FROM IMAGES TO MAPS

by

Ron Appel

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Copyright © 2008 by Ron Appel
Abstract

From Images to Maps

Ron Appel
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
2008

This work proposes a two-stage method that reconstructs the map of a scene from tagged photographs of that scene. In the first stage, several methods are proposed that transform tag data from the photographs into an intermediary distance matrix. These methods are compared against each other. In the second stage, an approach based on the physical mass-spring system is proposed that transforms the distance matrix into a map. This approach is compared against and outperforms MDS-MAP(P) when given human tagged input photographs. Experiments are carried out on two test datasets, one with 67 tags, and the other with 19. An evaluation method is described and the optimal overall reconstruction generates maps with accuracies of 47% and 66% respectively for the two test datasets, both scoring roughly 40% higher than a random reconstruction. The map reconstruction method is applied to three sample datasets and the resulting maps are qualitatively evaluated.
ACKNOLEDGEMENTS

I would like to thank Parham Aarabi for his extraordinary guidance and support; this work was completed because of all of the time and effort that he put in to helping me.

I would also like to thank my family (Eva, Julian, Lora, and Austin the blij) for everything, and my colleagues (Nevena and Alex) for their friendship and help.
CONTENTS

1 INTRODUCTION .............................................................................................................. 1
  1.1 PRIOR WORK .............................................................................................................. 2
  1.2 THESIS OUTLINE ..................................................................................................... 6

2 OVERVIEW OF THE APPROACH .................................................................................. 7
  2.1 THE MAP RECONSTRUCTION PROBLEM ................................................................. 7
  2.2 ROUGH-TAGGING ..................................................................................................... 10
  2.3 THE WIRELESS SENSOR POSITIONING PROBLEM ................................................. 13

3 FROM IMAGES TO MATRIX ......................................................................................... 18
  3.1 DISTANCE MATRIX GENERATION ............................................................................ 18
  3.2 DISTANCE METRICS ............................................................................................... 22
     Metric 1: Binary Existence ......................................................................................... 23
     Metric 2: Euclidean Distance .................................................................................... 24
     Metric 3: Density Adjusted Euclidean Distance ......................................................... 26
     Metric 4: Variance Adjusted Euclidean Distance ....................................................... 31
     Metric 5: XY-Variance Adjusted Euclidean Distance ................................................ 34
  3.3 GLOBAL MERGING METHODS ................................................................................. 39
     Merging Method A: Generalized Mean ...................................................................... 41
     Merging Method B: Inverse Sum ............................................................................... 42

4 FROM MATRIX TO MAP ................................................................................................ 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>MAP RECONSTRUCTION</td>
<td>49</td>
</tr>
<tr>
<td>4.2</td>
<td>DESCRIPTION OF MDS-MAP(P)</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>DESCRIPTION OF THE SPRING MODEL</td>
<td>58</td>
</tr>
<tr>
<td>5</td>
<td>EXPERIMENTS AND RESULTS</td>
<td>63</td>
</tr>
<tr>
<td>5.1</td>
<td>EVALUATION METHOD</td>
<td>63</td>
</tr>
<tr>
<td>5.2</td>
<td>CHAPTER 3 EXPERIMENTS</td>
<td>67</td>
</tr>
<tr>
<td>5.3</td>
<td>CHAPTER 3 RESULTS</td>
<td>69</td>
</tr>
<tr>
<td>5.4</td>
<td>CHAPTER 4 EXPERIMENTS AND RESULTS</td>
<td>77</td>
</tr>
<tr>
<td>6</td>
<td>SAMPLE APPLICATIONS</td>
<td>83</td>
</tr>
<tr>
<td>6.1</td>
<td>OFFICE MAP RECONSTRUCTION</td>
<td>83</td>
</tr>
<tr>
<td>6.2</td>
<td>SHOPPING CENTRE MAP RECONSTRUCTION</td>
<td>85</td>
</tr>
<tr>
<td>6.3</td>
<td>WEDDING MAP RECONSTRUCTION</td>
<td>89</td>
</tr>
<tr>
<td>7</td>
<td>CONCLUSIONS</td>
<td>92</td>
</tr>
<tr>
<td>7.1</td>
<td>SUMMARY OF CONTRIBUTIONS</td>
<td>94</td>
</tr>
<tr>
<td>7.2</td>
<td>FUTURE WORK</td>
<td>94</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

Over the past decades, the introduction of the affordable digital camera has brought on a vast change in our society’s culture. Today, practically anyone can take a photograph of any thing and any place on Earth and save it to a computer or share it with the rest of the world. What’s more, online storage sites even allow for tags to be added alongside photographs, thereby enabling the association of words or names with specific areas in the photographs. Overall, billions of tagged images exist on the Internet, and as a result, a great number of locations on Earth are well documented with sets of photographs.

But a photograph can be used as much more than just a pretty picture of a place; it can help us learn about the relationships of the objects in the scene that it captures. An interesting application is the reconstruction of a scene solely from a set of photographs taken of it. Not surprisingly, in the field of computer vision, researchers are trying to do just this: to learn about a scene – the positions, sizes, shapes, colours, and textures of the objects within the scene – given only photographs of that scene. This is also the focus of
our work: to learn about a scene – in particular, the map of the tagged objects in the scene – given only tagged photographs of that scene.

1.1 PRIOR WORK

Prior work in the area of scene reconstruction follows two general directions, one in which the camera positions of the photographs are known, and the other in which the camera positions or not known.

When the positions of the cameras are known, several methods have been proposed to reconstruct the scene. *Voxel Colouring* [1] treats the scene as a 3-D array of volumetric pixels (voxels). Each voxel is either assigned a colour or left as blank, making use of the pixel colours from the various viewpoints. *Space Carving* [2] also uses the voxel model, intersecting 3-D silhouette regions across multiple photographs, thereby carving out the 3-D hull of objects in the scene. *Graph Cuts* [3] is used to reproduces a scene using iterations of stereoscopy calculations followed by optimizations for overall scene smoothness. In general, these methods produce impressive results; however, they are impractical for most scene datasets, as camera position information is rarely available.
When the positions of the cameras are not known, scene reconstruction is performed using multiple view geometry by matching common points among images [4]. A sufficient number of common points must accurately be identified in each image. In the past, this was accomplished either by inserting easily distinguishable markers into the scene before taking the photographs or by manually identifying specific areas in the photographs.

Over the last decade, research in this area has led to the automatic identification of these common points. Interest point detectors are used to select distinctive regions within images that are invariant to various transformations, such as scaling and rotation. These points are then matched across the set of images to find correspondences in the visible regions of the scene.

One of the most popular algorithms for the matching of points is SIFT [5]. It selects regions of interest using a difference-of-Gaussians detector and describes each region as a histogram of image gradient orientations weighted by their magnitudes and a Gaussian window. The key points are matched using an approximate nearest-neighbour algorithm, followed by a verification process using the Hough transform. The matching accuracy of SIFT is highly dependant on the detected points, which can be unreliable with blurry, cluttered, or low-resolution images.
In the last year, Microsoft Research released Photosynth, a state-of-the-art program that reconstructs the 3-D positions of key points in a scene when given multiple partially overlapping photographs of that scene [6]. The points are detected and matched across images using the SIFT algorithm, and their 3-D positions are calculated using multiple-view geometry. Figure 1 consists of thumbnails of example input photographs and the output point cloud as generated using Photosynth.

![Figure 1: 2-D input thumbnails and 3-D output point cloud using Photosynth](image)

With this large set of highly overlapping photographs, we qualitatively see that Photosynth performs well. However, as mentioned above, SIFT is unreliable on blurry, noisy, or low-resolution input photographs, causing Photosynth to fail. Furthermore, if there are too few input photographs, there is simply not enough information to be able to exactly reconstruct a 3-D scene.
The constraints within which Photosynth performs well are not compliant with the majority of readily available sets of images on the Internet. But what if there was a way to use only the already-available human placed tags associated with a set of images to reconstruct the layout of the objects in a scene? Humans do not suffer from the same shortcomings as computers when faced with clutter, blurriness, and low resolution in a photograph. However, tags added by humans are often not accurately or consistently positioned, and are few in number.

