Indoor localisation application for visitors of the Museum of Fine Arts (MSK)

Cédric Bruylandt, Bart Ertveldt, Arthur Leclercq, Senne Loobuyck, and Ziggy Verhulst

Electronics and ICT Engineering Technology, Ghent University

Abstract. This paper presents a comprehensive study on indoor localisation for visitors of the Museum of Fine Arts (MSK) using computer vision techniques. The objective is to develop a solution that can accurately locate visitors based on video input. The paper introduces a methodology that involves frame selection, painting extraction, matching algorithms and a Hidden Markov Model.

Keywords: computer vision · painting detection · keypoint matching · localisation

1 Introduction

This paper presents a comprehensive study that provides a solution for Indoor localisation for visitors of the Museum of Fine Arts (MSK). By delving into the complexities of this problem, proposing innovative solutions, and presenting compelling results, this work offers valuable insights and contributions that make it worthwhile for readers to invest their time in understanding its content. Throughout the paper, the context is established by providing background information, defining key terms, and outlining the scope of the study. This contextualisation helps readers understand the relevance and significance of the different algorithms used in this research. This paper provides an extensive overview of the existing state-of-the-art approaches and methodologies related to frame selection, painting extraction, image matching and Hidden Markov models. The related work discussed in this paper has both benefits and shortcomings where certain methods are relatively fast but not 100% accurate. Through experimentation, analysis, and evaluation, this paper presents a comprehensive set of results. These results demonstrate the effectiveness, efficiency of the set of algorithms. The structure of the paper ensures a systematic flow of information, facilitating easy comprehension and navigation for the readers.

2 Method

In this section, the proposed methodology for visitor localisation using video input is presented. A comprehensive workflow diagram is depicted for every process to provide a clearer understanding of our approach.

2.1 Frame selection

General approach

First and foremost, the process of frame selection is essential to ensure that only frames containing a painting are considered. The workflow depicted in Figure 1 illustrates how this selection is performed. The blurriness of a frame is calculated using the variance of a Laplacian filtered version of the frame, as the Laplacian operator enhances high-frequency components. A higher variance will correspond with a sharper image. Because calculating the Laplacian of every frame would be overkill, it was decided to sample only every m frames (sample frames) and calculate the Laplacian only for those frames. After n sample frames, the one with the highest variance is saved as a good frame. The values for m and n are derived from the fps of the video to make sure our approach is adaptive to different fps values.

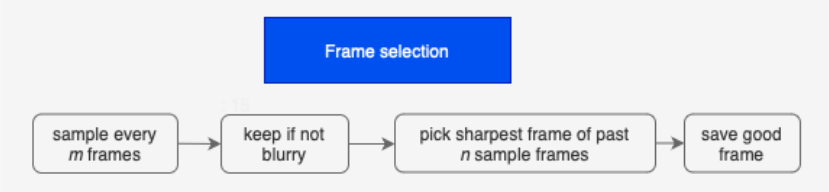

Fig. 1. Frame selection diagram

Calibration

Some of the video's are captured using a GoPro, which results in a fisheye effect. By calibrating the camera this fisheye effect can be undistorted. The calibration is done by using a video to calculate the intrinsic matrix. In this video, we observe a chessboard undergoing translation and rotation in various directions, causing the distortion to span across the entire image. This enables us to map the distortion throughout the entire video. The calibration method becomes more accurate as there are more frames used but this increases the processing time significantly. In order for the calibration to work properly there need to be approximately 30 frames. More frames do result in a more precise calibration, however this also increases the time to process the frames. Therefore the video is filtered until there are about 30 or 40 frames in order to have a good calibration in doable time. The filtering for good frames is done exactly like before in the painting extraction method, however with a further decrease in the number of frames.

