5-2015

Locating Camera Position in 3-D Space from Distinct Features of Architecture on 2-D Image

Yiran Fan
Trinity University, yfan@trinity.edu

Trinity University

Computer Science

Locating Camera Position in 3-D Space from Distinct Features of Architecture on 2-D Image

Author

Irene Fan

Bachelor of Science,

Trinity University

Adviser

Dr. Matthew Hibbs

Computer Science Department,

Trinity University

Presented: April 10th, 2015

Commencement: May 16th, 2015

Locating Camera Position in 3-D Space from Distinct Features of Architecture on 2-D Image

Irene Fan

A departmental senior thesis submitted to the Department of Computer Science at Trinity University in partial fulfillment of the requirements for graduation with departmental honors.

April 15, 2015

Thesis Advisor

Department Chair

Sheryl Tynes, AVPAA

Student Agreement

I grant Trinity University (“Institution”), my academic department (“Department”), and the Texas Digital Library ("TDL") the non-exclusive rights to copy, display, perform, distribute and publish the content I submit to this repository (hereafter called "Work") and to make the Work available in any format in perpetuity as part of a TDL, Institution or Department repository communication or distribution effort.

I understand that once the Work is submitted, a bibliographic citation to the Work can remain visible in perpetuity, even if the Work is updated or removed.

I understand that the Work's copyright owner(s) will continue to own copyright outside these non-exclusive granted rights.

I warrant that:

1) I am the copyright owner of the Work, or
2) I am one of the copyright owners and have permission from the other owners to submit the Work, or
3) My Institution or Department is the copyright owner and I have permission to submit the Work, or
4) Another party is the copyright owner and I have permission to submit the Work.

Based on this, I further warrant to my knowledge:

1) The Work does not infringe any copyright, patent, or trade secrets of any third party,
2) The Work does not contain any libelous matter, nor invade the privacy of any person or third party, and
3) That no right in the Work has been sold, mortgaged, or otherwise disposed of, and is free from all claims.

I agree to hold TDL, Institution, Department, and their agents harmless for any liability arising from any breach of the above warranties or any claim of intellectual property infringement arising from the exercise of these non-exclusive granted rights.”

I choose the following option for sharing my thesis (required):

[X] Open Access (full-text discoverable via search engines)
[ ] Restricted to campus viewing only (allow access only on the Trinity University campus via digitalcommons.trinity.edu)

I choose to append the following Creative Commons license (optional):

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to Dr. Matthew Hibbs, my thesis advisor for providing me an opportunity to conduct research with him and his guidance and encouragement in carrying out this research. I also want to thank my parents for their love and unconditional support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

1. CHAPTER I: INTRODUCTION

This research aimed to develop an algorithm that estimates the camera position in space from which an image was created using computer vision techniques. There are existing studies of camera recovery with 2-D images, one of which requires the initial establishment of a 3-D framework by 3-D imagery reconstruction with images taken by a single camera with known geolocation [1]. Our research suggests a more automated and efficient method using mainly 2-D computer vision techniques. In this paper, we will discuss, in order, the prior studies upon which this algorithm is built, the routes we took to the creation of our final algorithm, the algorithm design, and the results of the algorithm.

1.1 Potential Applications

This algorithm has potential applications in many areas, including the military, self-driving cars, and Art History. One thing that both a military vehicle and a self-driving car are interested in is the ability to identify their surroundings and the location of the cameras in space. This algorithm can be integrated as a means to identify different types of objects. For example, the algorithm can be extended to distinguish a pedestrian from a vehicle or a traffic light. Moreover, this algorithm could be used for the academic field of Art History. Because the algorithm works with any 2-D images, it could work with drawings and paintings created by artists as well as pictures taken by cameras. It would be interesting to apply this algorithm to some artworks and be able to estimate the location from which the artworks were created. Note that the actual location and the calculated location are not the same since artists have their own way of rendering an object and there exist various perspective methods. In cases where the actual location of creation for the art piece is already known,

by comparing it to the calculated location, one can possibly deduce the method of perspective used in creating the art piece.

1.2 Major Architecture Studied: Cathédrale Notre-Dame de Paris

Cathédrale Notre-Dame de Paris, also known as Notre-Dame, is chosen to be the subject for the case study. In this section, we will discuss the reason for the choice, and the notable features of Notre-Dame.

1. **Reasons for Choosing Notre-Dame.**

   Notre-Dame is one of the most internationally well known pieces of architecture with a classic French Gothic design. Gothic architecture is known for its emphasis on the ornate decorative style, which often leads to more noise in image acquisition and the simplification of the features. The rationale is that if we choose a challenging architecture for the case study, we are likely to encounter more difficulties that are possibly representative of other cases. Also, Notre-Dame has numerous features that form various geometric shapes, which also makes it highly representative of other architecture.

2. **Notable Features of Notre-Dame.**

   In the case study, we are interested in the features on the west facade of Notre-Dame. Figure 1.1 is an image of the west facade. The west facade of Notre-Dame has numerous features that are useful for this research. The ones we considered or studied include its concave outline, the two horizontal arrangements of columns and figures, and the rose window in the center. In the end, we decided to utilize the rose window as a unique feature of Notre-Dame for the case study. However, this assumption is not true in reality because there exists other architecture with similar rose windows. A generalized algorithm is discussed in this paper, which aims to be applicable to all objects.

Fig. 1.1. The west facade of Notre Dame.

2. CHAPTER II: PRIOR WORK

In this chapter, we will give a brief introduction of the significant prior work this research is based on, and a more detailed explanation of the prior work implemented in our final algorithm. In the following order, we will discuss image noise reduction methods, image feature detection, and image registration. In short, noise reduction is the process of reducing errors that occur throughout image acquisition and processing. Image feature detection is the recognition and localization of certain features in the image. Image registration refers to the mapping of coordinate systems from one or multiple 2-D images either to other 2-D images of the same object or to the 3-D object itself.

2.1 Image Noise Reduction Methods

In general, image noise refers to the random and unexpected image values introduced throughout all stages of image acquisition and processing. However, image noise could have slightly different meanings or additional information in different cases. Regardless of the context it is under, such inconsistencies can lead to errors in the results of any pixel manipulation and therefore is undesired. Thus, images should be processed with the intention of reducing, if not eliminating, image noise before any further processing or analysis is applied.

2.1.1 Grayscale

Color information rarely helps with identifying image features such as edges, corners, and lines, because changes in pixel values can be well represented within one channel in most cases. There are exceptions, for example, when the goal is to iden-

tify objects of a certain color, or the feature of interest is simply undetectable in its grayscale image. Otherwise, if the color information is insignificant, turning the original colored image into a grayscale image can reduce the image noise introduced in relevant stages of image acquisition.

Moreover, dealing with a grayscale image instead of its colored version is more desirable in terms of processing time, especially for large-scale projects where the factor of speed is no longer negligible.

2.1.2 Blur

Blurring an image is commonly used to reduce noise by reducing detail. In the context of image processing, blurring refers to the process of reducing the edge content so that the transition from one pixel to its adjacent pixels becomes smoother. There are various types of blurring, which can be achieved by applying variations of the basic linear filtering algorithm [2]:

$$I_A(i,j) = I * A = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} A(h,k) I(i-h, j-k) \tag{2.1}$$

Let I be a $N \times M$ image. A is a $m \times m$ convolution matrix of a linear filter. The filtered version I_A of I at each pixel (i,j) is defined by the equation above.

As mentioned previously, there are different types of filters that can be applied as A in the algorithm, the most commonly used ones are mean filter, weighted average filter, and Gaussian filter [3]. In the earlier stage of this research, a mean filter and Gaussian filter were implemented. However, they were not used in the final algorithm.