If we were able to find a way to make use of these few, inaccurate tags, then we would potentially be able to use the set of available tags corresponding to any location on Earth and reconstruct it without the need for image-processing methods. Eventually, as more tagged photographs are added to the online repository of images, we might even be able to connect all of the scenes together and form a reconstruction of the entire photographically documented world.

The availability of such a reconstruction would have many practical purposes. The remote exploration of a scene, the guiding of a robot through a scene with obstacles, crime scene investigation, virtual tourism, and archaeological tools are just a few of these exciting applications. In this thesis, we develop an approach to reconstruct the map of a scene from a set of tagged photographs – from images to maps. Although these end results may seem overzealous, they provide great motivation for our work.
To the best of our knowledge, there does not yet exist a method that performs map reconstruction from a set of images tagged by humans. However, methods do exist that perform map reconstruction from a set of wireless sensor distances [7,8]. Notably, SPA [9] reconstructs a map from distances using triangulation and a matching of coordinate systems. However, results rapidly degrade with range errors. MDS-MAP [10] relies on multidimensional scaling to generate an initial position estimate, followed by an optional refinement step. Although it is able to compensate for range errors, it does not perform well on irregular map topologies. More recently, MDS-MAP(P) [11] is an extension of MDS-MAP in which local maps are generated for each node after which they are merged together to form a global map. MDS-MAP(P) is able to reconstruct irregular topologies as well as compensate for range errors. It is described in greater detail in Chapter 4.

1.2 THESIS OUTLINE

In the following chapters, we describe our proposed method to reconstruct a map from a set of tagged images. Chapter 2 gives an overview of the problem and the proposed algorithm. Chapters 3 and 4 go into detail about the two stages of our approach: going from images to distance matrices, and going from distance matrices to maps. Chapter 5 provides experimental procedures and results. Chapter 6 consists of map reconstructions from several sample datasets. Finally, Chapter 7 concludes this thesis, and discusses directions for future work.
2 OVERVIEW OF THE APPROACH

In this chapter, we describe the map reconstruction problem and give an overview of our approach to solve this problem. We also describe a similar sensor positioning problem that is currently being researched, and with it, give justification for our approach.

2.1 THE MAP RECONSTRUCTION PROBLEM

We illustrate our problem with the following example. Imagine that we have just returned from New York, where we had taken several photographs of the Lincoln Center. While uploading these images online, we had quickly added a few tags to each of them, as shown in Figure 2. Each tag represents a location on the photograph and an object identifier. For simplicity, we labeled the tags with positive integers, although they could just as well have been labeled with their names – like “The Met” or “NY State Theatre”.
Once tagged, we are no longer concerned with any of the visual information in these photographs. The only information that we access is the tag data. Accordingly, the New York photographs now appear to us as the data structures shown in Figure 3.
Online photo-sharing websites such as *flickr* and *facebook* presently contain millions of such sets of photographs and tags [16,17]. These sites store their tag data in similar structures as ours (shown in Figure 3). We can now define our problem: using only the tag data extracted from a set of input photographs, we wish to reconstruct a 2-D map of the scene from which the photographs were taken; to find the relative positions of each of the tagged objects. Figure 4 shows a possible solution map as well as an aerial view of the real New York scene with which we can compare our results.

Figure 4: A solution map compared with the actual aerial map
2.2 ROUGH-TAGGING

The sets of photographs found on online photo-sharing sites are mostly tagged by humans, inherently leading to large amounts of noise in the tag locations. We denote these input photographs to be *roughly-tagged*, by which we refer to human inconsistencies across photographs in the tagging of objects. Many causes can lead to inconsistent human tagging, a few of which we describe below.

A human may be rushed, uncaring, or simply inaccurate, leading to different tag locations on the same object in different photographs, as demonstrated in Figure 5. The tag (shown as a star) in the photograph on the left is placed between two columns whereas it is placed on a single column in the photograph on the right.

![Figure 5: Rough-tagging caused by human inaccuracy](image)
Alternatively, if more than one human tags the set of photographs, the location of the tags on a particular object may be completely inconsistent due to a lack of tagging protocol. Figure 6 consists of two photographs tagged by different people. The building in the photograph on the left is tagged around its columns whereas the same building in the photograph on the right is tagged on its dome.

Figure 6: Rough-tagging caused by multiple taggers

Further, even when an attempt is made to keep the location of the tag consistent, a partial occlusion of an object may necessitate placing the tag in a different location, as shown in Figure 7. In the photograph on the left, the tag is placed on the dome, whereas in the photograph on the right, the dome is not visible, and so the tag is placed elsewhere.
Finally, since we are trying to reproduce a two-dimensional map, we assume that all of the objects in the scene are tagged on a single plane (as projected onto the image). In actuality, most of the objects in the scene do not lie on a single plane, but lie slightly above, below, or even completely off of it. A photograph of a grass field and buildings is shown in Figure 8. All of the buildings are built on the same ground plane. Due to varying heights, however, they are not all tagged on this ground plane.
Had we not accepted roughly-tagged photographs as input and had we ensured that each input photograph contained at least seven tags (not to mention several other conditions), we would have been able to employ existing multi-view geometry and optimization techniques to reconstruct the original scene. However, since we do accept roughly-tagged photographs as input, most of which contain no more than four tags, an alternate approach must be developed.

2.3 THE WIRELESS SENSOR POSITIONING PROBLEM

In the field of wireless sensor positioning, there exists a problem with many similarities to map reconstruction. This problem is defined as follows: given a slew of wireless sensors, each with the ability to measure the distance between itself and neighboring (within-range) sensors, it is desired to generate a map of the relative positions of all of the sensors. An example of such a setup is given in Figure 9, where the nodes represent wireless sensors. Only sensors within range of each other are connected by an edge.
Figure 9: Wireless sensors spread over a plot of land

The input data to this type of problem may be given as a single distance matrix. The elements of this matrix correspond to the distances between indexed sensors. Figure 10 is the input distance matrix corresponding to the above example. Distances between out-of-range sensors are undefined; consequently, the corresponding matrix elements are blank.

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>19</td>
<td>20</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>0</td>
<td>24</td>
<td>33</td>
<td>12</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>0</td>
<td>19</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>19</td>
<td>0</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>12</td>
<td>28</td>
<td>30</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>26</td>
<td></td>
<td></td>
<td>0</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>19</td>
<td>0</td>
<td>30</td>
<td>38</td>
<td>17</td>
<td>12</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>30</td>
<td>0</td>
<td>18</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>38</td>
<td>18</td>
<td>0</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>17</td>
<td>22</td>
<td>24</td>
<td>0</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td></td>
<td>25</td>
<td>0</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>23</td>
<td></td>
<td>13</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: Input distance matrix corresponding to the layout in Figure 9
We use an existing solution to the sensor positioning problem as the basis for our approach in solving map reconstruction. In order to easily compare our solution with the one used in solving sensor positioning, however, we split our approach into two stages. In the first stage, the tag data from all of the photographs is amalgamated into a single distance matrix, and in the second stage, a map is reconstructed based on the distance matrix generated in the first stage. We note that our second stage achieves the same goal as does the wireless sensor positioning problem.

To get to this second stage, however, we must initially complete the first stage. The first stage involves the generation of a single distance matrix, which is a lossy process. Three main loses result from the generation of this single matrix. Firstly, with a single distance matrix, the number of tags in each image is lost. Secondly, the position and orientation of the tags relative to the bottom-right corner of the images is lost, as shown in Figure 11.

![Figure 11: Border offsets and orientation losses](image-url)
Thirdly (and most importantly), several reconstruction ambiguities may arise. Figure 12 shows a situation in which tag 1 could be positioned either above or below the other two tags. This is known as a flip ambiguity.

![Figure 12: Example of a flip reconstruction ambiguity](image)

Figure 13 shows a situation in which distances are undefined for tags 1 – 3 and tags 2 – 4, (shown as blank matrix elements), leading to the possibility of tags 3 and 4 being either slightly above tags 1 and 2 or slightly below them. This is known as a flex ambiguity.

![Figure 13: Example of a flex reconstruction ambiguity](image)

Hence, an approach that does not first generate a single matrix may obtain better results as it could make use information that we lose with our approach. However, we accept
these potential losses because of the simplicity associated with the two modular stages as well as the ease of comparison with an existing approach. The procedural flowchart for the work done in the following chapters is given in Figure 14.