The calibration itself is done by identifying the chessboard corners using OpenCV's findChessboardCorners function. Distortion will distort the image so the corners are not in a perfect square. These corners are then used for OpenCV's function calibrateCamera. This function then returns the intrinsic matrix as well as some other parameters. The distorted and undistorted frame can be found in Figure 2 and Figure 3. It is clear that in the distorted image there is a fisheye

effect, this is most notable in the background where the iron bar is visibly curved. In the undistorted image this iron bar is clearly straight. To undo the distortion on the fisheye frames OpenCV's undistort function is used, where the intrinsic matrix is used. In this example the intrinsic matrix is:

$$\begin{bmatrix} 615.26113583 & 0 & 633.39614894 \\ 0 & 616.62461909 & 370.71498895 \\ 0 & 0 & 1 \end{bmatrix}$$

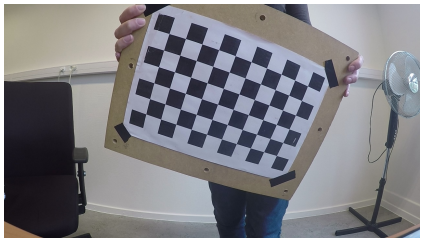

Fig. 2. Distorted image containing chessboard

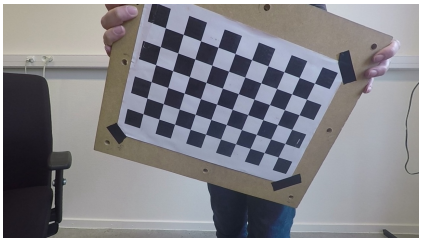

Fig. 3. Undistorted image containing chessboard

2.2 Painting extraction

After the selection process of only good frames, the next task involves finding a suitable method to extract the paintings contained within those frames.

Painting extraction on photographs of database

Initially, the paintings were extracted from the photographs captured in each hall. This step aimed to verify the effectiveness of our detection method on frames with higher image quality before proceeding to the evaluation of video frames from the MSK halls. The extraction method involved a sequence of operations, starting with a Gaussian blur, Canny edge filter and a dilation operation to enhance line thickness. Subsequently, contours are extracted from the resulting image and only the bounding boxes with lengths and widths greater than one-third of the dimensions of the entire image were retained. Next, the painting score of each remaining cutout is calculated using three metrics. The first one calculates the colourfulness of the image, which is an implementation of the paper presented in [1]. Secondly, the solidity of the image is calculated, which is defined as the ratio of the contour area to the convex hull area of an object. The convex hull calculates the smallest convex polygon that completely encloses the points of the contour and the contour area is the area of the bounding box derived from the contour. A higher solidity value indicates that the bounding box formed by the contour point closely matches its convex hull, while a lower solidity value corresponds to a more irregular shape. Lastly, edge density was determined as the ratio of edges

detected by a Canny filter to the number of pixels within the bounding box. The combination of the three metrics results in the computation of the painting score, which serves as an indicator of the likelihood of a bounding box representing a painting. Subsequently, the bounding box with the highest painting score is considered to have the highest probability of depicting a painting. Finally, to make sure a painting is selected, the best bounding box is only considered a painting if its painting score exceeds a chosen threshold.

The performance of this extraction method is evaluated by comparing the identified bounding box coordinates in the photographs with the ground-truth bounding box coordinates provided in the database log file. Intersection over Union (IoU) is used as a metric to assess this performance [2]. The IoU is calculated using the areas obtained through the Shoelace method [3]. As the main goal of this project is to locate visitors based on video input, it was decided that extracting only one painting from each photograph was sufficient. This choice was made to minimize the probability of False Positives (FP), which would result in only reliable extracted paintings to match with the database paintings, discussed in Section 2.3. In cases where multiple paintings are visible in a photograph, the extraction method aims to select the best IoU for that photograph.

Table 1. Results extraction method on photographs in every hall

Avg. IoU **FN**	**Hall number**							
	Hall 1	Hall 2	Hall 5	Hall 6	Hall 7	Hall 8	Hall 9	Hall 10
Avg. IoU	0.74	0.51	0.77	0.78	0.83	0.82	0.77	0.75
FN	1	7	0	0	0	0	0	1
	Hall 11	Hall 12	Hall 13	Hall 14	Hall 15	Hall 16	Hall 17	Hall 18
Avg. IoU	0.68	0.38	0.77	0.59	0.72	0.56	0.75	0.83
FN	1	0	0	2	8	1	0	0
	Hall 19	Hall A	Hall B	Hall C	Hall D	Hall E	Hall F	Hall G
Avg. IoU	0.62	0.34	0.45	0	0.55	0.57	0.28	0.64
FN	14	7	1	3	5	6	6	2
	Hall H	Hall I	Hall J	Hall K	Hall L	Hall M	Hall N	Hall O
Avg. IoU	0.28	0.34	0.4	0.21	0.09	0.82	0.34	0.32
FN	3	1	4	6	9	0	3	4
	Hall P	Hall Q	Hall R	Hall S	Hall V	Hall II	**TOTAL**	
Avg. IoU	0.66	0.96	0.77	0.52	0.84	0.48	**0.61**	
FN	3	0	0	20	0	1	**119**	

