Continuous room localization using painting detection

Bert De Saffel, Timothy Thiecke

Abstract—**This paper describes a localization method based on paintings from a museum, in this particular case The Museum of Fine Arts, Ghent. The method consists of three parts. The first part is painting segmentation which attempts to detect a painting from an arbitrary video frame using painting contours and a bounding box and transforms this painting into a standard format such that it can be used for analysis. The second part considers the transformed image and uses a brute-force ORB [1] matcher to detect key features and descriptors. These features are matched against a database using a linear lookup method. The third step uses the information of the painting, which contains the room it is located in, to mark it on a ground plan of the museum.**

I. Introduction

This paper introduces an algorithm to be used for indoor painting detection and localization applied to the frames of a video sequence. To achieve this goal we have constructed an algorithm that attempts to efficiently match detected paintings from a stream of images and to localize it on a ground plan.

This paper contains three contributions which will be discussed more in depth in the following section.

Our first contribution of this paper is a method for segmenting a given image or a video frame into a painting with its frame and the rest of the picture. This extracted painting is transformed to a standard format which serves as an input to our other contribution . It is inspired by the works of Canny (1986) [2] and Suzuki et al. (1985) [3] and combines them to allow robust approximation of a polygon and its extraction from the source material _ToDo: need paper, check wiki .

The second contribution of this paper is a robust algorithm for matching an extracted painting with an existing database of paintings. It is inspired by the works of Rublee et al. (2011) [1] which uses Brute Force matching based on ORB and should be more efficient than the existing work of Lowe (1999) _ToDo: check and add to library .

The final contribution of this paper is the usage of meta-data associated with the matched painting from the database. In our implementation, the meta-data is used to provide a strict way of localizing the painting and, by extension, the user on a floor plan (or its graph counterpart). The algorithm is not limited by this meta-data in its current state, as it can be expanded should the need arise.

The database of 688 paintings is matched every 50 frames with the videos running at 30 frames per second. All paintings are compressed to 1000 x 1000 pixels in order to speed up the matching procedure. In order to test the efficiency of our algorithm, a ground truth was established by taking a small sample of 30 paintings and manually labeling them. We achieved a segmentation accuracy of 88.57% and a painting matching accuracy and subsequently localization accuracy of 47.10%.

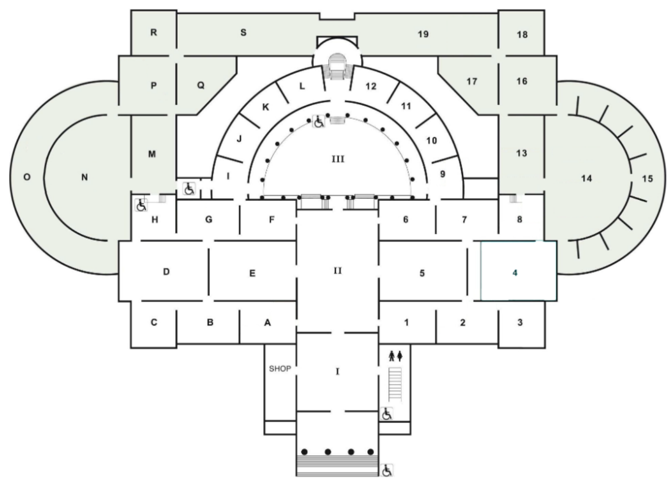

Fig. 1. A ground plan of The Museum of Fine Arts, Ghent.

This algorithm can have practical uses beyond the detection of paintings. As this method may be used in a similar fashion for detecting other types of objects or localizing a user in a different museum. This method is also viable for implementation on a smartphone where users may receive additional information on the painting or the room that they are localized in. Again the user must not be specifically located in a museum for this method to work as the database that the method builds upon can be exchanged between different runs of the algorithm.

A. Overview

The succeeding sections of this paper will discuss the implementation of the proposed algorithm. Section II will discuss the inner workings of the algorithm. Section III will describe the used performance metrics to evaluate the effectiveness of the algorithm, the experimental results and finish it up by presenting a qualitative analysis where the strengths and weaknesses of our proposed algorithm are laid bare.