2.1.3 Threshold

Thresholding is the process of creating a binary image from a grayscale image [4]. There are several different thresholding methods as well, the simplest being fixed-level thresholding, which was implemented in this research. Fixed-level thresholding takes a constant as input, and replaces each pixel in the image with a black pixel of value

0 if the original pixel value is less than the inputted constant, or with a white pixel of value 255 if the original pixel value is greater than the inputted constant. Once the process is applied to all pixels, the result would be an image with pixel values of either 255 or 0, that is a binary image. This step is usually conducted after converting the original image to a grayscale image as it enhances the advantages of a grayscale image in terms of noise reduction and speed.

2.2 Basic Image Feature Detection

In computer vision, the term *image feature* refers to two possible entities [2]:

1. a *global* property of an image, for instance the average grey level; or
2. a *local* property, that is a part of the image with some special properties. For example, an ellipse, a straight line, a corner, etc.

So far we are only interested in the local image features as defined above for two general reasons. First, the global features will change from image to image, and are too environment-dependent. For instance, the average grey level would change under different lighting. Second, it is unlikely to identify objects by the global image features for the reason that many pictures of different architecture would yield similar results. However, global image features could help with some potential future work of this research. For example, if the architecture of interest is mostly composed of only one color, which is the case for Notre-Dame, we could use the average color of the area that has been identified as the architecture to further identify and analyze 2-D images of that architecture. With that said, the definition of image features will be limited to local properties in this thesis.

2.2.1 Edge Detection: Canny Edge Detection

Edges are consistent sharp variations in terms of image value from pixels to adjacent pixels. The detection of edges is generally a three-step process [2]:

1. **Noise Reduction.** Reducing noise using the Linear Filter equation (2.1) with a Gaussian filter.

2. **Edge Enhancement.** Design a filter that locates edge pixels by returning a large value at edge pixels and small values elsewhere.

3. **Edge Localization.** First, suppress non-maxima edge points by analyzing edge strength and orientation at each pixel, which results in thinned and therefore more accurate edges represented by a collection of local maxima. Second, differentiate the local maxima that are real edges from those that are caused by noise by establishing the minimum value to declare a local maxima an edge (thresholding).

The Canny Edge Detection algorithm was fully implemented in the earlier stage of research, but our implementation was not used in the final algorithm.

2.2.2 Corner Detection

The idea behind corner detection is that it searches for locations where a strong edge turns rapidly [5]. The motivation for detecting corners is a rather intuitive one: areas that contain corners are more likely to be distinctive than the ones without. Differentiating one corner from other corners is an easier task than distinguishing a part of the cloudless sky from the rest of the sky. Therefore, if the significant corners of an object in one image can be matched accordingly with those in other images of the same object, we could proceed with registration and then locate the camera position of specific images.

The corner detection algorithm [2] was implemented in the earlier stage of research, but not used in the final algorithm. The idea of registration by matching corners failed mainly because architectural images usually contain too much noise in the background such as clouds, tourists, and trees that are unlikely to be filtered without human input or additional information.

2.2.3 Straight Line Detection: Hough Transform

The Hough Transform can be used to detect straight lines. First, we find the edges on the image using Canny Edge Detection 2.2.1. Second, because any straight line can be defined with the angle θ of its normal and its distance ρ from the origin, the equation of the line is:

$$x \cos\theta + y \sin\theta = \rho, \theta \in [0, \pi] \tag{2.2}$$

Thus, in order to locate image regions containing straight lines, a two-dimensional array is constructed as an accumulator, with one dimension representing ρ and the other representing θ, so that each cell describes a straight line. For each *edge* point (i, j) in the image, cells in the accumulator are incremented by 1 if the corresponding straight line passes through the point (i, j). After all edge points are processed, the accumulator is inspected to find cells with counts that are more than an assigned threshold. The lines whose corresponding cells pass the threshold are the output [6]. Note that ρ and θ values are binned so that we are dealing with a finite amount of lines. The number of cells in the accumulator and the accuracy of the result have a positive correlation. On a side note, Hough Transform can also be utilized for ellipse detection [7].

However, the idea of locating the top edge and side edges of the architecture did not succeed for similar reasons as we discussed in Corner Detection 2.2.2 - too much background noise led to many edges caused by noise to pass the threshold creating muddled results that were not optimal. Note that the error in this case could possibly be reduced by improving the accuracy of Canny Edge Detection, because the input of the Hough Transform is the output of Canny Edge Detection. If more time were allowed, we could attempt to determine a more suitable Gaussian filter as well.

2.3 More Image Processing with OpenCV

We have discussed some basic image feature detection such as Edge Detection, Corner Detection, and Straight Line Detection. Here we discuss two more complex feature detections provided by the OpenCV library [8].

2.3.1 Find Contours: findContours()

The OpenCV method, findContours(), implements an algorithm that converts a binary picture into its border representation, and then extracts the topological structure from it [9].

The result of the OpenCV contour detection is a collection of groups of points, where each point group represents an *object* contour. Different from edges, a contour indicates that all points within the contour belongs to a single feature or shape. That grouping of points is helpful for identifying distinctive features in this research, and the OpenCV method findContours() is used for that purpose [8]. A more detailed explanation of this method is included in section 3.3.1.

2.3.2 Fit Ellipse: fitEllipse()

The OpenCV function fitEllipse() takes in a collection of points as input and finds the ellipse that fits the collection of points the best in a least-square sense [8]. Note that the method requires a minimum number of 5 points for the calculation. Theoretically, 3 points are sufficient to describe an ellipse. However, if the input collection contains fewer than 5 points, the method will consider the calculation as unreliable and refuse to return an ellipse. However, note that the algorithm returns an ellipse regardless of data quality or other factors as long as the data fulfills the minimum number of points required [10].

2.4 Image Registration: Determining the Location of Objects in Space

Image registration can be defined as a mapping between images with respect to space and intensity [11]. There are many methods for image registration using collections of points or lines, which we previously defined as the basic image features [12]. In this research, however, we focus on the combination of features and their pattern of relationships. More specifically, in the case of Notre-Dame, we are looking for the circle-shaped rose window that is composed of two sets of radially contiguous ellipses, as explained in section 1.2. Note that the two sets of radially contiguous ellipses could be simplified as two circles that describe their pattern, which simplifies the problem to image registration using ellipse(s). We mainly studied two approaches from the past that dealt with registration using ellipses and the mathematics behind these methods.

Both methods share the same fundamental concept in that they search for a circle in 3-D space that is projected to a 2-D image plane and determine the mapping of the projection. In detail, the orientation of the plane that the circle in space is on can be found by rotating the camera so that the intersection of the cone with the 2-D image plane becomes a circle [2]. The cone refers to the imaginary cone with the camera location being its vertex and the circle in space being its base. Also, the output of both algorithms is the unit normal of the circle plane. The difference between the two methods will be explained in detail in the two subsections below.

2.4.1 Pose From Ellipses: Registration with Perspective Projection

In this method, the rotation of the camera mentioned above is estimated as two consecutive rotations. The first rotation puts the Z axis through the center of the circle in space, and aligns the X and Y axes with the corresponding axes in the 2-D image plane. The second rotation puts the plane of the circle in space parallel to the 2-D image plane. Figure 2.1 demonstrates the basic idea behind the mapping from 3-D space to a 2-D image plane. While this method would generalize to nearly any image

taken of a 3-D circle, we found that a closed-form solution based on an assumption of orthogonal projection was sufficient for this research, as discussed further below.