Table 1 shows the average IoU (Avg. IoU) and the number of False Negatives (FN) for each hall. It should be noted that for an extracted bounding box that results in a FN, the IoU is zero. The number of total FNs is 119 out of a total of 552 photographs. This indicates that our extraction method fails to identify a bounding box in approximately one-fifth of the photographs. Considering this,

the overall average IoU is reasonable, as it suggests that in cases where no FNs occur, the IoU values tend to be high. Furthermore, no FPs were found.

Painting extraction on frames of videos

The second step consists of applying our extraction method on the videos, i.e. on the frames selected by the frame selection method. As our method of the first step did not work so well alone here, an additional extraction method was needed to obtain a 'rough' contour, i.e. a 'zoomed' version of a frame that still encapsulates all the paintings within that frame. The zoomed version is then used as input to the first extraction method used on the photographs of database, the 'fine' method as depicted in Figure 4.

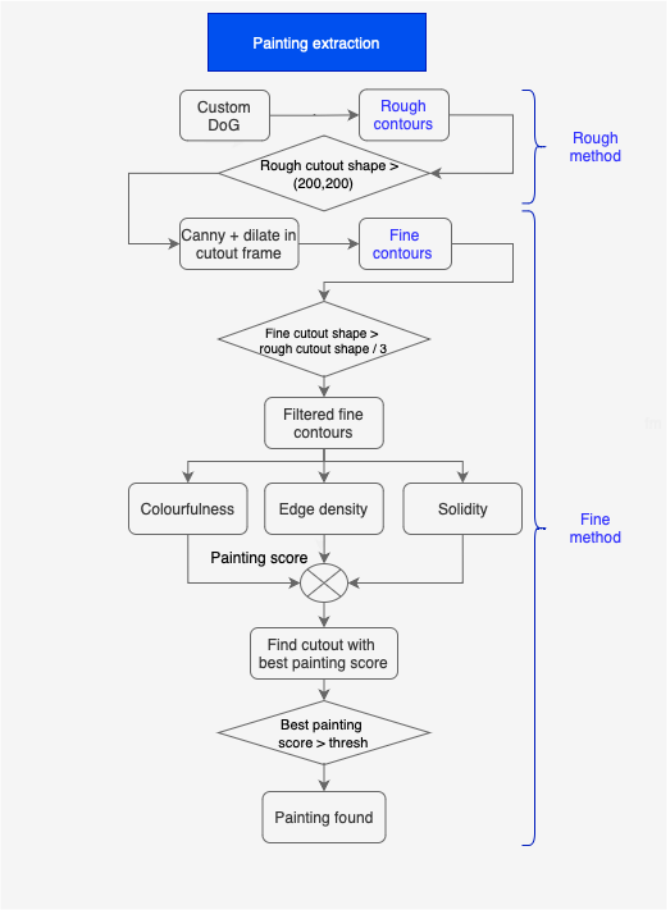

Fig. 4. Painting extraction diagram

As the first 'fine' method was already discussed earlier, our attention will now be directed towards the 'rough' method. In this regard, a custom Difference of Gaussians (DoG) function was developed, which after applying a Canny edge filter results in 'rough' contours. These rough contours can then be used by the 'fine' method to locate the paintings more specifically, as the custom function zooms in on the rough contours. This means that if the rough contours are wrong the whole painting extraction process is in danger, therefore the custom DoG function will be explained thoroughly.

The image is first rescaled (Figure 5), this is crucial for eliminating the requirement to adjust the filters according to the resolution, and also for minimising the image processing time. After resizing the grayscaled image, an unsharp maskening technique is applied to it in order to enhance the details , resulting in more pronounced edges. This is crucial as the next step is to apply a DoG-filter, where two Gaussian smoothed images are subtracted. If the images were not resized, those with varying resolutions would still use the same Gaussian filter size. However, due to the differences in resolution, these Gaussian filters would yield entirely different effects, leading to different results for the DoG.