As shown, the outputs of both of the stages undergo an evaluation. The method with which we evaluate our results is described in a later chapter. In the following two chapters, we describe in detail the specifics of the two stages of our approach: the distance matrix generation, and the map reconstruction.
3 FROM IMAGES TO MATRIX

The goal of our work in this chapter is to take the available tag data from the individual input photographs, and with it, to generate a single distance matrix. Several distance metrics are proposed to extract the tag data from the photographs and two merging methods are examined with which the tag data is combined into a single matrix.

3.1 DISTANCE MATRIX GENERATION

As input, we are given a set of photographs, all taken of a single scene containing several distinguishable objects of interest, possibly at different vantage points and zoom levels. None of the input photographs contain less than two tagged objects, and very few of them contain more than four tagged objects. Figure 15 shows an example of a scene with four objects of interests: a circle, a triangle, a diamond, and a cross.
Figure 15: A Scene with several objects of interest

Figure 16 consists of four tagged photographs taken of the scene shown in Figure 15, from different vantage points and with various zoom levels.

Figure 16: Sample photographs taken of the scene in Figure 15
Each tag consists of two pieces of information: an identifier and a coordinate. For simplicity, the identifier is a positive integer used to reference a particular object. Tags describing the same object on different photographs have the same identifier. As shown in Figure 16, each time the circle is tagged, its identifier is 1, the triangle’s is 2, the diamond’s is 3, and the cross’ is 4. The coordinate is a two-dimensional point on the photograph that corresponds to the tagged object’s pixel location. Although most objects occupy many pixels in a photograph, only a single pixel’s coordinate is recorded.

We note that the location of the tags of the same object in different photographs is not consistent. For example, in the bottom-left photograph in Figure 16, the diamond is tagged closer to its upper vertex, whereas in the bottom-right photograph, it is tagged at its center. Furthermore, we note that an object may appear in a photograph but not be tagged, such as the triangle in the bottom-left photograph. These are some of the results of rough-tagging, as was previously discussed.

Using the extracted tag data from the individual photographs, we estimate the distance between each pair of objects. As is suggested by the title of this section, we store these estimated distances in the organizational structure known as a distance matrix. Given a set of \( n \) points: \( p_1, p_2, \ldots, p_n \), and a distance metric \( m \) that defines the distance between any two points: \( p_i \) and \( p_j \), the distance matrix, \( D = [d_{ij}] \), is generated such that each element is assigned a value equal to the pair-wise distance between the two respective points indexed by that element; \( d_{ij} = m(p_i, p_j) \).
From each input photograph, a distance matrix can be generated using its tag data. The two bottom photographs from Figure 16 above as well as their corresponding distance matrices are shown in Figure 17. The Euclidean norm was used as the distance metric. Blank matrix elements correspond to undefined distances in the photographs.

![Figure 17: Two tagged photographs and their individual distance matrices](image)

We note that the two distance matrices in Figure 17 report different distances between tags 1 and 3. However, it is necessary to obtain only a single value for the distance between any two objects. Hence, the individual distance matrices from each of the input photographs must be merged together in some way as to result in a single matrix, which we call the *global distance matrix*, at which point the goal of this chapter is fulfilled.
Consequently, this chapter consists of two main sections. In the first section, a metric is used to generate the individual distance matrices from the tag data of the individual photographs. In the second section, a merging method combines the information from all of the individual distance matrices into a single global distance matrix. The procedural flowchart for this chapter is shown in Figure 18.

![Procedural flowchart for this Chapter](image)

3.2 DISTANCE METRICS

In this first section, the information from each individual input photograph $V$ is used to generate an individual distance matrix, $D_v = [d_{ij}]$. We propose five metrics, starting with the basic binary existence metric and progressively refining it to resolve issues that may arise. Although many other metrics exist, the proposed metrics are simple and
follow a logical progression in their development. These metrics and the issues that they attempt to resolve are given in Table 1.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Intuitive Rationale for Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Binary Existence</td>
<td>Propose a basic initial metric</td>
</tr>
<tr>
<td>2 Euclidean Distance (Eucl. Dist.)</td>
<td>Account for actual pixel distances</td>
</tr>
<tr>
<td>3 Density Adjusted Eucl. Dist.</td>
<td>Account for varying zooms across images</td>
</tr>
<tr>
<td>4 Variance Adjusted Eucl. Dist.</td>
<td>Propose a more robust zoom estimator</td>
</tr>
<tr>
<td>5 XY-Variance Adjusted Eucl. Dist.</td>
<td>Account for different zooms in x and y directions</td>
</tr>
</tbody>
</table>

Table 1: The metrics that we examine and their intuitive rationales

**Metric 1: Binary Existence**

For any given photograph \( V \) in the dataset, the most elementary information that can be extracted from its tags is represented as an *existence* function – denoted \( e_v \). The input to this function is a tag identifier \( i \), and the output, \( e_v(i) \), is 1 if the tag exists in the photograph, and 0 otherwise. Further, we extend it into a two-input existence function, \( e_v(i,j) = e_v(i) e_v(j) \), which allows us to determine the existence of pairs of objects within a photograph. Given this existence function, we can easily define the distance matrix:

\[
d_{v,y} = \begin{cases} 
0, & e_v(i,j) = 1, \quad i = j \\
1, & e_v(i,j) = 1, \quad i \neq j \\
\text{undef}, & e_v(i,j) = 0 
\end{cases}
\]
Figure 19 contains a sample tagged image (of a different scene than that shown in Figure 16 on a previous page) and its corresponding distance matrix, generated using Metric 1.

![Image of a tagged image and its corresponding distance matrix using Metric 1]

**Figure 19: A tagged image and its corresponding distance matrix using Metric 1**

### Metric 2: Euclidean Distance

As shown in Figure 19, the reported distances between any two of the three tags were all equal to 1 using the previous metric. However, from the image itself, it is apparent that the three objects are not equally distant in the scene, but that the triangle and circle are closer together, and the diamond further apart from the other two.

To overcome this problem, a second metric is proposed – a slight variation on the first. Instead of assigning only 0s and 1s, the pixel distance between two tags is recorded. In
this way, we avoid the problematic situation described above by reporting a smaller
distance between the circle and triangle and a larger distance between the diamond and
the other two objects. This metric is the standard Euclidean norm, and the distance matrix
is defined with the following equation:

\[
d_{vij} = \begin{cases} 
\sqrt{(x_{vi} - x_{vj})^2 + (y_{vi} - y_{vj})^2}, & e_v(i,j) = 1 \\
\text{undef}, & e_v(i,j) = 0
\end{cases}
\]

Figure 20 shows reconstructions of the scene from the image in Figure 19 based on the
binary existence metric (on the left) and the Euclidean distance metric (on the right). The
Euclidean distance metric evidently generates a more accurate map of the original scene.

Figure 20: Reconstructions based on binary existence and Euclidean distance metrics
Metric 3: Density Adjusted Euclidean Distance

An obvious issue that the previous metric does not resolve is shown in Figure 21, which contains two images of the same scene. The image on the left contains three tagged objects of interest. The image on the right contains the same three objects, but is further zoomed-out and contains several other tagged objects as well.

Figure 21: Two images of the same scene with different zoom levels

Metric 2 determines the distance between the triangle and diamond in the image on the left to be greater than the corresponding distance in the image on the right, although the true distance is the same. It does not account for the intrinsic zoom levels of the images.

The zoom level of an image corresponds to its linear scaling. The distance between two objects in an image zoomed by 2x will always be twice the distance between the same two objects in an image zoomed by only 1x. An example of this is shown in Figure 22.
Figure 22: The distance between two objects in images zoomed by 1x and by 2x

Letting the zoom level of a given image \( V \) be \( z_v \), the distance between two objects in an image be \( d_v \), and the real, normalized distance between the same objects on the scene be \( d_r \), then we can formulate the following:

\[
d_v = z_v \cdot d_r \quad \Rightarrow \quad d_r = \frac{d_v}{z_v}
\]

When we take a photograph of a scene, we essentially perform a perspective transformation of that scene onto a plane, forming the actual image. Figure 23 consists of two images of a gridded scene. In this scene, the gridlines are perpendicular to each other and form squares of the equal size. The image on the left is a perspective view of this scene, whereas the scene on the right is an orthographical view.
In the perspective image, parallel lines seem to meet. Furthermore, the closer we look to the horizon, the smaller the distance between parallel lines. In the orthographical image, parallel lines remain parallel and there is no notion of a horizon. If we approximate photographs as being orthographical projections (instead of the perspective projections that they really are), then several simplifications result.