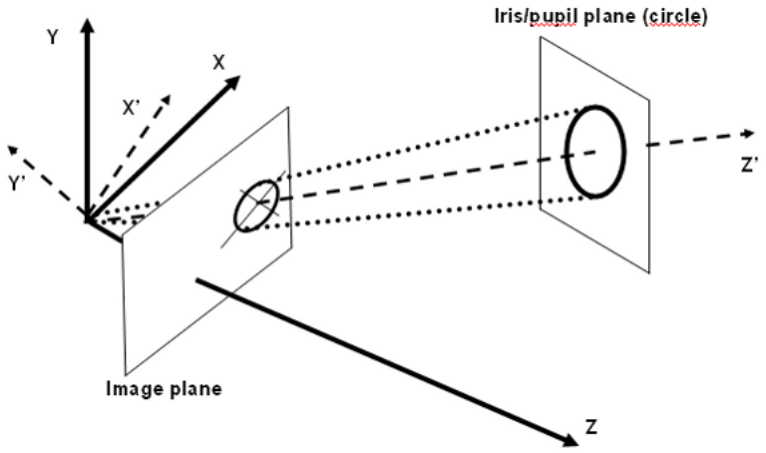

Fig. 2.1. The cone and two planes used by algorithm POSE_FROM_ELLIPSE. The Iris/pupil plane in the figure refers to the plane on which the circle in space rests [2].

2.4.2 Vehicle Pose Identification: Registration with Orthogonal Projection

The second method we studied is different in the sense that it assumes parallel projection, more specifically orthogonal projection. In an orthogonal projection, all the projection lines are orthogonal to the projection plane, leading to the fact that it disregards the factor of focal length and does not preserve the angles or distance between lines or points on different planes [13]. An algorithm that assumes orthogonal projection is not ideal for image registration because cameras work in a more realistic manner with their attempt to mimic a human eye. However, it works for this research because, even though the features are 3-D objects, they are generally "flat", meaning that the depth variance of individual features within the 3-D circle is negligible. Also, because the size of the rose window is relatively small comparing to the entire west facade and because most pictures are taken at angles near the window's normal, the

angle of the rose window does not create much difference in depth. Therefore, in the case of Notre-Dame, we can disregard the depth of its rose window and simplify the registration math.

The algorithm is composed of the following steps [14]:

1. **Describe the Ellipse with Its Covariance Matrix.** The ellipse refers to the one in the 2-D image plane. The mapping of the standard implicit equation of an ellipse, the general matrix and the covariance matrix is shown below [15]. The implicit equation of an ellipse:

$$AX^2 + BXY + CY^2 + DX + EY + F = 0$$

The general matrix of the same ellipse:

$$\begin{bmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{bmatrix}$$

The covariance matrix of an ellipse is

$$\begin{bmatrix} A & B/2 \\ B/2 & C \end{bmatrix}$$

2. **Calculate the Eigenvalues of the Matrix.** The two eigenvalues of the covariance matrix are $\lambda_1, \lambda_2 (\lambda_1 > \lambda_2)$

3. **Calculate the Major and Minor Axes Lengths of the Ellipse.** The major axis $(a1) = 2\sqrt{\lambda_1}$ and the minor axis $(a2) = 2\sqrt{\lambda_2}$.

4. **Calculate the Unit Normal of the Circle.** The unit normal of the circle relative to the coordinate system of the image plane ϕ is the following, with M_{00} corresponding to A in the general matrix, and M_{11} corresponding to C.

$$\phi \equiv \begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \end{bmatrix} = \frac{1}{a_1} \begin{bmatrix} \pm\sqrt{a_1^2 - 4M_{00}} \\ \pm\sqrt{a_1^2 - 4M_{11}} \\ \pm a_2 \end{bmatrix} \tag{2.3}$$

The calculated unit normal of the circle is the normal of the plane that the circle is on (iris plane). Again, it is calculated by rotating the plane of the 2-D ellipse until the 2-D ellipse aligns with the circle in 3-D space.

3. CHAPTER III: ALGORITHM DESIGN

In the previous chapter, we have listed the significant studies that either failed to perform but contributed to the ideation of the working solution, or was implemented in the final algorithm. Now, we will discuss the algorithm design from two different approaches: a more general outline that is applicable to all architecture and possibly objects, and a very specific procedure customized for Notre-Dame. Please note that for the following sections, the term "a distinct combination of features" is sometimes shortened as "distinct features".

3.1 A Generalized Approach

Below is the basic outline for the generalized approach:

1. **Define and describe a distinct combination of features of the architecture.** "A distinct combination of features" refers to the composition of certain features of an architecture that is so unique that with them the architecture can be distinguished from any other objects. Such a composition is most likely constructed with basic features; it is often the relative arrangement of those basic features that forms the distinctiveness. Note that the the pattern or arrangement described by the combination of features should be detectable even under variation introduced by different viewing angles, lighting conditions, etc.

2. **Filter out images that do not contain the distinct features of the architecture of interest.** If the combination of features is unique, then it is easy to perform a check on images to see whether the features are present. If they are present on an image, we can say that image contains the architecture of interest.

3. **Locate the combination of features on each image that contains the architecture.** Locate each feature in the coordinate system of the image plane, and define each feature as a geometric shape that best describes itself.

4. **Perform registration on each feature.** There are various methods through which image registration can be done. The geometric representation of the feature decides the appropriate method(s) for the registration of that feature. The result of registration of each feature is a mathematical representation of the relationship between the object in the 2-D image plane and in the 3-D space, for example, the unit normal of the image plane in the coordinate system of the 3-D plane. Because 2-D images do not necessarily capture things in an ideal state due to perspective projection, extraneous objects, etc., there will be errors in the process of registration resulting in numerous mathematical representations. A best fit approach should be used to resolve that problem.

   Moreover, note that because of the nature and similarity of geometry, there are cases where multiple methods seem reasonable. For example, a circle-shaped object can be treated as an ellipse or a set of circularly aligned points. Therefore theoretically, both ellipse registration and point set registration methods between the image plane and the 3-D space could both be viable. In that case, further assessment should be conducted regarding the accuracy of the geometric representation - if the image is obscured in a way that the circle cannot be perfectly described with an ellipse, perhaps a point set registration will cause less error.

5. **Locate camera position.** Once the one mathematical representation of the spatial relationship is agreed upon, we can dissect that information in order to obtain more information that describes the camera location, such as the angle and distance from which the image was created.

3.2 A Case Study: the Cathedral of Notre Dame

As discussed above, the generalized algorithm is designed to work with any type of architecture. Such generalization can be achieved with a comprehensive set of tools so that the algorithm is able to configure the most appropriate sequence of methods tailored to different combinations of features. Due to many restrictions, we decided to implement a more feasible solution for a specific architecture: Notre-Dame, with the assumption that the rose window on its west facade is adequate to identify Notre-Dame among all architecture. The steps below are similar to those discussed above but modified in more specific terms, also, the steps below are arranged and grouped differently for clarity. Now, before we dive into the details, here is a brief summary of the algorithm:

1. Define and describe a distinct combination of features of the object.
2. Recognize images that contain Notre-Dame by identifying these distinct features.
   (a) Basic processing: Grayscale, Blur, and Threshold
   (b) Find object contours
   (c) Fit an ellipse to each contour
   (d) Find radially contiguous ellipses
   (e) Fit an ellipse to the center points of contiguous ellipses of each group
   (f) Filter by additional features to improve accuracy (optional)
   (g) Estimate location of object outline to filter out background noises (optional)
3. Locate Camera Position.
   (a) Perform registration of distinct features.
   (b) Calculate camera position.