An example is applied to 20190203_110204.jpg from the test set. The outcome of the DoG-filter can be seen in Figure 6. In essence a DoG-filter is a high-pass filter, the next step uses the thresholding technique of Otsu, see Figure 7. By applying this thresholding process, a clear binary distinction is made, conserving only the high-frequency elements of the image, primarily the edges. Otsu thresholding seeks to find a threshold where the sum of foreground and background spreads is at its minimum [4]. After applying Otsu-thresholding the edges of the paintings are visible but also details in the painting and of the surroundings.

Fig. 5. Original rescaled image

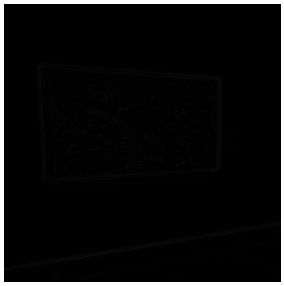

Fig. 6. DoG applied to resized and grayscaled image

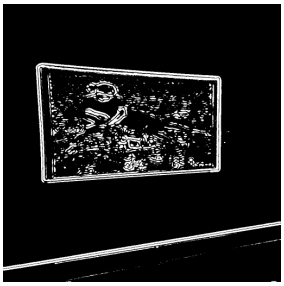

Fig. 7. Otsu thresholding applied to DoG-filtered image

The objective of this custom DoG method is to roughly find the edges, Therefore, the next step involves applying a huge Gaussian filter to blur the image significantly and getting rid of the details of the Otsu thresholded image.

This also means the image is very smeared, which can be seen in Figure 8. By applying another unsharp masking the smearing is limited. Now the image roughly contains a mask indicating where the edges might be, however the edges are not entirely clear. By applying a Gaussian-adaptive threshold the edges become more defined as the pixels are either black or white. To smoothen out the edges there is another Gaussian applied, which is followed by another unsharp masking to make the edges clearer. Finally, the final product of custom DoG can be seen in Figure 9.

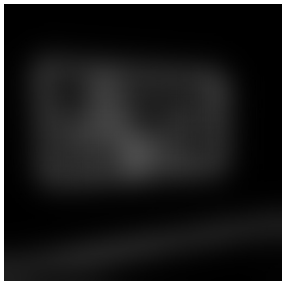

Fig. 8. Big gaussian applied to Otsu-thresholded image

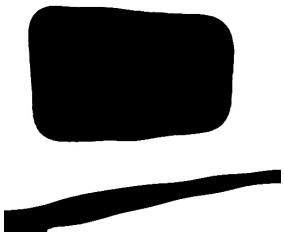

Fig. 9. End product of the function

Fig. 10. Bounding boxes after applying function

This method can now be used to find the rough edges and determine the bounding boxes like in Figure 10. Due to the many Gaussian filters and thresholds being applied, the method is not that fast and prone to errors. If the background of the wall is not plain and has a large amount of textures in it, the method will work less good. To conclude, the method is useful for finding rough edges but additional processing is needed, this additional processing is done by looking inside the detected contours.

2.3 Matching

After extraction the paintings in the frames, the next step is to match those images with the database paintings. In order to do this, the first step involves sharpening the images using a combination of Gaussian blur and additive weighting techniques. This is done to improve the quality of the images and increase the chances of getting good matches.

After sharpening the images, an object detection algorithm is used to detect keypoints and compute descriptors for each keypoint with distance measure the hamming distance. The descriptors are used in the k-Nearest Neighbors (k-NN) algorithm to find the best matches. To further improve the quality of the matches, the code applies Lowe's ratio test [5], which filters out matches that have a distance ratio larger than a specified threshold. This helps to eliminate

false matches and improves the accuracy of the final matching result. As second method a chi-square kernel is used to determine the similarity between 2 images. Afterwards both scores are combined by multiplication and results in the final score which is used to determine the best match.