One of the resulting simplifications is that the zoom level of an image is inversely proportional to any linear dimension of the projected boundary of that image onto the scene. This is illustrated in Figure 24, which contains the two images taken of the same scene as shown in Figure 22 with the two different zoom levels. The boundaries of the images are projected onto the scene as if they were orthographic projections.
As shown in Figure 24, if we treat images as orthographic projections, then their zoom levels are inversely proportional to the widths of their projected boundaries. Letting the width of the projected boundary of image $V$ be $w_v$, then:

$$w_v \propto \frac{1}{z_v}$$

Furthermore, the area of a shape is a quadratic function of any linear dimension of that same shape. For example, the area of a circle is a quadratic function of its diameter. Letting the area of the projected boundary of image $V$ be $A_v$, then:

$$A_v \propto w_v^2 \implies \frac{1}{z_v} \propto w_v \propto \sqrt{A_v} \implies d_v \propto d_v \cdot \sqrt{A_v}$$
With Metric 3, we attempt to make use of the normalization factor $\sqrt{A_v}$ by asserting the following assumption: the density of objects of interest in a scene is uniform. Given uniform object density, the number of objects – $n_v$ – found within any given boundary on the scene is directly proportional to the area of that boundary. Hence, we conclude that:

$$n_v \propto A_v \Rightarrow d_v \propto d_v \cdot \sqrt{n_v}$$

Assuming that most objects of interest are tagged in a photograph $V$, then the number of tags that appear in that photograph can be trivially computed as follows:

$$n_v = \sum_i e_v(i)$$

As we have shown, the distance between two objects that appear in a photograph $V$ is proportional to the square root of $n_v$. However, we add a parameter $q$ as the exponent of $n_v$, so that a range of values can be tested. If we find that the optimal $q$ is indeed $\frac{1}{2}$, it gives us an indication – though not a guarantee – that the assumption of uniform object density may be correct. With this metric, the distance matrix is defined as follows:

$$d_{v_{ij}} = \begin{cases} (n_v)^q \sqrt{(x_v^i - x_v^j)^2 + (y_v^i - y_v^j)^2}, & e_v(i,j) = 1 \\ \text{undef}, & e_v(i,j) = 0 \end{cases}$$
Metric 4: Variance Adjusted Euclidean Distance

The assumption that tags are uniformly distributed throughout a scene does not hold when there are a small number of objects in the scene, and particularly, in photographs taken at the boundary of the scene. Since there are always a finite number of objects of interest in any scene, the density of objects is zero beyond this boundary, and greater than zero within it.

Figure 25 consists of a scene with four objects of interest in which the true distances between the triangle and circle and between the circle and diamond are equal.

![Figure 25: A scene with equal distances between the shown objects](image)

Figure 26 consists of two images, both taken of the scene shown in Figure 25. Although the image on the right is further zoomed-in than the one on the left, the previous metric assigns the same normalization factor to both images, as they both contain the same
number of tags. This results in the computed distance between the triangle and circle being greater than that between the circle and diamond, which is incorrect.

To solve this problem, we propose that the particular image distances be normalized using the variance of the tag positions instead of using the number of tags. This positional variance of the tags – \( \sigma_v^2 \) – for a given image \( V \) is calculated as follows:

\[
\overline{x}_v = \frac{1}{n_v} \sum_{i} x_{vi}
\]

\[
\overline{y}_v = \frac{1}{n_v} \sum_{i} y_{vi}
\]

\[
\sigma_v^2 = \frac{1}{n_v} \sum_{i} [(x_{vi} - \overline{x}_v)^2 + (y_{vi} - \overline{y}_v)^2]
\]

Modifying our assumption so that the density of objects is uniform only within the boundary of a scene, we note that tags in images that are zoomed-in are further apart
leading to a larger positional variance, whereas tags in images that are zoomed-out are closer together leading to a smaller positional variance. Figure 27 consists of two images of the same view; one zoomed by 1x and the other by 2x, and their respective variances.

![Image of two zoom levels with positional variance numbers]

**Figure 27:** Positional variances of images of the same view with different zoom levels

As in the example in Figure 27, if an image \( V \) is zoomed-in by \( z_v \) relative to an image \( W \) (that has a tag variance \( \sigma_w^2 = 1 \) for simplicity), then we can conclude:

\[
\sigma_v^2 = \frac{1}{n_v} \sum_i \left[ (z_v x_{wi} - z_v \bar{x_w})^2 + (z_v y_{wi} - z_v \bar{y_w})^2 \right] = \frac{z_v^2}{n_w} \sum_i \left[ (x_{wi} - \bar{x_w})^2 + (y_{wi} - \bar{y_w})^2 \right] \\
= z_v^2 \sigma_w^2 = z_v^2 \Rightarrow z_v \propto \sqrt{\sigma_v^2}
\]

And therefore, it follows that:

\[
d_r = \frac{d_v}{z_v}, \quad z_v \propto \sqrt{\sigma_v^2} \Rightarrow d_r \propto \frac{d_v}{\sqrt{\sigma_v^2}}
\]
Although this relationship is accurate when dealing with images of the same view with different zoom levels, it is not necessarily accurate for images of different views. However, we can still gain the intuition that our distance equation should contain some exponent of the variance in its denominator. As with the previous metric, we introduce a parameter $q$ as this exponent of the variance and allow it to vary over a range of values. Again, an optimal value for $q$ of $\frac{1}{2}$ gives us an indication – though not a guarantee – that our intuition is correct. With this metric, we define the distance matrix as follows:

$$d_{vij} = \begin{cases} \sqrt{(x_{vi} - x_{vj})^2 + (y_{vi} - y_{vj})^2} \left( \frac{1}{\sigma^2} \right)^q, & e_v(i,j) = 1 \\ \text{undef}, & e_v(i,j) = 0 \end{cases}$$

**Metric 5: XY-Variance Adjusted Euclidean Distance**

With the two previous metrics, we approximated a photograph as being an orthographic projection. Orthographic projection is a specific case of parallel projection where the viewing direction is normal to the scene, resulting in the same apparent linear zoom factor in both the $x$ and $y$ directions. If we instead approximate photographs as being general parallel projections, then we decouple the apparent linear zoom factors, allowing for the possibility of a different zoom factor in both the $x$ and $y$ directions.
Figure 28 consists of a scene and an image of it. In the scene, the true distance between the triangle and circle is equal to the distance between the circle and cross.

In this image, the pixel distance between the triangle and circle is 315.75 and that between the circle and cross is 204.02. With the approximation of orthographic projection, regardless of the zoom factor, the first distance will always be greater than the second, although the distances are equal. However, with the approximation of general parallel projection, $x$-zoom and $y$-zoom factors exist such that the two distances are equal.

In our example, supposing that the real distance between these pairs of objects was 300.00 (units), if only a single zoom factor were implemented, its ideal value would be 0.95x, yielding a distance of 300.00 between the triangle and circle, but not between the circle and cross. If both the $x$-zoom and $y$-zoom factors were implemented, their ideal values would be 0.91x and 1.69x respectively, yielding distances of 300.00 between both
pairs of objects. Figure 29 shows the image from Figure 28 expanded by a single zoom factor alongside the same image expanded by both $x$-zoom and $y$-zoom factors.

![Figure 29: A photograph with a single zoom and one with both $x$- and $y$-zoom factors](image)

We note that this example is slightly misleading. With the approximation of parallel projection, any projection of our two-dimensional scene allows for two degrees of freedom in the image, which we implement as the $x$-zoom and $y$-zoom factors. Since we only fit two distances in our example, we are always able to find a solution.

If our example had included another object, then regardless of which zoom factors we chose, the approximation of parallel projection would not be able to yield accurate distances between all of the objects.
Figure 30 illustrates this dilemma using the same scene as above but with an extra object added, also equidistant from the cross. A normal and stretched version of an image of that scene is also shown, as in Figure 29 above. Even with the more general approximation of parallel projection, the distance between the cross and pentagon is 76.85 and not 300.00.