II. Painting Detection

The core of this work is painting detection, which consists of two steps: painting segmentation and feature matching. The segmentation tries to extract paintings from an arbitrary image while feature matching attempts to find distinguishable features.

A. Painting Segmentation

The first step of the algorithm is the segmentation of an arbitrary video frame to detect a painting. A typical painting contains the art on its own enclosed by a painting frame. This painting frame causes a strong change in environment around its edges. For that reason, the Canny operator [2] is applied to the initial video frame, resulting in a new image which contains strong edges. Afterwards, we attempt to find contours using [3], which yields a vector of points for each contour. To make this process easier and more accurate, a dilation step is first applied on the edge image. We consider only contours which consists of 4 points and take the first 10 which have the highest area, as paintings tend to have a higher area than other quadrilaterals on a video frame. It is possible that multiple paintings exist on a single video frame. However, the algorithm's goal is to detect in which room the user is located, and not in particular which painting it is. Multiple paintings on the same wall belong to the same room. If the painting segmentation step selects either one of these paintings, the end result will be the same.

The detected painting is then transformed through a homography to a rectified version which serves as the input of the following stage, feature detection and matching.

B. Feature Detection and Matching

Feature detection and extraction is applied to the extracted painting from the segmentation phase and will be matched with an image from the database. Feature extraction is done with ORB [1].

Matching is done by invoking a matching procedure between the extracted keypoints and the keypoints of the database images. A match between descriptors is defined by its distance metric. The lower this number, the more likely that the match is valid. We calculate the sum of all matches and sort the matches between the source and database images by this sum. The first entry in this collection of matches is the image that is estimated to be a match for what is currently seen on-screen.

Additional meta-data is associated with the matched image and is used for the next phase.

C. Path Tracking

Once a painting is identified and matched, it can be localized on the ground plan. To achieve this, the ground plan is converted into a directed graph. The nodes of this graph are the rooms of the museum and the edges define the connections between rooms. When a user starts recording paintings, the matching algorithm will be performed on each frame and a location will be found. The graph is able to mark nodes in three distinct ways. A green node is the start of the path, an orange node is an intermediate path and the blue node is the end of the path. The path ends when the user stops recording. The path direction is also visualized by coloring the corresponding edges green. Note that when a cyclic path occurs which was walked in both directions, information of order is lost.

To illustrate the path tracking algorithm, a small segment consisting of rooms 1, 2, 3, 4, 5, 6, 7 and 8 are converted into such a graph and is show on figure 2.

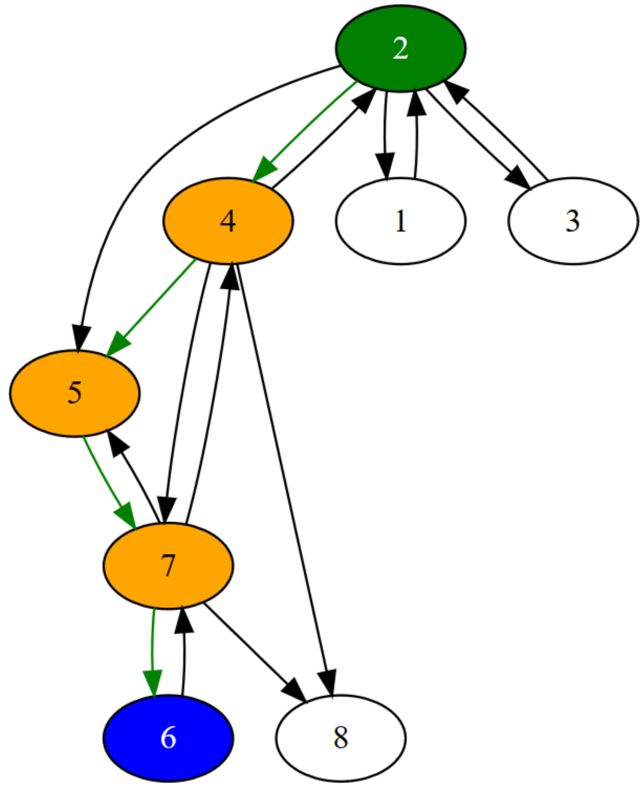

Fig. 2. Path tracking using a graph. A green node marks the path's start, orange nodes the visited rooms, and blue the last node that was visited. Edges in green denote help visualize the path.