3.3 A Case Study: Implementation

3.3.1 Define and describe a distinct combination of features of the object

In this implementation, we assume the distinct combination of features to be the rose window on the west facade of Notre-Dame. Its attributes can be described in various ways; we will discuss two of them below.

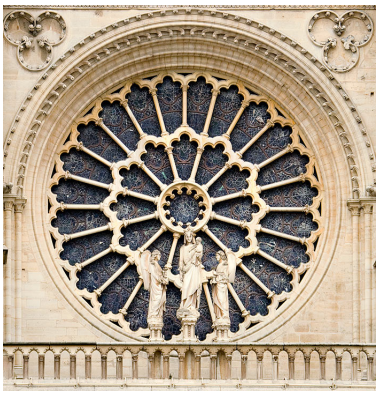

Fig. 3.1. Detail of rose window on the west facade of Notre-Dame.

The outline of the rose window is similar to a circle. Therefore one way is to describe the rose window by its outline as a circle. However, it is a basic feature, therefore, as discussed in section 2.2, is very unlikely to be distinguishable among all other potential ellipses detected by the ellipse finder. Thus, the simple outline of the rose window is inadequate to be unique.

Then we realized that the individual glass panels that compose the rose window construct an interesting pattern. Upon closer examination, the glass panels can be grouped into two sets by the difference in size. If we treat each glass panel as an ellipse, the center points of the panels from each set form a double-nested circle, resulting in a more complex feature that is more likely to be unique. After testing, we concluded that this feature is often distinguishable from noise and other features.

Note that the lower part of the circular pattern is obscured by three figures. However, the identification of the pattern does not require the full circular pattern to be detected, but only some partial indication of the pattern. This is one of the notable advantages of this algorithm, which will be further explained in the following section.

3.3.2 Recognize images that contain Notre-Dame by identifying these distinct features.

Search for the double-nested circular pattern that describes the rose window in each image from the input. Because the rose window is assumed to be a unique component of Notre-Dame, its presence ensures the identity of Notre-Dame. Note that often Notre-Dame-like architecture also contains similar windows which can lead to false positives. The assumption that the rose window of Notre-Dame is a unique composite feature is made to simplify the algorithm.

The following procedure is implemented for this step. A null return value on any of the non-optional steps indicates the absence of the rose window. Therefore images that fail to pass this filter are not considered as Notre-Dame images.

1. **Basic processing: Grayscale, Blur, and Threshold.**

   The purpose of this step is to reduce image noise. The basic idea and importance of noise reduction is discussed in section 2.1. We have written several noise reduction tools including Gaussian blur with linear filter. However, we decided to use the standard OpenCV functions for speed and less maintenance. Three output images of these procedures are shown in figure 3.2

2. **Find object contours.**

   This step implements the OpenCV function, findContours(). This function requires the image to be binary, which is achieved by thresholding the original image in grayscale. The output of the function is a collection of groups of points,

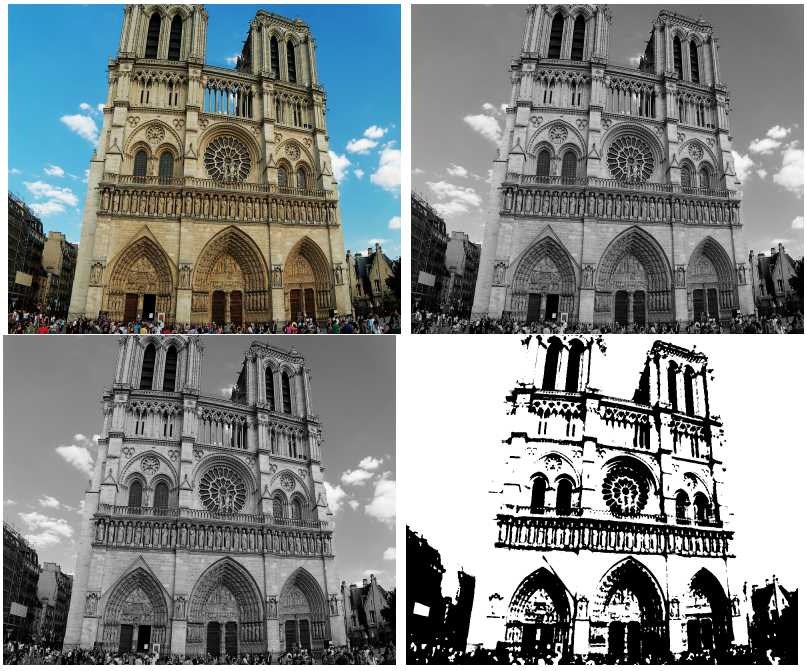

Fig. 3.2. Top row from left to right: original colored image, image after grayscale. Bottom row from left to right: image after blur, image after threshold.

with each point group representing an object contour. An image demonstrating a sample output of this step is shown in figure 3.3.

As you can see, the contours are not simply edges. They imply additional information such as the grouping of edges that indicates a feature. For a more detailed explanation of the algorithm it utilizes, please refer to section 2.3.1.

3. **Fit an ellipse to each contour.**

   The OpenCV function, fitEllipse(), is implemented. At the beginning of this section, we discussed treating the individual glass panels of the rose window as ellipses. This function essentially takes in a set of points and performs a best fit algorithm on the points in the sense of least square. An image demonstrating a sample result is shown in figure 3.4. For a more detailed explanation of the algorithm it utilizes, please refer to section 2.3.2.

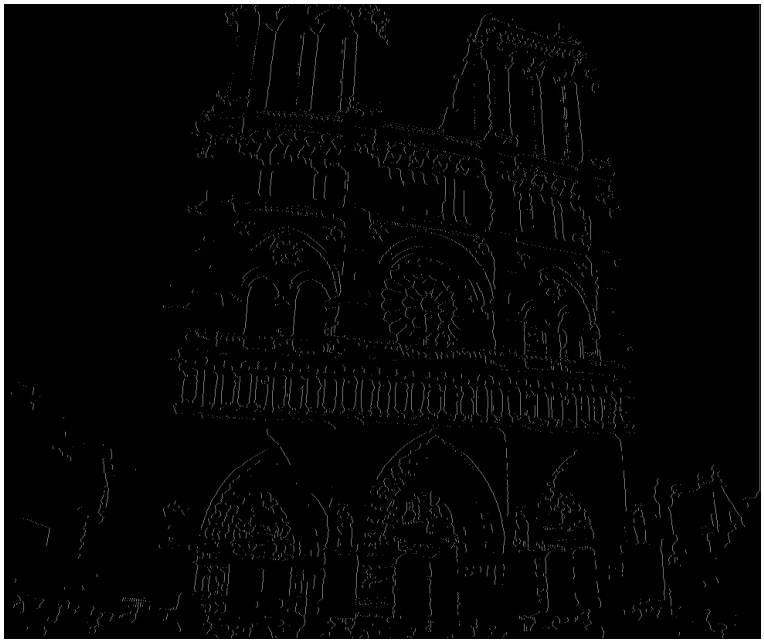

Fig. 3.3. Image after findContours().