Had we treated photographs as perspective projections instead of approximating them as parallel projections, we would have been able to report all inter-object distances accurately. However, fitting an image with the perspective projection model involves solving for eleven independent parameters for each photograph. Even with simplifying assumptions – such as that the same camera was used to take all of the pictures or that it was always held at approximately the same height and orientation – the model cannot be reduced to contain fewer than seven degrees of freedom per image.
Although we guarantee that each input photograph contains at least two tags, we do not guarantee any more than that, and further, do not expect any to contain more than four or five. This leads to the potential situation of an underdetermined system of equations for some of the photographs. It is therefore impractical – if not entirely impossible – to fit the perspective camera model using our available dataset. For this reason, we stop at parallel projection as our most complex model, and do not inspect any further metrics.

As hinted by the name of this metric, the $x$-zoom and $y$-zoom factors are estimated with the positional variances along the $x$ and $y$ axes of an image. As with the previous metrics, we add a parameter $q$ as the exponent to both of the variances, and allow it to vary over a range of values. The following equations define the distance matrix:

\[
\sigma_{vx}^2 = \frac{1}{n_v} \sum_i (x_{vi} - \bar{x}_v)^2
\]

\[
\sigma_{vy}^2 = \frac{1}{n_v} \sum_i (y_{vi} - \bar{y}_v)^2
\]

\[
d_{vx} = \sqrt{\left(\frac{x_{vi} - x_{vj}}{\sigma_{vx}^q}\right)^2 + \left(\frac{y_{vi} - y_{vj}}{\sigma_{vy}^q}\right)^2}, \quad e_v(i, j) = 1
\]

\[
\text{undef}, \quad e_v(i, j) = 0
\]
3.3  GLOBAL MERGING METHODS

In this section, the individual distance matrices – $D_v$ – generated using the metrics described above are merged together to form a single global distance matrix, $D$. There are many methods that can accomplish this goal, however, we examine only two simple families of merging methods – A) the generalized mean and B) the inverse-sum.

The same scene as was shown in Figure 15 (at the start of this chapter) is shown in Figure 31. The *true* distances between the objects in this scene are given in the accompanying distance matrix.

![Figure 31: A scene and its optimal global distance matrix](image)

The same four images of the above scene that were shown at the start of this chapter are shown again in Figure 32. The respective individual distance matrices (as generated using Metric 5) are given below each of these images.
It is the goal of this section to combine these individual distance matrices into the global distance matrix that best reproduces the true distances in the scene.

![Distance matrices and photographs](image)

Figure 32: Photographs taken of the scene form Figure 15 and their distance matrices
Merging Method A: Generalized Mean

The generalized mean is an obvious method that takes the values from the individual distance matrices and merges them together to obtain a single value. In essence, this method simply averages the distances between the pairs of objects in each of the photographs. A parameter $p$ is included as an exponent, controlling the effect of larger distances over smaller ones.

For example, if $p = 1$, this method degenerates into an arithmetic mean, if $p = 2$, the larger values are biased slightly, and in the limit as $p \to \infty$, the maximum value of all of the individual distances is used in the global matrix. Conversely, if $p = -1$, this method degenerates into a harmonic mean, and expectantly, in the limit as $p \to -\infty$, the minimum value of all of the individual distances is used as in the global matrix. The following equation describes this merging method:

$$d_{ij} = \left[ \frac{1}{n} \sum_{v} e_{v}(i,j)(d_{vij})^p \right]^{1/p}, \quad p \neq 0$$

Figure 33 shows the resulting global distance matrix using this merging method (with $p = 1$) on the four distance matrices shown in Figure 32 above.
**Global Distance Matrix using Merging Method A:**

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>282.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>282.8</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>282.8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>282.8</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>208.1</td>
<td>296.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>208.1</td>
<td>0</td>
<td>242.5</td>
<td>264.1</td>
</tr>
<tr>
<td>3</td>
<td>296.0</td>
<td>242.5</td>
<td>0</td>
<td>242.4</td>
</tr>
<tr>
<td>4</td>
<td>264.1</td>
<td>242.4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 33:** Individual distance matrices merged to form a single global matrix

**Merging Method B: Inverse Sum**

Intuitively, it is expected that two objects that are near one another in a scene appear together in more photographs than two objects that are further apart. Conversely, if two objects appear together in many photographs, their *true* distance in the scene is likely to be smaller than the distance between objects that do not frequently appear together in the set of photographs. An illustration of this is shown in Figure 34.
In Figure 34, we see that the triangle and diamond appear together in more photographs than the circle and diamond. This gives us an indication that the triangle and diamond are closer together than the circle and diamond in the scene. This intuition leads to the formation of this second merging method.

Instead of considering the values in the distance matrix as distances, we can consider them as inverse affinities – the smaller the value, the greater the affinity the pair of objects has for one another. With this paradigm, we propose a merging method that emulates a circuit of resistors. When multiple resistors connect the same two nodes in a circuit, they can be treated as a single equivalent resistor. The equivalent resistance is equal to the inverse-sum of the individual resistances as a result of Ohm’s law, as demonstrated in Figure 35.
The following equation defines the equivalent single resistance for $K$ parallel resistances:

$$R_{eq} = \left[ \sum R_{k}^{-1} \right]^{-1}$$

For this merging method, the individual affinities (the elements of the individual distance matrices) are treated as parallel resistances. The global affinity between two tags is calculated by taking the inverse-sum of the individual affinities. When a pair of tags appears together in multiple photographs, this merging method results in the final value being smaller than any of the individual values. These values correspond to inverse affinities, and therefore, the final affinity between the pair of frequently co-occurring tags increases, adhering to the intuition described at the beginning of this method.

Many possible parameters can be included in the inverse-sum to add degrees of freedom. The previous method included the parameter $p$ as an exponent to control the relative importance of greater operands over smaller ones, and similarly, the same parameter $p$ is
The following equation describes the calculation of the global distance matrix:

\[ d_{ij} = \left[ \sum_v e_{v(i,j)}(d_{vij})^p \right]^{1/p}, \quad p < 0 \]

We note that with merging method A, \( p \) could take on both positive and negative values. Contrastingly, with method B, only for negative \( p \) is the merged affinity greater than that of each of the individual affinities, as per the intuition on which this method is based. As an observation, this merging method results in values that are smaller by a factor of \( \frac{1}{\sqrt{n_{ij}}} \) than the values resulting from the previous merging method, and hence in the limit as \( p \to -\infty \), the two merging methods are equivalent.

Figure 36 shows the resulting global distance matrix using the inverse-sum merging method (with \( p = -1 \)) on the same four distance matrices shown in Figure 32 above.
Global Distance Matrix using Merging Method B:

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>90.6</td>
<td>147.7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>90.6</td>
<td>0</td>
<td>120.7</td>
<td>264.1</td>
</tr>
<tr>
<td>3</td>
<td>147.7</td>
<td>120.7</td>
<td>0</td>
<td>242.4</td>
</tr>
<tr>
<td>4</td>
<td>264.1</td>
<td>242.4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 36: Individual distance matrices merged to form a single global matrix

The optimal distance matrix from the scene in Figure 31 and the global matrices formed by the two merging methods are shown in Figure 37. The matrices have been normalized (with the distance between tags 1 and 2 as 120) for an easier visual comparison.

**A) Generalized Mean:**

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>120</td>
<td>170.7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>0</td>
<td>139.8</td>
<td>152.3</td>
</tr>
<tr>
<td>3</td>
<td>170.7</td>
<td>139.8</td>
<td>0</td>
<td>139.8</td>
</tr>
<tr>
<td>4</td>
<td>152.3</td>
<td>139.8</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Optimal:**

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>120</td>
<td>220</td>
<td>365</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>0</td>
<td>160</td>
<td>245</td>
</tr>
<tr>
<td>3</td>
<td>220</td>
<td>160</td>
<td>0</td>
<td>290</td>
</tr>
<tr>
<td>4</td>
<td>365</td>
<td>245</td>
<td>290</td>
<td>0</td>
</tr>
</tbody>
</table>

**B) Inverse-Sum:**

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>120</td>
<td>159.9</td>
<td>349.8</td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>0</td>
<td>159.9</td>
<td>321.1</td>
</tr>
<tr>
<td>3</td>
<td>195.6</td>
<td>159.9</td>
<td>0</td>
<td>321.1</td>
</tr>
<tr>
<td>4</td>
<td>349.8</td>
<td>321.1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 37: Merged global distance matrices compared with the optimal matrix
After a quick qualitative inspection of the values in the matrices shown in Figure 37, we note that neither merging method gives accurate results when compared with the optimal matrix. Further, neither of these methods are able to assign a value for the distance between tags 1 and 4 simply because there is not enough available tag data.