III. Results

This section describes the experimental results of the final recognition system. It includes the performance of the individual steps: painting segmentation, the matching algorithm and room localization.

A. Dataset

The main dataset consists of 688 pictures of all art items in the museum which functions as a database. These pictures are taken at eye-level height and each picture contains one or multiple art items. One part of the dataset is taken with a Nokia 7 Plus camera, which offers a base resolution of 3024 by 4032 pixels and the other part is taken with a Samsung A3 2016, which offers a base resolution of 3096 by 4128 pixels. All pictures in the dataset are compressed to 1000 x 1000 pixels. To reduce the load time of this dataset, a prebuilding stage was implemented. This stage reduces each image to a collection of interest points and corresponding descriptors for these interest points as generated by the ORB [1] algorithm.

B. Test set

Apart from the dataset, two different testing sets exist to evaluate the algorithm. The first testing contains still images of various shots at the museum. This dataset contains more difficult examples such as steep angles or hard to detect paintings. A random sample ($n = 30$) was selected and labeled manually. This first test set is used to evaluate the segmentation and matching part. A second testing set are various videos which emulates a person recording paintings while moving trough the museum. Similar to the first test set, a small segment (1 minute) of the video was taken and labeled manually. This last test set is used to evaluate path tracking.

C. Segmentation Accuracy

The random sample from the test set was first manually segmented to indicate a perfect segmentation. This results in 4 coordinates of a polygon which represent the ideal polygon. Afterwards, the segmentation is done automatically by the algorithm, which also gives four coordinates of a polygon. To illustrate, both polygons are shown on figure 3. To measure the similarity of these two polygons, and thus the correctness of the segmentation, we first calculate the intersection area A_i and the area of the ideal polygon A_{pi}. The ratio of A_i to A_{pi} describes the closeness of two polygons with 100% being a perfect match and 0% meaning there is no intersection at all between the two polygons. There is one case where this statistic does not work. When the ideal polygon is fully enclosed by the predicted polygon, A_i will be equal to A_{pi}, resulting in a fake perfect match. To prevent this, the roles of the green and red polygon are switched such that we now consider the area of the predicted polygon, A_{pp}, instead of A_{pi}.

Using this metric, the segmentation method achieves 88.57% correctness score. Due to the use of the Canny edge detector, the segmentation works fairly well when the painting frame and the background wall differ greatly in color intensity. Problems start to arise when shadows are visible (figure 4) or when the painting frame is not fully visible. In the former case, the shadow and the background wall also differ in color intensity, resulting in an edge. These edges usually extend the painting frames and because the contour with the largest area is sought after, the segmentation step will include the shadow as part of the painting.

D. Matching Accuracy

The matching algorithm has to be evaluated manually by comparing the matcher's result. The correctness of the matching algorithm is simply the ratio of the correct matches against the false matches.

To evaluate the room localization, a sample of the video dataset was taken. The generated path is compared against the actual path.

Because our method relies heavily on edge detection, there are cases where this could have a negative impact. In many cases, there is a shadow underneath the painting, as shown on figure 4.

E. Qualitative analysis

In this subsection, we will present a qualitative analysis of our algorithm. We will discuss its strengths, flaws and, with each point, present an example case to help as a visual aid. The flaws in particular help paint a picture of what can be done better in a future iteration of the algorithm.