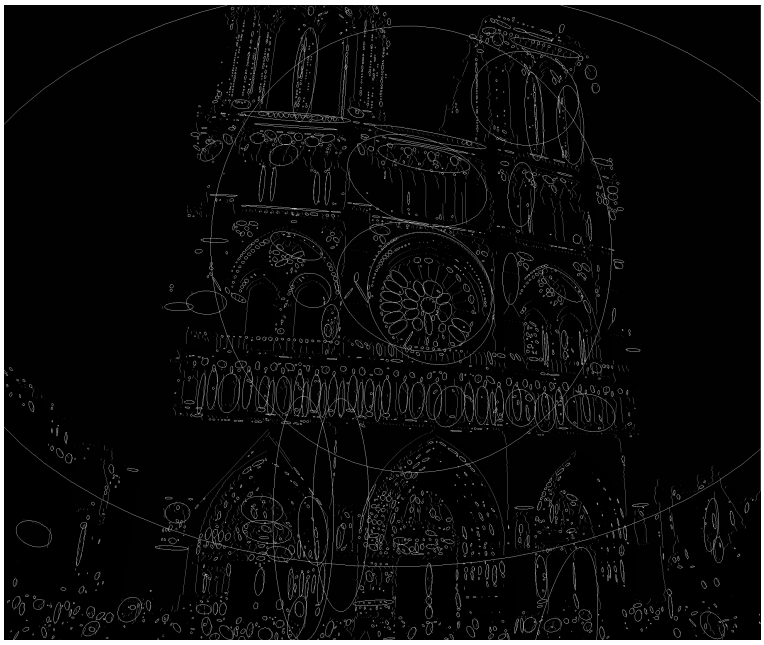

Fig. 3.4. Image after findEllipse() is applied on the result of findContours().

4. **Find radially contiguous ellipses.**

As we discussed at the beginning of this section, the glass panels are radially contiguous. In the terms that we used previously, the two sets of glass panels are aligned circularly such that the center points of ellipses fit to those panels form a double-nested circle. Here is a detailed explanation of the method we developed in order to search for groups of radially contiguous ellipses:

```
select an unvisited ellipse, e_0.
  \\the ellipse has a center point p_0,
  \\and a minor axis of length a_2.
calculate the coordinates of the point p_1
  \\so that the distance between p_0 and p_1 is a_2,
  \\which extends from p_0 in one of the two semi-minor axis direction;
if(there exists an unvisited ellipse e_1 containing p_1) THEN
  store e_0
  mark e_0 as visited
  recurse the previous procedure on e_1
ELSE
  if(collection.size() > 5) THEN
    store collection
  ELSE
    mark the ellipses in the collection as unvisited
```

At the end of this algorithm, each collection of ellipses stored represents a group of radially contiguous ellipses that has a minimum size of 5.

5. **Fit an ellipse to the center points of contiguous ellipses of each group.**

   As we obtain the groups of radially contiguous ellipses, we can fit an ellipse to the center points of the ellipses in each group. In the end, the pattern of each group of ellipses is described by an ellipse. Figure 3.5 is an image demonstrating a sample output of this step. Note that we expect the ellipse obtained by this

step to be contained in the image plane, which removes ellipse groups that are linearly contiguous.

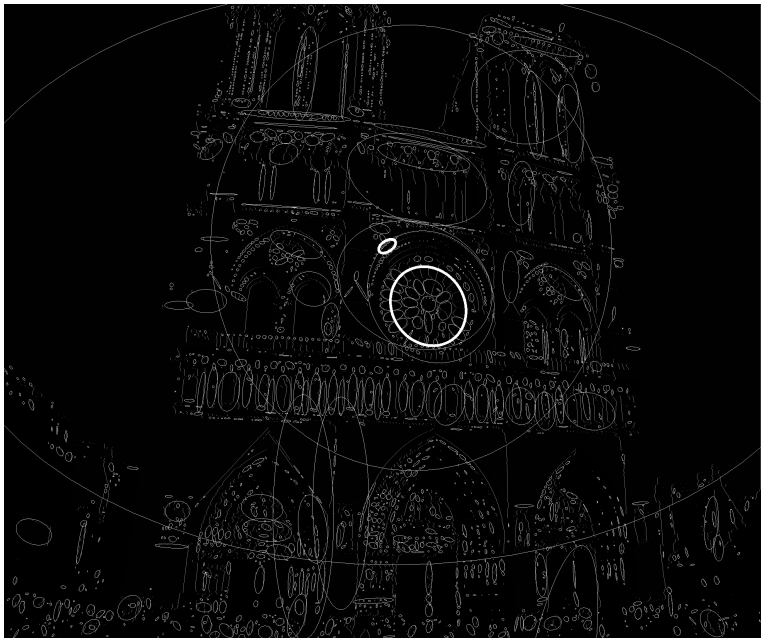

Fig. 3.5. Image after fitting ellipses to groups of radially contiguous ellipses.

6. **Filter by additional features to potentially improve accuracy (optional).**

   Some potential features are the horizontal panel of figures right below the rose window and the outline of the west facade. Adding those features to the existing combination of features leads to a stricter filter. However, it does not imply higher accuracy. The more features the composite integrates, the more error will potentially occur in the process, and therefore the harder it is for an image to pass the filter. It might cause some target images to be filtered out because some features are not captured clearly enough in the image to be identified. Thus, a more sophisticated algorithm that involves a more comprehensive yet forgiving filtering system should be developed to enhance the accuracy.

7. **Estimate location of object outline to filter out background noises (optional).**

   The outline of Notre-Dame can be estimated by the relationship between the composite features and solution itself. As long as we have the knowledge of the 3-D object and its representation on the 2-D image plane, we can express the size difference in terms of ratios to known features, and use these ratios to estimate the outline of Notre-Dame.

3.3.3 Locating Camera Position

This step is similar to its generalized counterpart in step 4 and 5 of section 3.1. Same idea and basic procedure discussed previously apply to this step.

1. **Perform registration of rose window.**

   Because the rose window is an (approximately) perfect circle in 3-D space that is projected onto an image plane as a 2-D ellipse, registration of an ellipse is implemented in this algorithm. The result of registration is in the form of a unit normal of the 3-D plane of the original circle. In this case, we implemented the orthogonal projection approach. The reasoning behind the choice of this method and the method itself are included in section 2.4.2. In this step, we obtain the unit normal of the plane on which the rose window is, in terms of the coordinate system of the 3-D iris plane.

2. **Calculate camera position.**

   With the unit normal of the 3-D circle, the angle between the normal and the projection direction of the 2-D image plane can be calculated. In addition, the distance from the camera to the rose window (the west facade of Notre Dame) can be estimated with the knowledge of the angle of the camera, and the size of rose window both in reality and in the image. Unfortunately, we were not able to integrate the distance calculation due to time constraints.

This concludes the detailed explanation of algorithm. With the input of any image, we were able to recognize whether it is an image of the west facade of Notre-Dame by identifying the rose window. The position of the rose window can be used to further filter image noises and to calculate the camera position, from which the image was taken.

4. CHAPTER IV: RESULTS

4.1 Results from Early Stages of Research

4.1.1 Edge Detection Tools

We started the research by studying edge detections. For a general introduction of edge detections, please refer to section 2.2.1. We implemented Canny Edge Detection [2] with two different kernels: the linear mean filter and the linear weighted average filter. Also, the process of grayscale and blurring takes place prior to the calculation of image gradients, which is the first step of the major procedure. One sample result is demonstrated in figure 4.1. We also tried several edge detection tools such as the

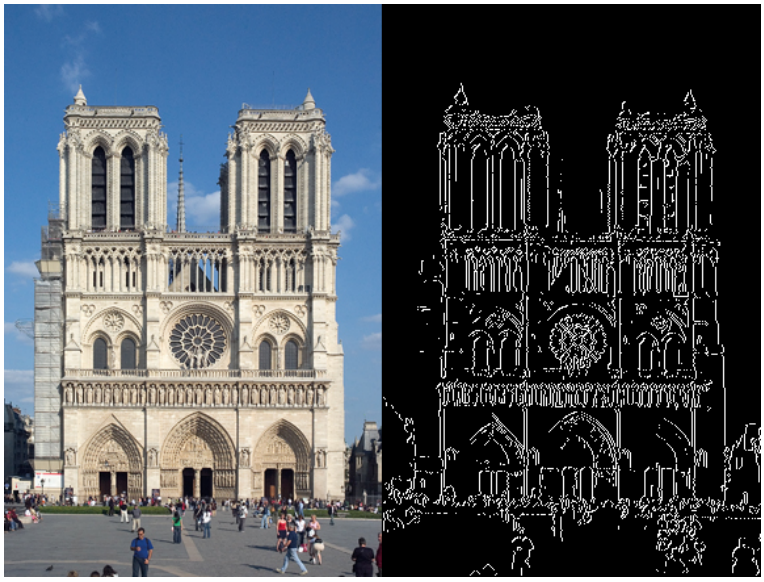

Fig. 4.1. A comparison of the original image and the result of our implementation of the Canny Edge Detection.