In brief, the provided code utilises sharpening techniques and keypoint matching with an object detection algorithm, norm Hamming distance, and Lowe's ratio test to establish matches between two images. Sharpening is crucial as it enhances image quality and improves matching accuracy, especially for images captured with cameras of bad quality. Various algorithms are employed to detect keypoints and extract descriptors, while a matching algorithm is utilised to determine the matches. The outcomes obtained from these processes will be thoroughly examined and discussed in the results section of the paper.

Preprocessing the image

Sharpening an image involves enhancing its edges and increasing its overall contrast, resulting in a crisper and more defined appearance. One common technique for image sharpening is using a Gaussian blur. The Gaussian blur is a widely used image filtering technique that helps reduce noise and smoothens an image while preserving its edges. In order to sharpen an image using a Gaussian blur, we first convert the image to greyscale. This can be done using the cvtColor function in OpenCV, which takes the original image and converts it to greyscale using the BGR2GRAY conversion code. Once we have the greyscale image, we apply the Gaussian blur using the GaussianBlur function. This function takes three parameters: the input greyscale image, the size of the Gaussian kernel, and the standard deviation of the kernel in the x and y directions. The size of the kernel determines the extent of the blur and the standard deviation determines the intensity of the blur. After applying the Gaussian blur, we create a mask by combining the original greyscale image and the blurred image. This can be done using OpenCV's addWeighted function, which takes two input images, two weighting coefficients, and a scalar value. The first input image is the original greyscale image, and its corresponding weighting coefficient is set to 1.5. The second input image is the blurred greyscale image, and its weighting coefficient is set to -0.5. The scalar value is set to zero. The resulting mask image will have enhanced edges due to the high-pass filtering effect of the Gaussian blur. Finally, we apply the mask to the original greyscale image using the addWeighted function again. The first input image is the original greyscale image, and its corresponding weighting coefficient is set to two. The second input image is the mask image, and its weighting coefficient is set to -1. The scalar value is set to zero. The result of this process is a sharpened image that has enhanced edges and increased contrast. The amount of sharpening can be adjusted by varying the kernel size and standard deviation of the Gaussian blur, as well as the weighting coefficients in the addWeighted function. It's worth noting that sharpening an image using a Gaussian blur has its limitations. Sharpening an image too much can result in unwanted artefacts and noise, while sharpening an image that is already sharp can lead to over-enhancement and unrealistic results. Therefore,

it's important to use sharpening thoughtfully and to experiment with different parameters to achieve the desired result. In this case all the chosen weights above are achieved through experiment.

In conclusion, sharpening an image using a Gaussian blur is a powerful technique for enhancing edges and increasing contrast. It involves converting the image to greyscale, applying a Gaussian blur, creating a mask by combining the original and blurred images, and applying the mask to the original greyscale image. By adjusting the kernel size and standard deviation of the Gaussian blur, as well as the weighting coefficients in the addWeighted function, we can control the amount of sharpening and achieve the desired result.

Table 2. Comparison of Feature Detection and Description Algorithms

Algorithm	**Description**
ORB	Uses the FAST algorithm for efficient keypoint detection and provides scale and rotation invariance [6]. Suitable for real-time applications.
SIFT	Widely used feature detection and description algorithm known for its scale and rotation invariance [7]. It identifies robust keypoints that can be reliably matched across different views of the same scene or object.
AKAZE	Uses a nonlinear scale space representation, detects keypoints through local extrema, and generates feature descriptors using the Surflet descriptor [8]. It provides robustness to scale, rotation, and illumination changes and is designed for real-time applications.

Matching keypoints and descriptors

Keypoint matching is a fundamental task in computer vision and is widely used in applications such as object recognition, image stitching, and augmented reality. In this task, keypoints or interest points are identified in two images, and then these keypoints are matched to establish correspondences between the images. The approaches for keypoint matching are Oriented FAST and Rotated BRIEF (ORB), Scale-Invariant Feature Transform (SIFT) and Accelerated-KAZE (AKAZE) feature detector and descriptor, combined with various matchers like Brute Force (BF) and Fast Library for Approximate Nearest Neighbours (FLANN). These methods are explained in Table 2. As parameter the norm Hamming distance is used and afterwards the Lowe's ratio test is executed on the results of the kNN matcher to check the validity of the match. The score is a ratio of the amount of valid matches with the image out the database on the amount of valid matches the image out of the videos has with itself. The Chi-square (Chi2) [9] kernel is a similarity measure commonly used in machine learning and pattern recognition tasks. It is particularly useful when dealing with histogram-based data, where

each data point is represented by a histogram or a frequency distribution. The function returns a similarity score. The higher the similarity score, the more similar the feature vectors are according to the chi-square measure. This similarity measure varies between zero and one is subsequently multiplied with the score of the keypoint matching and results in the final score which will be used later on in the Markov algorithm.

