.01 coin in both cases and why the 0.05 coin is recognized as a 0.01 coin in the first case. In both cases, 5 coins are correctly classified so even though the detection with KMeans is better than the one that results from adaptive thresholding we cannot conclude that KMeans is definitely the best choice as we did when we were using the radius descriptors, where when switching to KMeans, all coins were correctly recognized resulting in 8 correct classifications opposed to 5 correct classifications when

using the adaptive threshold method. This is happening because the SIFT descriptors are not as sensitive as the radius descriptors which require perfect fits of circles to the coins – perfect detections and consequently perfect, error free calculations of the radiuses. SIFT descriptors do not work with radius values so there are far more robust to imperfect detections especially in this case where the background is smooth and the same for both the test images and the ground truth images used to extract the SIFT descriptors. So, in this case we might actually consider the adaptive thresholding method to be the best choice because it is as mentioned, a lot faster than KMeans. But before we make any final conclusions let's see what happens when the scaling and the brightness is changed.

The results that we get when testing on the scaled (s=0.5) version of the previous images are the following -adaptive threshold detection method utilized-:



And with the KMeans detection utilized:



The results on the scaled image are the same for both detection methods. For the adaptive thresholding method, the differences we observe between the results on the original image and the results on the scaled version of it, is that the 0.02 and 0.05 cent coins are now recognized as 1.0 euro coins. The number of correct classifications still remain the same, as well as which coins are correctly classified. When it comes to the differences observed between the original and the scaled image when using KMeans, we see that the same is happening. The number and the coins that are correctly classified are the same, but the values of the 0.02 and 0.05 cent coins are now predicted to be 1 euro. The scaling did not interfere with the number of correct classifications but we observed some of the misclassified coin values being again misclassified but predicted to be a different value. Ideally this should not happen because SIFT is designed to be scale invariant. In practice though, SIFT is robust to scaling but not completely invariant.

We continue with the brightness adjusted test image.

Adaptive thresholding + SIFT:

Comparing the above results to the ones that we got when applying adaptive thresholding + SIFT to the original test image we see that the only difference is that the 1cent coin is now correctly classified as 0.01 and not 1 as it was on before.

KMeans + SIFT:

Comparing the results that KMeans + SIFT yields for the original and the brightness transformed image we notice that the 5cent coin is now predicted as 1euro whereas before it was classified as 0.1 and secondly, the 1cent coin is not correctly classified as opposed to being predicted as 1 euro before.

In both cases, the linear transform that was applied to the original image in order to reduce the brightness levels, seems to have sparked some differences between the results. The SIFT algorithm is designed to be robust to illumination changes but in practice as it happens with scaling, it is not completely invariant to them and that is probably why we notice these small differences.

The final comment I want to make about the SIFT descriptors is that the program developed should only be used with coins that are lying on their heads side. The tails side

should be the one that is visible, and that is because the SIFT descriptors were extracted from the tails side of each type of coin. The reason why I chose to extract the SIFT descriptors from the tails side is that, all euro coins share the same tails side (there is a slight difference in the tails side of coins minted before and after 2007 but I do not take it into consideration during this project, but in real case scenarios - applications it should definitely be considered). Contrariwise, the head's side is not constant. We can find a specific type of coin having a number of different scenes depicted on it's heads side mostly depending on the country of origin. One final thing to notice is that the level of corrosion in a coin might influence the predicted value when the prediction is made through SIFT descriptors matching. That is something that does not affect the recognition through the matching of radius descriptors unless we are talking about severe deformations which are not that common.

CONCLUSION

The radius descriptors method should only be coupled with the KMeans method of detection due to its sensitive nature in regard to error free calculation of radiuses which is something that cannot be achieved through the adaptive threshold method for detection. Furthermore, because of it being not scale invariant and producing very inaccurate results after the slightest change in scale, it should only be considered when the scale remains constant between the ground truth images and the test images. The necessary use of the KMeans algorithm for detection when recognizing coins through radius descriptor matching limits the practical applications as the time-consuming nature of the KMeans algorithm might make it not suitable for real time applications. To be fair though, the images that were used were of high resolution (3072 x 4096) so it might actually be feasible to use this detection and recognition scheme for real time applications if the images are of smaller size.

Instead, if the recognition is based on SIFT descriptors matching, both KMeans and adaptive thresholding can be used for detection due to the SIFT matching method being more robust to slightly incorrect detections. The KMeans method is robust to scale and illumination changes whereas the adaptive thresholding method has been proven to be robust to scaling under specific conditions and only proven to be robust to a specific illumination transform, with other type of transforms yet to be checked. So, if the final goal is a robust algorithm, KMeans is the best option. For real time applications the image dimensions should be limited in order for this solution to be viable. Finally, we should highlight that the SIFT descriptors are robust to illumination and scaling changes but not completely invariant.

Between the two descriptors, the best option is without a doubt the SIFT descriptors. With the radius descriptors being scale sensitive their practical applications are extremely limited. But for cases where the scale is to remain constant, they should be chosen over SIFT descriptors because their accuracy was found to be better. We observed the problematic cases were SIFT descriptors failed to classify the low value coins in most cases. But if the scale is constant, then the radius descriptors matching method should have no problem identifying these coins (as long as KMeans is used, and the detections are accurate).

In the future, I will build upon this work and experiment with different methods of detection and recognition. I plan to experiment with deep learning architectures and build a more robust and accurate model that is based on convolutional neural networks.

	Scale Invariant	Illumination Invariant	Time - consuming
Adaptive Thresholding	Under circumstances	Proven for specific type of transformations	No
KMeans Clustering	Yes	Yes	Yes**

	Scale Invariant	Rotation Invariant	Illumination Invariant	Invariant to the type of coin side visible	
Radius Descriptor	No	Yes	Yes	Yes	
SIFT descriptors	Yes*	Yes	Yes*	No	

*Not completely.

**At least for high resolution images