Scharr operator and the Sobel operator, provided by the OpenCV library. Some sample results are shown in figure 4.2

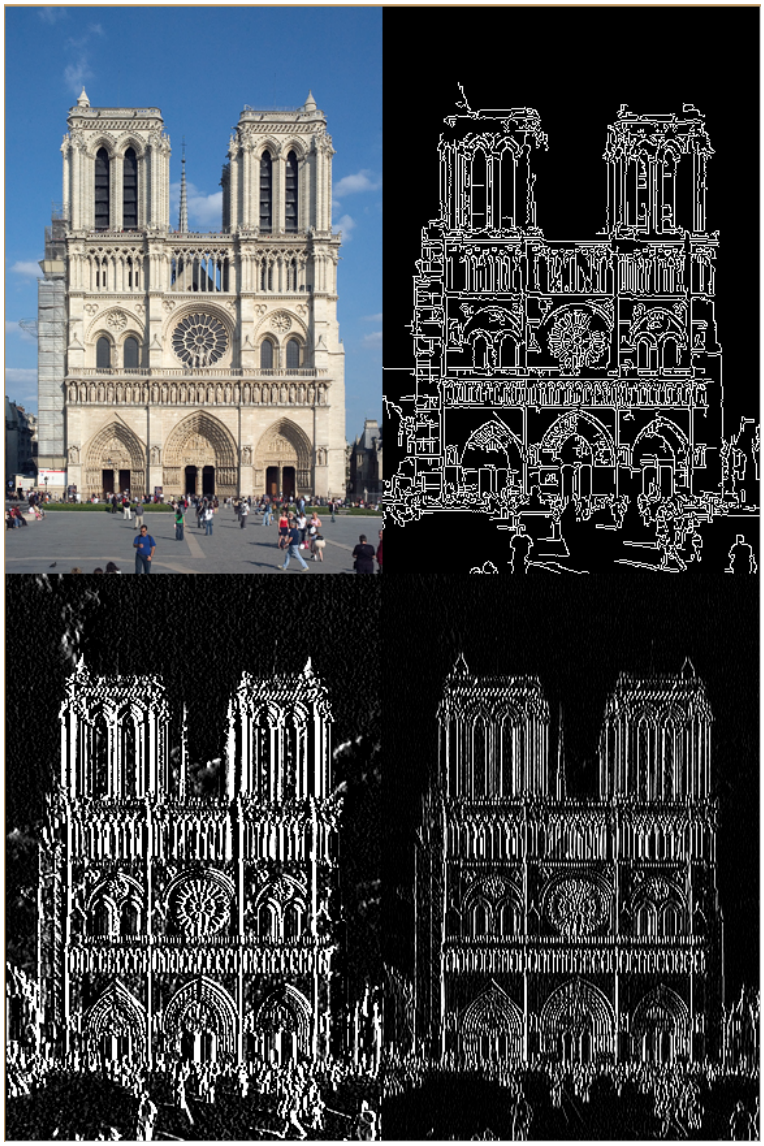

Fig. 4.2. Result of three different edge detection. Top row from left to right: the original colored image, image after the Canny Edge Detection from OpenCV. Bottom row from left to right: image with Scharr operator, image with Sobel operator.

If we compare our implementation of the Canny Edge Detection to the one by OpenCV, the library implementation is less strict so more edges pass the filter, resulting in more continuous edges. The level of strictness can be adjusted with different linear filters.

4.1.2 A More Sophisticated Method

The other basic image processing algorithms we have written are corner detection and straight line detection with Hough Transform. However, the basic tools and even their combination were insufficient to identify the outline of Notre-Dame. The reasoning behind the inadequacy of the basic image processing tools is discussed in detail in section 2.2. Therefore, a more sophisticated method needed to be develop to filter out the unwanted background noise. The distinctiveness of composite features is discussed in section 3.3.1. We eventually found a solution in more complex and therefore more distinctive features, specifically ellipses and the their alignment. In order to detect those distinctive composite features, we developed the algorithm using the two OpenCV functions: findContours() and fitEllipse().

The figures exemplifying each step of the final algorithm can be found in section 3.3.

4.2 Final Result

To demonstrate the final result, several images were processed with the algorithm that we developed, and the camera positions from which the images were created are plotted on a map, for those images on which the rose window was detected. Note that in this implementation, we assume a focal length of 1. However, to calculate the distance between the camera and Notre-Dame, we would need to take focal length into consideration. Figure 4.3 is an image of the calculated camera angles plotted on a map.

In the next section, we will select one case for a detailed analysis, and then some other cases will be briefly showcased in another section.

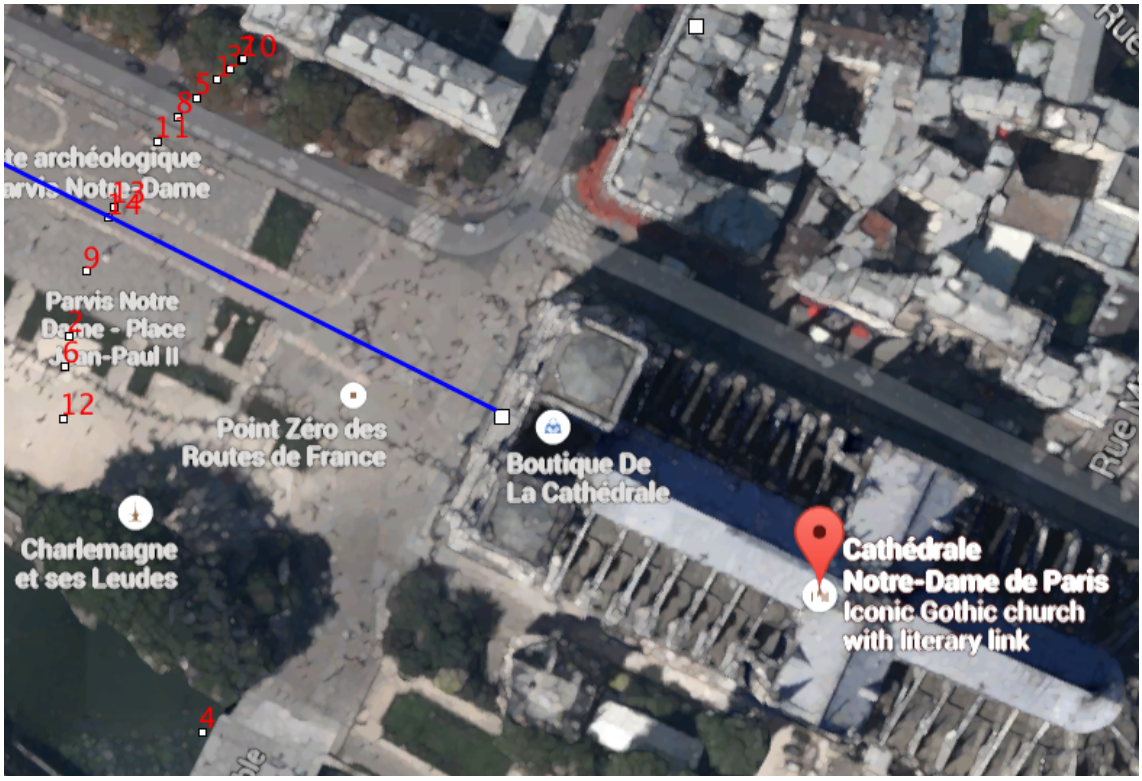

Fig. 4.3. Calculated camera angles plotted on a bird's-eye view snapshot of Notre-Dame on google map. Each point corresponds to an image of Notre-Dame. The blue line is an estimate of the line starting from the center of the rose window, which is perpendicular to the west facade of Notre-Dame.