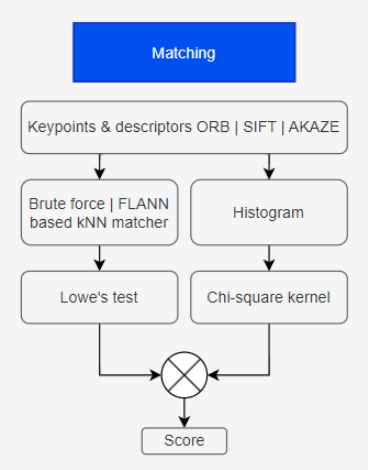

Fig. 11. Matching diagram

To summarise the workflow in Figure 11, keypoint and feature vector matching, norm Hamming distance, kNN-matcher, and Lowe's ratio test are a popular and effective method for establishing correspondences between two images. The scores of keypoint matching and feature vector matching will be multiplied and result in the final score. This method is widely used in computer vision applications such as object recognition, image stitching, and augmented reality which is ideal for matching paintings. There are a variety of possible combinations of algorithms to match paintings which will be discussed in the results.

2.4 Localisation

Hidden Markov models (HMMs) are tractable to capture long-term dependencies but intractable to compute the transition probabilities of higher-order process [10]. The HMM can therefore be used to handle long-term dependencies. In this scope, it is used as an extra layer to calculate the probabilities of the room in which the person is present. The first two stages have already been mentioned in Section 2.3.

After the matching, probabilities are calculated for each painting. This probability is then further adjusted by adding the Markov Model to it. This maintains

a long-term dependency on the previous position. Depending on the current room, the probabilities are multiplied by a different chance. Four different classes are distinguished in Table 3.

Table 3. Probabilities assigned to each room for the Markov model

Current room, compared to previous room	**Assigned probability**
Same room	0.40
Room along the logical path	0.35
Adjacent room to previous room	0.30
Other room	0.05

In addition to each class, a weight is assigned. These weights are not percentage-based but factorial. In this way, Markov adds a dependency for the chance of a new room. In this way, the chance for the matches for the paintings in the same room as before is 8 times greater than a room that is not adjacent to the previous room. The second largest factor is assigned to a logical path that is defined in advance. This is a path that we try to make logical. This has only a slightly larger factor than the adjacent rooms. The lowest factor is assigned to the rooms that are not adjacent or the same room. This has therefore been given the lowest chance.

This Markov model can try to compensate for certain errors in the previous step, namely the matching. Suppose the current room is room 5. The matching indicates that the probability is highest with a painting from room 19. The actual correct painting is in room 5 and gets the second best matching score. The chance for the painting from room 19 is multiplied by a factor of 0.05 and the painting from room 5 is multiplied by a factor of 0.40. As a result, the painting from room five can get the best final score. On the other hand, the probability for a non-adjacent or same room cannot also get a factor of zero. It can happen, for example, that a person walks quickly through the room and no painting is detected and now goes from room 5 to 7 without a painting being detected in room 6. Hence the factor is not set to zero, but rather a lower score is assigned.

To achieve this, a connectivity graph was created. This is represented by a matrix of size (A+1)x(A+1), where A is the number of different rooms. The zeroth row and zeroth column were left completely empty. In this way, room one could be accessed by index one. The connectivity graph determines which rooms are connected to each other. The room itself and the rooms that are connected to it, and are therefore adjacent, are assigned a value of one, while the other rooms are assigned a value of zero. Additionally, an emission graph is also provided. This graph assigns the weights to the connectivity graph. The weights have already been mentioned before and distinguish three classes. The combination of these two graphs forms the complete Markov model. The flow chart of the localisation phase is shown on Figure 12.