It is the scope of the work in the following chapter to both fill in the blanks in the global distance matrix and better approximate the optimal distance matrix.
4 FROM MATRIX TO MAP

The goal of our work in this chapter is to take the global distance matrix generated in the previous chapter, and with it, to reconstruct a two-dimensional map of the original scene. We propose a new reconstruction method, which we call Spring Model, and compare the map resulting from our approach to the map resulting from an existing approach for a similar problem in the wireless sensor positioning domain, namely MDS-MAP(P). The procedural flowchart for this chapter is shown in Figure 38.

Figure 38: Procedural flowchart of this chapter
4.1 MAP RECONSTRUCTION

As input to this chapter, we are given a single distance matrix, the result of merging the tag data using a metric described in the previous chapter. Due to the possible lack of data, some of the elements in this matrix may be undefined (left blank). Further, even the defined values of this matrix are noisy due to the nature of their generation, as previously discussed. Hence, there most likely does not exist an exact two-dimensional reconstruction for any given distance matrix.

An adequate solution to this problem should interpolate using the defined values in the global matrix and approximate an appropriate two-dimensional map. Figure 39 illustrates an example of the reconstruction of a two-dimensional map from the global distance matrix that was generated in the previous chapter (from Figure 37).

![Figure 39: Reconstruction of a map from a distance matrix](image)
The problem of generating a two-dimensional map given a distance matrix is not a new one; in positioning of ad-hoc wireless sensor networks, a similar problem exists. Given a slew of wireless sensors distributed throughout a plane, with each sensor having the ability to determine as well as to transmit the distance between itself and within-range sensors, a corresponding distance matrix can be generated. An example of a wireless sensor layout is given in Figure 40. The numbered nodes represent wireless sensors and only within-range sensors are connected by edges (whose lengths are shown).

![Figure 40: Example wireless sensor layout](image)

The sensors can transmit their distances; hence, a master sensor can compile all of the distance information into a single distance matrix. Figure 41 is the distance matrix that corresponds to the sensor layout shown in Figure 40. Again, undefined values in this matrix signify pairs of sensors that are out of range of each other; all other values are the measured sensor-to-sensor distances.
Once the distance matrix is given, the problem of map reconstruction and of sensor positioning appears to be analogous. Therefore, any method used to perform sensor positioning should be able to perform map reconstruction. One such recently published method is MDS-MAP(P), which we describe in detail below.

### 4.2 DESCRIPTION OF MDS-MAP(P)

MDS-MAP(P) is a recently published method used in the reconstruction of the layout of wireless sensor networks. Like our map reconstruction problem, it takes a distance matrix as input. The global distance matrix generated in the previous chapter may contain undefined elements. Similarly, the input matrix to MDS-MAP(P) is allowed (and expected) to contain undefined elements. It outputs a two-dimensional set of node positions, attempting to reconstruct the layout from which the given distance matrix was generated.
MDS-MAP(P) consists of two main stages: the generation of a local map for each node, and the merging of all of these local maps, forming a single global map. These two stages are described in detail below. The procedural flowchart of MDS-MAP(P) is shown in Figure 42.
The first stage of MDS-MAP(P) involves generating a distance matrix for each individual node, known as a local distance matrix. For any given node, its local distance matrix consists of only the nodes that are at most two hops away from it. A ‘hop’ refers to a connected edge in the graphical layout of the sensors and corresponds to a defined value in the input distance matrix.

In the matrix given in Figure 41 above, we note that nodes 1 and 2 are one hop apart, since the corresponding matrix element – \(d_{12}\) – is defined. Nodes 1 and 3 are two hops apart, since \(d_{13}\) is undefined but both \(d_{12}\) and \(d_{23}\) are (therefore, we can hop along the path from node 1 to 2, and then from node 2 to 3). Since nodes 1 and 4 are more than two hops apart, node 4 is not present in the local matrix of node 1, as we can see in Figure 43.

![Figure 43: Local (two-hop) distance matrix for node 1](image)

Although global distance matrices may have undefined elements, local ones are always fully defined. Where the global matrix had an undefined value, an estimate is used as the corresponding value in the local matrix. This estimate is the shortest multi-hop path-
length between the respective pair of nodes. For example, \(d_{13}\) is undefined in the global matrix in Figure 41 (on a previous page), but is defined in the local matrix shown in Figure 43. To get from node 1 to node 3, we could either hop via node 2 or via node 6. However, \(d_{12} + d_{23} = 14\) is less than \(d_{16} + d_{63} = 24.8\); hence 14 is used as this path length.

Since the wireless sensors are all situated on a plane, we expect to be able to generate the local distance matrix from a set of two-dimensional points. However, this is seldom the case due to noise in the distance measurements and the inaccuracies of the multi-hop distance estimates. To enforce a 2-D solution, classical multidimensional scaling is performed, resulting in a set of (2-D) points whose pair-wise distances resemble those of the local distance matrix [12].

The locations of these points are often good initial guesses and are further optimized using the Levenberg-Marquardt algorithm [15] to reduce the squared error in the point-to-point distances. In the cost function of this optimization, more weight is given to the error in the one-hop distances than in the multi-hop estimates to account for the fact that they are only estimates. The final point locations are plotted to form a local map. Continuing with our example, the local map generated for node 1 is shown in Figure 44.
We note that if the pair-wise distances for a set of nodes are all defined, then shifting all of the nodes by the same amount, rotating the nodes around a common centre of rotation, or reflecting the nodes along a common line of symmetry would not change any of the reported distances. Consequently, any reconstruction of a local map is accurate to at most a translation, a rotation, and a reflection when compared to the global map.

Once a local map is generated for each node, the second stage of MDS-MAP(P) combines all of the local maps to form a single global map. Starting with the two local maps that have the greatest number of common nodes, a transformation is applied to one of the maps so that it best aligns with the other. This transformation consists of only a translation, a rotation, and a reflection, as these are the only invariants in the reconstruction of the local maps. The best alignment between two maps corresponds to the least squared error in the positions of their common nodes. Figure 45 shows two local maps before and after the optimal transformation is applied.
Once the optimal transformation is found, the local maps are merged forming a collective map. The positions of the common nodes in this collective map are the averages of the positions of those nodes in the local maps. All other nodes are placed on the collective map in the same position as on their respective (post-transform) local map. The merging process is then continued; the local map with the greatest number of common nodes as the collective map is chosen next. After its optimal alignment transformation is applied, it is merged into the collective map. When all of the local maps are merged into the collective map, MDS-MAP(P) is complete, and this map is output as the global map.

We note that there is a final step to the MDS-MAP(P) algorithm that is only applied if some of the nodes have known positions (anchor nodes). Although this is never the case
with our map reconstruction problem, it may be the case with a layout of wireless sensors. In this final step, a resolution of forces technique – aptly named the Mass-Spring Model – is used to refine the positions of all of the nodes given the fixed locations of the anchor nodes, and is based on the physical spring-mass system.

Although MDS-MAP(P) appears to solve an analogous problem to our map reconstruction problem, we note that there is a fundamental difference. In the input to MDS-MAP(P), the values of the distance matrix are relatively accurate sensor-to-sensor distances as measured by the respective pairs of sensors. The average relative noise in the distance matrix for the wireless sensor networks is 5%.

Conversely, in map reconstruction, the distance matrix is generated from roughly-tagged photographs, and further, uses approximations in its camera model. Hence, regardless of the metric and merging method used, it consists of highly noisy values. The average relative noise in the distance matrix for the test cases was around 45%, an order of magnitude greater than the input noise to MDS-MAP(P).

For this reason, we propose a different reconstruction method – simpler and expectantly more robust when given such noisy distance matrices. Our proposed method – which we call the Spring Model – is based on the Spring-Mass Model used to refine MDS-MAP(P) when given anchor nodes, and is described below.
4.3 DESCRIPTION OF THE SPRING MODEL

The Spring Model is so named because it mimics the movement of a number of nodes attached to one another by ideal springs. Each spring has a resting length equal to the corresponding value in the input distance matrix. Each node represents the position of a tagged object and has a mass equal to the number of springs attached to it. The Spring Model is an iterative process that starts with three nodes, and with every iteration, adds the next node in a specific order. This order ensures that nodes with the greatest number of common connections with other nodes are added first.