4.2.1 Detailed Analysis of One Case

For consistency, we will analyze the same image used in the step-by-step explanation of our algorithm. The original image is shown in figure 4.4.

The camera angle corresponds to the point labeled with the number 8 on figure 4.3. Intuitively, one can judge that the image was taken from the left side, facing the west facade, with an angle of no more than $\frac{\pi}{2}$ from the perpendicular line. In fact, point number 8 fits the estimation above. Unfortunately, the level of accuracy of the algorithm cannot be evaluated without a precise geotag of the image. However, the accuracy of the ellipse registration depends on the accuracy of the feature detections.

Fig. 4.4. The original colored image.

The errors during the process of registration are negligible because it is a closed-form mathematical procedure, which means the error or inaccuracy is inevitable as long as assumptions are made that are unlikely in reality. At the same time, such error is constant for every single procedure performed and therefore is insignificant.

4.2.2 Other Cases

In this section, several other cases will be demonstrated briefly. Specifically, 4 other images from different angles will be selected.

Case 1.

Figure 4.5 is another image that was taken from the left side, facing the west facade of Notre Dame. Its result is plotted on the map as point number 11. As one can estimate intuitively, the calculated angle shown on the map is reasonable.

Case 2.

Figure 4.6 is an image that was taken from the right side, facing the west facade of Notre Dame. Its result is plotted on the map as point number 6, which is intuitively correct as well.

Case 3.

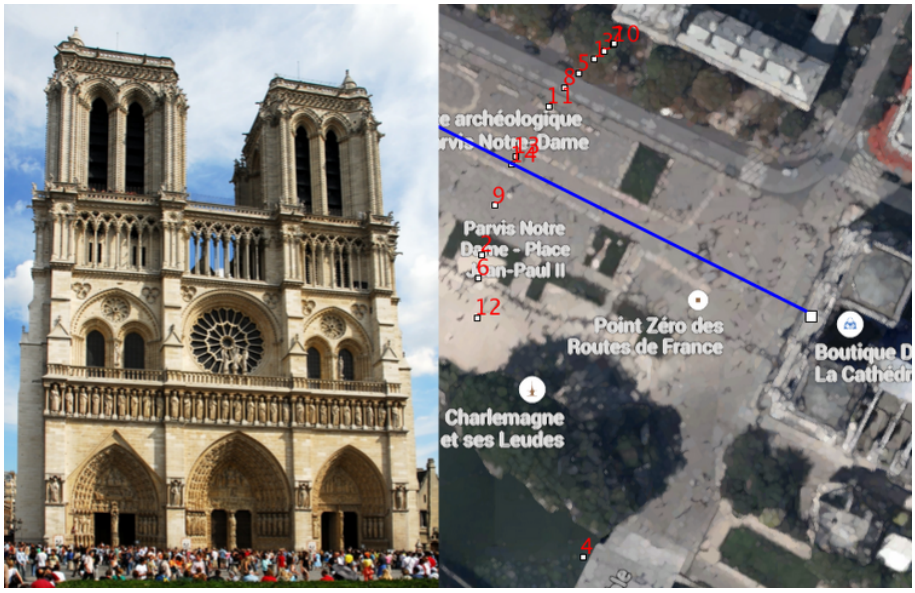

Fig. 4.5. Left: the original colored image. Right: the plotted results. Point number 11 represents the calculated camera position disregarding the distance factor.

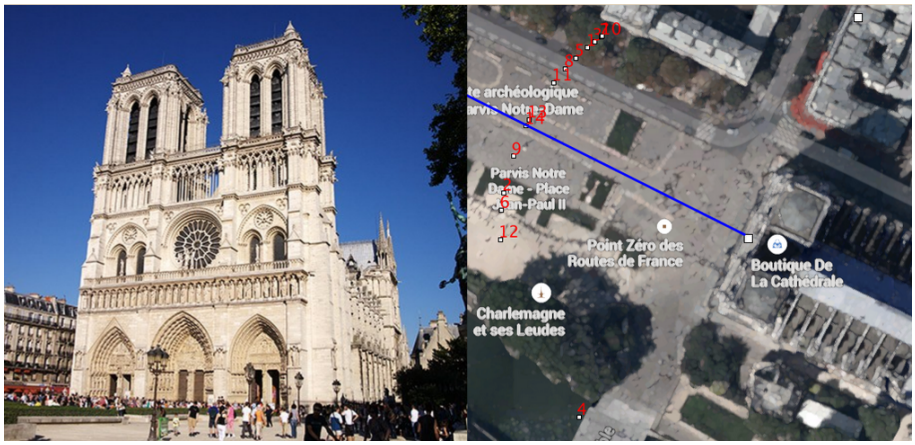

Fig. 4.6. Left: the original colored image. Right: the plotted results. Point number 6 represents the calculated camera position disregarding the distance factor.

Figure 4.7 demonstrates an incorrect result. The original image shows that it was taken from the right side with a very slight angle from the perpendicular line of the west facade. However, the result shows that the inputted image was taken from the

right side, but the angle between the camera and the perpendicular line is apparently much larger than the original image suggests.

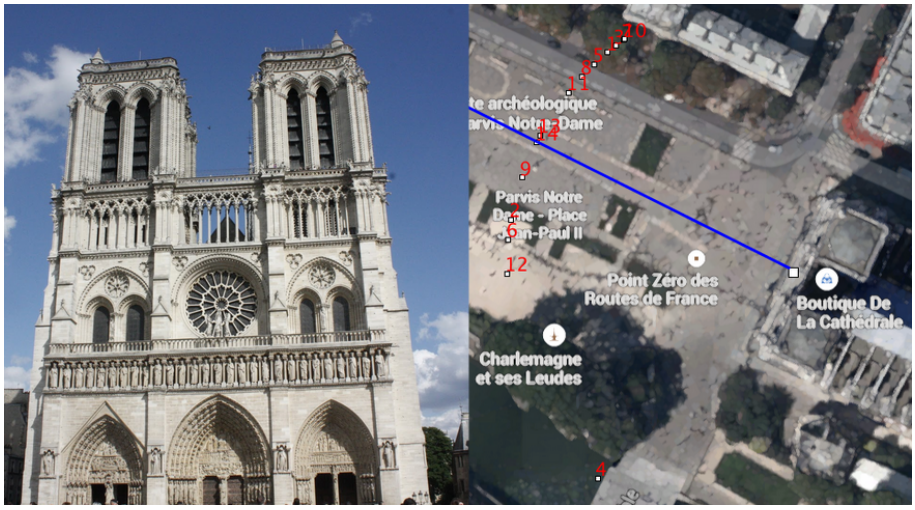

Fig. 4.7. Left: the original colored image. Right: the plotted results. Point number 4 represents the calculated camera position disregarding the distance factor.