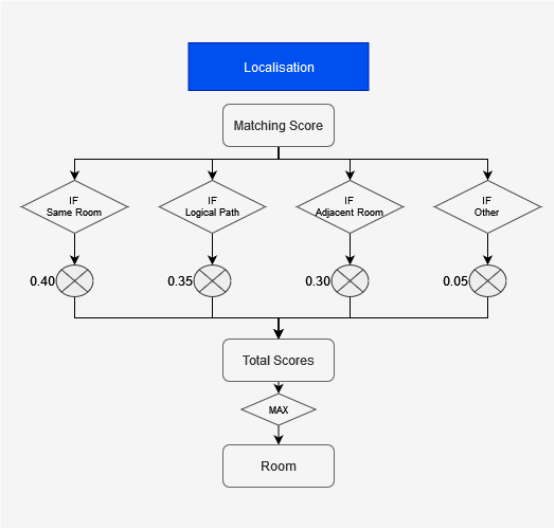

Fig. 12. Localisation diagram

3 Results

Evaluation of the model was performed on a system with 16 GB RAM and a Ryzen 7 5800U. Five different setups were tested for matching; AKAZE with descriptors, AKAZE with descriptors and histograms, ORB with descriptors, ORB with descriptors and histograms and SIFT with descriptors. A database is created by iterating over each of the 772 images, totalling 4.1 GB, and analysing them using the method and its metrics. The extracted features from each painting are subsequently stored into a JSON file, which acted as the database. Table 4 displays the time it takes to construct a database with the aforementioned methods and metrics and its total file size. As expected, large differences appear in both the time to build and file size when comparing each method. SIFT was quickly discarded as it was considered disproportionately large for this task.

Table 4. Comparing build time and file size of databases based on different methods and metrics.

Method	Metrics	Time to build	File size
AKAZE	Descriptors	7 min 50 sec	559 MB
AKAZE	Descriptors and histograms	10 min 44 sec	560 MB
ORB	Descriptors	1 min 20 sec	45 MB
ORB	Descriptors and histograms	2 min 23 sec	46 MB
SIFT	Descriptors	15 min	2104 MB

Upon discarding SIFT, both AKAZE and ORB with only descriptors or descriptors and histograms will be tested for speed and accuracy. The test video is a 717 MB 1920x1080 video with 30 FPS, resulting in 9024 total frames. Of these 9024 frames, 120 were filtered out and considered to be good quality. The algorithm detected 61 paintings in these 120 frames, with 38 having a sufficiently high painting score. This was reduced further by removing the duplicates (only keeping the best painting), resulting in 27 paintings cutouts of paintings that will be matched against the database. It is worth noting that the AKAZE database required 1 min 9 seconds to be loaded into the system memory, while the ORB database took 51 seconds: a negligible difference compared to their respective file sizes. The results of comparing the 27 paintings to each respective database are shown in Table 5, always using the same metrics to analyse the cutout paintings as the ones used in the database. As expected, ORB performed faster, though despite having a file size 10 times smaller than AKAZE, it only performed 4-5 times faster. Adding histograms did not improve the score for either method. AKAZE delivered the best results, having 2.5 times fewer misses than ORB. Based on these results, AKAZE with descriptors would be the preferred method for this task.

Table 5. Comparing time to match and matching scores of paintings against databases with different methods and metrics.

Method	Metrics	Time to match	Score on test
AKAZE	Descriptors	2 min 17 sec	23/27
AKAZE	Descriptors and histograms	2 min 20 sec	23/27
ORB	Descriptors	16 sec	17/27
ORB	Descriptors and histograms	31 sec	17/27

Table 6 and Table 7 offer a detailed analysis of our localisation system's performance. These tables highlight the number of 'good frames', images with contoured paintings, finalised extracted paintings, and paintings with high enough scores for consideration. They also provide the overall accuracy of our algorithm, categorized by the type of footage used: smartphone or GoPro.

'Good frames' refer to those that are advanced to the contouring stage. As evident from the tables, a substantial quantity of frames is eliminated at this step. Among these good frames, contour-identified paintings are extracted, leading to another significant reduction in frame count. These remaining frames are then forwarded to the matching phase.

An interesting observation made from the tables is the parallel accuracy between smartphone and GoPro footage, both registering a performance level of roughly 76.50%.