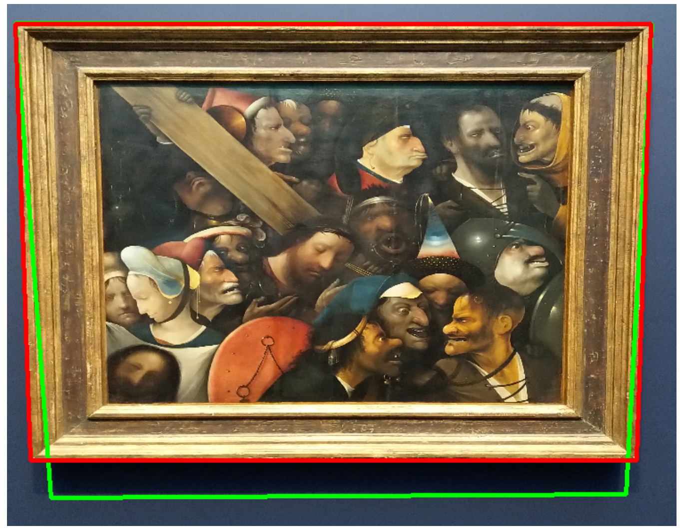

Fig. 3. A comparison of a manually selected polygon (red) and the polygon found by the segmentation algorithm (green).

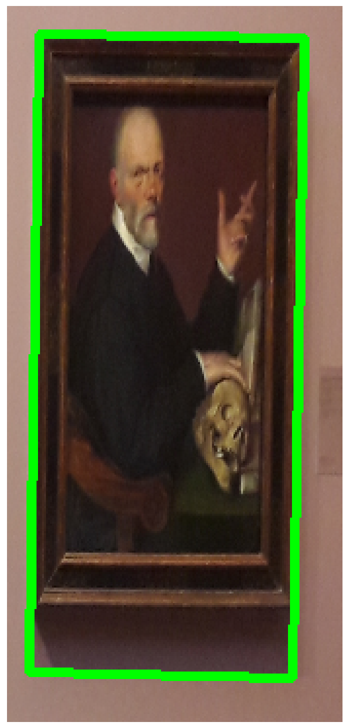

Fig. 4. An example of a shadow underneath the painting. This usually results in the segmentation algorithm to include this shadow as part of the painting because of the strong edge.

One of the algorithm's major strengths lies in its simplicity. The block diagram shows a very linear approach to the problem with concrete begin- and endpoints. Though machine learning is often used in the field of computer vision and has a lot of benefits, the algorithm does not employ it. We do not deny its usefulness and believe that it may be beneficial to implement it in a future version of the algorithm _ToDo: waar nodig?, kan zijn dat ML gebruikt wordt bij matching . Even though the algorithm's simplicity is presented as one of its strengths, it can also be considered as one of its weaknesses, much like a double edged sword.

Furthermore, the algorithm works relatively fast for matching and determining the location of a potential user using a single image. Table _ToDo: tbl gives an overview of the average elapsed time when running the algorithm on a single image. _ToDo: chart met timing van algoritme voor 1 afbeelding

In addition, the use of ORB causes the results of each program run to be invariant given the same input image, implying a deterministic nature. This is of course on the premise that certain parameters such as the Canny thresholds remain the same between runs. Figures _ToDo: fig and _ToDo: fig shows how two images have the same match and detected keypoints between two runs. However, due to not detecting different potential candidate matches between runs results in the algorithm never being able to present a different solution for the given image.

On top of that, the algorithm is resilient when it can not successfully segment an image. The default behavior is trying to find a frame, extract it, and using it as the input of the matching stage of the algorithm. If no frame can be found, rather than ignoring the current video frame or image, we supply the entire thing to be matched. This may slow the matching stage down by a small margin but has the benefit of still being able to spit out a potential positive match. Figure _ToDo: fig shows an example where an entire frame is supplied to the matching function, but still manages to find a correct match. This resiliency comes at a price. The matching phase performs slightly worse, as the frame may contain additional features that can help with matching a painting.

As was hinted in the previous point, supplying a full frame or image to the matching phase may slow things down. The input image's size definitely affects the time needed to complete in a negative way. Therefore, images used for the segmentation and matching phases are resized to between `25%-50%` their original size. This may result in some loss of precision but should not affect the overall efficiency of the algorithm. Table _ToDo: tbl shows that resizing an image from the query set to a quarter of its size quadratically decreases the amount of pixels needed to evaluate. The same resizing is applied to the building stage of the database.