To find out what went wrong in this case, we will take a look at the ellipse detection. The result of the ellipse detection is shown in figure 4.8. As one can tell, the ellipse drawn on the figure does not match with the circular pattern created by the ellipse-shaped glass panels. Therefore we can conclude that the error of the final result is caused by the inaccuracy of the ellipse detection in this case, which might be caused by the shadow casted on the glass panels that obscures their shapes.

Case 4.

Figure 4.9 shows that it was created from the very center of the west facade, with a trivial angle between the camera location and the perpendicular line. The result matches with its indication that the point representing it, point number 14, seems to be on the perpendicular line.

One might notice that this image is not a picture taken by a camera, but a drawing. The fact that our algorithm can process this image shows that one can perform this

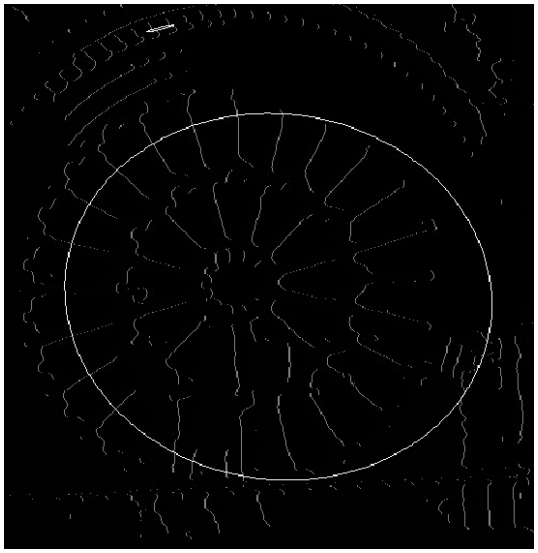

Fig. 4.8. The result of ellipse detection for case 3.

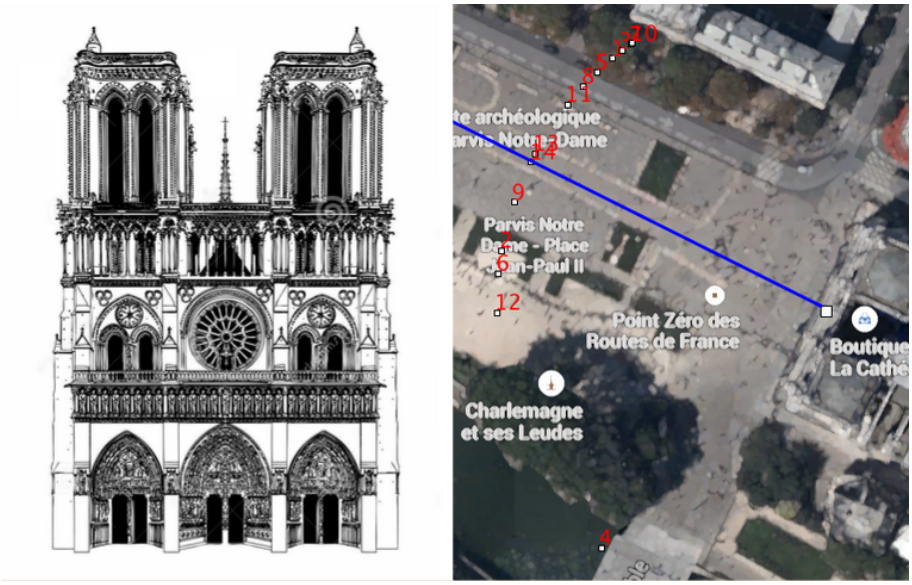

Fig. 4.9. Left: the original colored image. Right: the plotted results. Point number 14 represents the calculated camera position disregarding the distance factor.

algorithm to 2-D images of any form including art forms, which leads to potential applications in the field of Art and Art History.

5. CONCLUSION

In previous chapters, we have discussed the prior work on which this research is based, the algorithm design, and some results of algorithm. However, this research does not end here - there are many potential future works to be done, and here are some examples:

1. **Integrate the distance factor in locating the camera position.**

   So far, the calculation of the camera position only takes into account the angle; it assumes the camera focal length to be 1, resulting in the distance between the cameras and the object to be the same for all images, which is not the case in reality. Thus, the next step for this research is to integrate the distance factor in the calculation.

   The distance between the camera and the object when the image was taken can be determined by knowing the focal length of the camera, the size of the object in 2-D image plane, and its size in 3-D space (reality).

2. **Include other features of Notre-Dame from all facades.**

   As mentioned in section 1.2, Notre-Dame has numerous features on all facades. Even though the rose window on the west facade is not a unique feature of Notre-Dame, it is very likely that the combination of the rose window and another feature, either on the same facade or on a different facade, would be unique. It goes back to our idea that a composite feature is more likely to be distinctive than basic features. Thus, including more features from all facades would not only make the image filter more accurate and realistic, but also deal with images that include other facades so that our algorithm is not exclusively applicable to the west facade of Notre-Dame.

3. **Generalize the algorithm to be applicable to other architecture.**

   The implemented algorithm is a specific case study of the west facade of Notre-Dame. However, the potential applications mentioned in 1.1 require an algorithm that is able to identify a broad range of objects. We have discussed a possible design for a generalized algorithm in section 3.1. One of the greatest challenges is defining the distinct features of different objects, which could involve machine learning.

4. **Integrate Machine Learning to automate the process of defining and determining distinct combinations of features.**

   The process of defining and determining distinct combinations of features of an object must be conducted on images of that object. However, part of our implemented algorithm is to identify the object in the image. This introduces a contradiction, so we need to have a set of images of an object before we can study its features and formulate them into composite features. That set of images can be collected using a combination of file name, tag names, and geotag information. Then, a machine learning algorithm needs to be implemented so that the object in the known image set can be studied. This is one of the ways to implement an automated generalized algorithm.

LIST OF REFERENCES

LIST OF REFERENCES

[1] P. Cho and N. Snavely, "3d exploitation of 2d ground-level amp; aerial imagery," in *Applied Imagery Pattern Recognition Workshop (AIPR), 2011 IEEE*, pp. 1–8, Oct 2011.

[2] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[3] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego, CA, USA: California Technical Publishing, 1997.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[5] L. Kitchen and A. Rosenfeld, "Gray-level corner detection," tech. rep., DTIC Document, 1980.

[6] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, pp. 11–15, Jan. 1972.

[7] A. Chia, M. Leung, H.-L. Eng, and S. Rahardja, "Ellipse detection with hough transform in one dimensional parametric space," in *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, vol. 5, pp. V – 333–V – 336, Sept 2007.

[8] G. Bradski, "opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[9] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following.," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[10] M. Fitzgibbon, A. W.and Pilu and R. B. Fisher, "Direct least-squares fitting of ellipses," *Pattern Analysis and Machine Intelligence*, vol. 21, pp. 476–480, May 1999.

[11] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, pp. 325–376, Dec. 1992.

[12] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3-d point sets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, pp. 698–700, Sept 1987.

[13] P. Maynard, *Drawing Distinctions: The Varieties of Graphic Expression*. Cornell University Press, 2005.

[14] M. Hutter and N. Brewer, "Matching 2-d ellipses to 3-d circles with application to vehicle pose estimation," *CoRR*, vol. abs/0912.3589, 2009.

[15] C. Young, *Precalculus, Student Solutions Manual*. Wiley, 2010.