During each iteration, the system is set into motion, allowed to resolve the forces applied to the nodes by the various interconnecting springs following the laws of physics. Eventually, when all of nodes have been added, the resulting layout is used as the final output map. This overall process is described in detail below. The procedural flowchart of the Spring Model is shown in Figure 46.

Figure 46: Procedural flowchart of the Spring Model
Our Spring Model idea is based on the Mass-Spring Model refinement step of MDS-MAP (which was briefly mentioned in the previous section). In this refinement step, each node \( i \) is treated as a mass on a plane, with position \( \tilde{p}_i \) corresponding to the reconstruction that led up to that point. The masses are attached to one another with ideal springs of unit stiffness, whose resting lengths are equal to the values of the respective elements of the input distance matrix.

The update equation for a node \( i \) (with \( K \) neighbouring nodes) is given in the following equation. During every iteration, each of the node positions is updated in turn, thereby reducing the error in the node distances. Overall, a total of 20 iterations are used.

\[
\tilde{p}_i = \tilde{p}_i + \frac{1}{K} \sum_k \left( 1 - \frac{d_{ik}}{\| \tilde{p}_k - \tilde{p}_i \|_2} \right) (\tilde{p}_k - \tilde{p}_i)
\]

Our proposed Spring Model is a slight variation of this. However, unlike in MDS-MAP, we do not use Multidimensional scaling as an initial object placement. Instead, we build up this placement using the following approach. For each pair of tags \( i \) and \( j \), we calculate the number of remaining tags that are attached (meaning that they have a defined value in input distance matrix) to both \( i \) and \( j \). The pair with the greatest number of common attachments is chosen first. Sequentially, the remaining tags are added in order of decreasing number of common attachments with one of the previously chosen tags.
Each time a new tag is added (in the form of a node), the system is allowed to run its course, converging to a solution. Although each new node is added to a random position, the existing nodes start off where they converged in the previous iteration. Since the first nodes to be added have the greatest influence on other nodes, as more and more tags are added, the overall structure of the layout of the nodes should not change drastically.

At the start of every iteration, the springs connecting the nodes are either stretched or compressed; hence, likely not at their resting lengths. The system is then set into motion, allowed to run until it converges. Each node is assigned a mass equal to the number of springs attached to it. At every iteration – corresponding to a differential step in time $\Delta t$ – each node is acted upon by the $K$ springs attached to it, resulting in a net force, and hence, an acceleration. This acceleration is defined in the equation below:

$$\ddot{a}_i = \frac{1}{K} \sum_k \left( 1 - \frac{d_{ik}}{\|p_k - p_i\|_2} \right) (p_k - \bar{p})$$

Following Newton’s laws of motion, each object’s velocity and position is updated. To ensure convergence, we further include a velocity-proportional drag factor that controls the effect of momentum on the system. The following are the update equations, with the superscripts $\dot{v}^+$ and $\dot{p}^+$ referring to the velocity and position just after the iteration and the superscripts $\dot{v}^-$ and $\dot{p}^-$ referring to the velocity and position just before the iteration.

$$\ddot{v}_i^+ = \text{drag} \cdot \dot{v}_i^- + \ddot{a}_i \cdot \Delta t$$
$$\ddot{p}_i^+ = \dot{p}_i^- + \dot{v}_i^+ \cdot \Delta t$$
At the end of every iteration, the average absolute velocity – $v_{avg}$ – is computed as shown in the equation below.

$$v_{avg} = \frac{1}{n} \sum_{i} |v_i^+|$$

if $v_{avg} < v_{min}$, stop

If this value drops below a minimum threshold value – $v_{min}$ – then the model is assumed to have converged, and the positions of the objects are recorded. Once all of the nodes have been added to the system, the final iteration results in the output positions of each of the objects, thereby forming a map of the scene.

If we inspect the physical properties of our model, we note that there are only two forms of energy that can be stored: kinetic energy in the moving masses and elastic energy in the stretched or compressed springs. The motion of our model follows the laws of physics; hence, its total energy will never increase. Before we set the model into motion, finite energy is incorporated into it, corresponding to the stretch or compression in the springs as given by Hooke’s law, shown in the following equation:

$$E_{elastic} = \frac{1}{2} \sum_{i} \sum_{j \neq i} (d_{ij} - \|\vec{p}_i - \vec{p}_j\|_2)^2$$

With the presence of a drag factor in our model, at every iteration, some of the overall kinetic energy is lost to drag. Therefore, since we start with finite energy and lose some at every iteration, we are guaranteed to converge to a final solution. The total energy at
convergence is again equal to only the elastic energy. It is interesting to note that this elastic energy is equivalent to the squared error in the distances – only differing by an irrelevant multiplicative factor. Hence, the Spring Model guarantees a reduction in squared error; and the convergent solution of any run is necessarily a local minimum.

In every run of the Spring Model, we used $\Delta t = 0.01$, $drag = 0.931$, and $v_{\text{min}} = 0.1 \cdot \bar{d}$, where $\bar{d}$ is the average distance in the input distance matrix. These values were chosen empirically to speed up the run times of the reconstruction, and do not have a significant effect on the resulting output map.
In each of the previous chapters, the procedural flowcharts included an evaluation step. In this chapter, we describe this evaluation procedure, as well as apply it to the outputs of both of the stages of our approach. For the first stage, we compare the five proposed metrics and two proposed merging methods against each other, whereas for the second stage, we compare our proposed Spring Model to an existing wireless sensor positioning method, MDS-MAP(P). From the results of these experiments, we choose the optimal metric and merging method, and show that our Spring Model performs better than MDS-MAP(P) when given roughly-tagged input.

5.1 EVALUATION METHOD

As described in the previous chapters, the full reconstruction of a map consists of two stages. In the first stage, a distance matrix is generated and in the second, objects are
positioned to most accurately represent the distance matrix generated in the first stage. Once the positions of the objects are established in the second stage, it is trivial to generate a second distance matrix from them, corresponding to their final positions. For this reason, an obvious choice of comparable is the distance matrix as it is readily available with the output of both stages of the reconstruction process.

Inherently, the reconstruction of a map from a distance matrix can only be determined up to a translation, a rotation, and a reflection. Further, since the distance matrix is generated from a set of uncalibrated photographs, the distances are only correct up to a uniform scaling factor. Hence, it is important for an evaluation to be invariant to these transformations. In conforming to these requirements, we chose to employ the k-Nearest Neighbours test (kNNT) as our method of evaluation, due to its appropriateness and simplicity, as we describe below.

In kNNT, a test matrix is compared to a reference matrix. For each object $i$ in the scene, two lists are created. The first list contains the $k$ closest objects to $i$, as reported by the reference matrix; similarly, the second list contains the $k$ closest objects as reported by the test matrix. The number of objects that are found on both lists is the numerator, and the total number of objects found on the list of the test matrix is the denominator, with this ratio recorded as the final kNNT score. The denominator in the score takes into account the possibility that some objects (in a scarcely defined distance matrix) may have
less than $k$ defined distances to other objects. Figure 47 shows an example of a reference matrix and a test matrix, building up to the corresponding kNNT score, with $k = 2$.

![Reference Distance Matrix](image1)

![Test Distance Matrix](image2)

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>20.3</td>
<td>82.1</td>
<td>86.6</td>
<td>88.6</td>
</tr>
<tr>
<td>2</td>
<td>20.3</td>
<td>0</td>
<td>82.4</td>
<td>67.9</td>
<td>73.9</td>
</tr>
<tr>
<td>3</td>
<td>82.1</td>
<td>82.4</td>
<td>0</td>
<td>88.7</td>
<td>65.6</td>
</tr>
<tr>
<td>4</td>
<td>86.6</td>
<td>67.9</td>
<td>88.7</td>
<td>0</td>
<td>28.1</td>
</tr>
<tr>
<td>5</td>
<td>88.6</td>
<td>73.9</td>
<td>65.6</td>
<td>28.1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>13</td>
<td>24</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>0</td>
<td>19</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>19</td>
<td>0</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>26</td>
<td>14</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 47:** An example kNNT score comparison

The resulting kNNT score is calculated as follows:

\[
\frac{\text{Numer}}{\text{Denom}} = \frac{2 + 1 + 0 + 1}{2 + 2 + 2 + 1 + 2} = 55.6\%
\]

Upon a quick analysis of kNNT, we note that it is invariant to translation, rotation, reflection, and uniform scaling. We also note that since only the $k$ closest objects are compared, a bias is added to matrices with more accurate inner-cluster distances. A scene such as the one shown in Figure 48 below consists of several such clusters (higher density areas) of objects, with the clusters relatively spread out. In such a scene, kNNT favours matrices that report more accurate distances for the same-cluster objects and less
for the different-cluster objects over matrices that report more accurate distances on average but not for the same-cluster distances.