Another problem arises due to an assumption in the matching stage. A match is determined by the smallest sum of distances calculated over the matches between two images. This assumption works well when the amount of matches is fixed between every match between the query image and the training set. ORB detects keypoints and associates descriptors up to a maximum of 200. But what if it can not detect a lot of keypoints, maybe even none? Figure _ToDo: fig shows how paintings that have a relative 'flat' surface without discerning features (even to the naked eye) have trouble being matched to a picture in the data set. Suppose that the segmentation phase detects such a painting and supplies it to the matching phase. A cascade of erroneous matches and room localizations may occur. The inverse is also true, as having an entry in the data set with few keypoints can result in a painting with many keypoints being wrongfully matched with a 'flat' one.

Figure 5 shows the flaws of the segmentation phase. If no quad can be approximated from the painting then there is a chance that polygons with a smaller surface area may be erroneously detected and fed to the matching stage. The problem lies in the combination of using the Canny operator along with the approximation of quad polygons. Our algorithm is unable to deal with 'valid' paintings that have an approximated polygon with an amount of sides higher than 4.

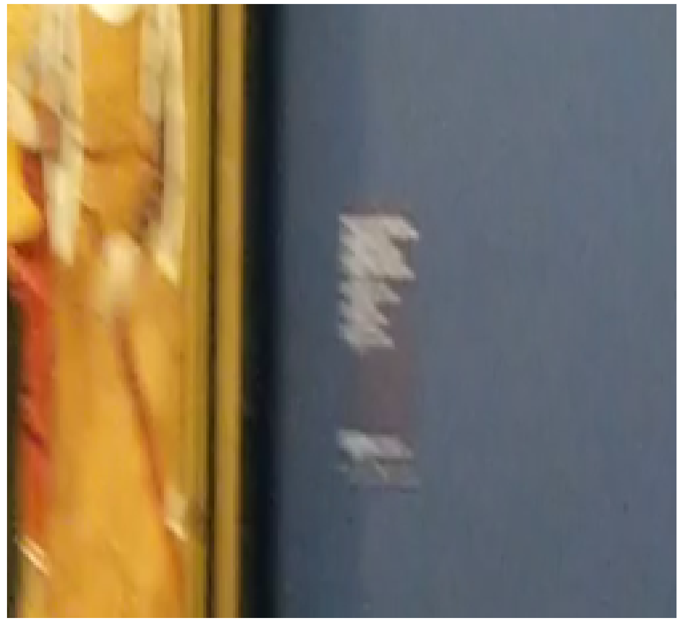

Fig. 5. Information about the paintings hanging next to them have a polygonal shape and may be extracted erroneously

The final major flaw that will be presented lies in the matching phase: matching a painting is done by doing a linear lookup in the database. For a single image this does not pose that much of a problem. The same can not be said when we expanded our algorithm to analyze the frames of a video. This slowdown is mitigated through two changes to the original algorithm. First, we run it only every 30 or so video frames which is equivalent to around 1 second in real life. Suppose the user is using their smart phone's camera, even when running, it will take some time to cross from one room to the other. Finally, the algorithm offloads the matching procedure to a multi-threading environment, freeing the video code from freezing. But what causes this in the first place?

Considering there are 688 pictures in the database, coupled with the keypoints and descriptors that were extracted during the prebuilding phase, are matched with the descriptors of the segmented painting results in a total.

Furthermore, there is also no stop condition to the lookup. This suggests that our matching code has $O(n^2)$ behavior. which with a small amount of pictures is not a factor that we can simply leave. A possible solution to this problem may lie in other journals _ToDo: ref naar boek/conference over bag of words/large databases . _ToDo: laatste paragraaf herschrijven

IV. Conclusion

_ToDo: overview of the most important contributions and the results, without introducing anything new

_ToDo: after the reader has read the paper, the reader can look at the contributions and results from a different viewpoint

_ToDo: statements can be made more explicit
_ToDo: eventueel future work

References

[1] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[2] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov 1986.

[3] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32 – 46, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0734189X85900167