While the overall performance remains consistent across both types of footage, the accuracy within individual videos varies substantially. Some videos reach an accuracy level of 100%, whereas others, like MSK_07, show significantly lower

Table 6. Performance Evaluation on smartphone footage

Video	**Good frames**	**Frames with detected paintings**	**Paintings extracted**	**Correct**	**Accuracy (%)**
MSK_01	82	34	10	7	70.00
MSK_02	63	13	10	7	70.00
MSK_03	17	8	3	3	100.00
MSK_04	29	19	6	6	100.00
MSK_05	114	69	27	20	74.07
MSK_06	26	17	6	6	100.00
MSK_07	106	41	13	2	15.38
MSK_08	110	31	7	5	71.43
MSK_09	126	46	14	14	100.00
MSK_10	57	20	2	2	100.00
MSK_11	120	61	26	21	80.77
TOTAL	850	359	124	93	76.23%

Table 7. Performance Evaluation on GoPro footage

Video	**Good frames**	**Frames with detected paintings**	**Paintings extracted**	**Correct**	**Accuracy (%)**
MSK_12	90	36	12	5	41.67
MSK_13	113	44	13	7	70.00
MSK_14	6	4	1	1	100.00
MSK_15	150	56	33	26	78.79
MSK_16	85	36	6	5	83.33
MSK_17	35	22	6	3	50.00
MSK_18	77	29	10	10	100.00
MSK_19	275	126	59	48	81.36
TOTAL	831	353	140	105	76.64%

accuracy—around 15.38%. This variance is due to inadequate matching after the second painting, causing a jump to the incorrect room (room 19). The system struggled to return to the correct room, likely due to the influence of the Markov model.

However, the Markov Model demonstrates its utility in most videos, enabling the system to revert to the correct room following an error. It also rectifies scenarios where the initial matching was imperfect, ensuring the system remains in the correct room post-correction.

4 Conclusions

The process of extracting paintings is carried out in two stages. In the initial stage, a rough contour method is being used that zooms in on a frame and encapsulate all the paintings in that frame. Subsequently, these contours serve as input for a more refined extraction process that is capable of extracting a single painting. For the matching process, it was found that AKAZE is best suited for this application when compared to other methods such as ORB and SIFT. This is because AKAZE describes keypoints with higher precision than ORB, while also generating these keypoints and their descriptors more quickly than SIFT. The results implied that AKAZE is an ideal compromise between SIFT and ORB, balancing precision and speed effectively during the matching process. By incorporating a Hidden Markov Model, we account for long-term dependencies. This means that the final matching score relies on the data from the preceding room. Overall, the results indicate that our method gets an overall accuracy of about 76% to locate the visitors.

References

1. D. Hasler and S. Suesstrunk, "Measuring colourfulness in natural images," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5007, pp. 87–95, 06 2003.
2. "Intersection over union (iou) for object detection - pyimagesearch," https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, (Accessed on 06/03/2023).
3. "How to calculate area of polygon from unordered coordinate points. algorithm and code in python," https://www.geodose.com/2021/09/how-calculate-polygon-area-unordered-coordinates-points-python.html, (Accessed on 06/02/2023).
4. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
5. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
6. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," pp. 2564–2571, 2011.
7. J. Ruiz-del Solar, P. Loncomilla, and P. Zorzi, "Applying sift descriptors to stellar image matching," vol. 5197, pp. 618–625, 2008.

8. S. K. Sharma and K. Jain, "Image stitching using akaze features," *JOURNAL OF THE INDIAN SOCIETY OF REMOTE SENSING*, vol. 48, no. 10, pp. 1389–1401, OCT 2020.
9. H. Sadeghi and A.-A. Raie, "Approximated chi-square distance for histogram matching in facial image analysis: Face and expression recognition," pp. 188–191, 2017, 10th Iranian Conference on Machine Vision and Image Processing (MVIP), Isfahan Univ Technol, Isfahan, IRAN, NOV 22-23, 2017.
10. Z. Lin(B) and J. Song, "Neural Hidden Markov Model — link.springer.com," https://link.springer.com/chapter/10.1007/978-3-030-37494-5$_3$, 2019, [$Accessed 02-Jun-2023$].