![Figure 48: A scene with several clusters of objects (with one cluster circled)](image)

In Figure 48, one of the clusters is circled for clarity, and we expect kNNT to give higher scores for matrices that report more accurate distances between the objects that are found within this circled cluster. With this evaluation method in hand, we carry out our experiments, with the goal of finding the optimal combination of metric and merging method as well as map reconstruction method. For all of our experiments, we used $k = 5$. 
In Chapter 3, we proposed five metrics and two merging methods that extracted the tag data from the input images and output a global distance matrix. Each one of the metrics followed by one of the merging methods is a unique overall combination, which we label 1A, 1B, …, 5B, with the number corresponding to the metric and the letter corresponding to the merging method. These are all compared against each other.

As described in Chapter 3, each of these combinations has two parameters incorporated into it: $q$ and $p$. The parameter $q$ is an exponent of either the number of tags or of the positional variance, depending on the metric. We gave an intuition for why we expect that $q$ should equal $\frac{1}{2}$, however, we allow it to take on a range of values from $-5$ to $+5$ (with a precision of 0.1) so that an optimal value can be identified. We note that Metric 2 is a degenerate case of Metric 3 when $q = 0$. For this reason, we combine the two and only inspect 3A and 3B as possible combinational methods, disregarding 2A and 2B.

The parameter $p$ is an exponent of the operands in the merging methods, allowed to take any non-zero value in the generalized mean (Merging Method A) and only positive values in the inverse-sum mean (Merging Method B). With merging method A, we test the discrete set of values: $\{-\infty, -2, -1, -0.1, 0.1, 1, 2, +\infty\}$, and with merging method B, we test the discrete set of values: $\{-5, -2, -1, -0.1\}$. After the approximate level of
accuracy is found for each method, we further inspect only the method(s) with the highest accuracy, and test a more precise range of values, *exhaustively* finding the optimal combination of parameters $q$ and $p$.

Two test datasets are used in these experiments, taken from photographs of the University of Toronto campus. A tagged satellite map is used to generate the reference distance matrix. The first dataset consists of the entire set of tagged objects, dispersed around the whole campus area. The second dataset consists of only a subset of these tags, condensed within a smaller region of the campus. The satellite maps corresponding to the dispersed and condensed datasets are shown in Figure 49.

![Satellite maps of the dispersed and condensed datasets](image)

Figure 49: Satellite maps of the dispersed and condensed datasets

Table 2 compares the two datasets with respect to the total number of objects, the total number of associated images, the average number of tags per image, the minimum number of tags per image, and the positional variance (spread) of the objects in the
satellite map. The final value compared in this table is the average kNNT score achieved by a random placement of objects, signifying the worst possible score, obtainable if our method was to simply assign random values in the output global distance matrix.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dispersed</th>
<th>Condensed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Objects</td>
<td>67</td>
<td>19</td>
</tr>
<tr>
<td>Number of Images</td>
<td>64</td>
<td>36</td>
</tr>
<tr>
<td>Average Tags/Image</td>
<td>5.7</td>
<td>4.0</td>
</tr>
<tr>
<td>Minimum Tags/Image</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Positional Variance</td>
<td>(1.09 \cdot 10^4)</td>
<td>(2.01 \cdot 10^3)</td>
</tr>
<tr>
<td>Random kNNT Score</td>
<td>7.7 %</td>
<td>27.3 %</td>
</tr>
</tbody>
</table>

Table 2: Comparison between the dispersed and condensed datasets

5.3 CHAPTER 3 RESULTS

In this section, we carry out our experiments, finding the optimal \(q\) and \(p\) for each of the metric and merging method combinations. Since we use two different datasets, the optimal values are chosen that correspond to the highest average kNNT score between the two datasets. Throughout the graphs in this section, blue corresponds to the dispersed dataset, red to the condensed dataset, and grey to the average of the two.
Methods 1A and 1B do not require optimization because Metric 1 does not use $q$ as a parameter, and further, the individual distance matrices only consist of the binary values 0 and 1; hence, $p$ also has no effect on the output. Figure 50 shows the resulting kNNT scores for the two datasets using methods 1A and 1B.

![Figure 50: kNNT for methods 1A and 1B](image)

Methods 3A, 3B, 4A, 4B, 5A, and 5B each use both of the parameters $q$ and $p$. For each of the values of $p$ and $q$ that were tested, Figures 51 to 56 below show graphs of the kNNT scores for the two datasets and the corresponding average score of these methods, leading to the identification of the optimal values of $q$ and $p$. 

70
Figure 51: kNNT scores with Method 3A

Figure 52: kNNT scores with Method 3B
Figure 53: kNNT scores with Method 4A

Figure 54: kNNT scores with Method 4B
Figure 55: kNNT scores with Method 5A

Figure 56: kNNT scores with Method 5B
Table 3 summarizes the results from the graphs above, listing the maximum kNNT score achieved by each method for each discrete \( p \), and at which value of \( q \) this maximum was obtained.

<table>
<thead>
<tr>
<th>3A</th>
<th>Max = 0.662 @ ((q, p) = (3.4, -\infty))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>-( \infty ), -2, -1, -0.1, 0.1, 1, 2, +( \infty )</td>
</tr>
<tr>
<td>( q )</td>
<td>3.4, 3.4, 2.6, 3.0, 3.0, 3.0, 1.6, 1.2</td>
</tr>
<tr>
<td>Max</td>
<td>0.662, 0.661, 0.645, 0.655, 0.655, 0.622, 0.616, 0.566</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3B</th>
<th>Max = 0.682 @ ((q, p) = (2.2, -1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>-1, -2, -1, -0.1, 0.1, 1, 2, -0.01</td>
</tr>
<tr>
<td>( q )</td>
<td>0.6, 2.2, 4.0, 4.4</td>
</tr>
<tr>
<td>Max</td>
<td>0.681, 0.682, 0.653, 0.653</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4A</th>
<th>Max = 0.654 @ ((q, p) = (3.0, -\infty))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>-( \infty ), -2, -1, -0.1, 0.1, 1, 2, +( \infty )</td>
</tr>
<tr>
<td>( q )</td>
<td>3.0, 3.2, 3.8, 1.8, 1.8, 0.6, 0.4, 1.0</td>
</tr>
<tr>
<td>Max</td>
<td>0.654, 0.640, 0.640, 0.628, 0.628, 0.597, 0.605, 0.557</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4B</th>
<th>Max = 0.691 @ ((q, p) = (1.4, -1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>-1, -2, -1, -0.1, 0.1, 1, 2, -0.01</td>
</tr>
<tr>
<td>( q )</td>
<td>1.0, 1.4, 1.6, 1.4</td>
</tr>
<tr>
<td>Max</td>
<td>0.679, 0.691, 0.657, 0.657</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5A</th>
<th>Max = 0.655 @ ((q, p) = (1.2, -\infty))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>-( \infty ), -2, -1, -0.1, 0.1, 1, 2, +( \infty )</td>
</tr>
<tr>
<td>( q )</td>
<td>1.2, 1.2, 1.0, 1.6, 1.6, 1.2, 1.0, 1.8</td>
</tr>
<tr>
<td>Max</td>
<td>0.655, 0.639, 0.642, 0.642, 0.642, 0.647, 0.636, 0.595</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5B</th>
<th>Max = 0.695 @ ((q, p) = (0.6, -1))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>-1, -2, -1, -0.1, 0.1, 1, 2, -0.01</td>
</tr>
<tr>
<td>( q )</td>
<td>1.8, 0.6, 3.2, 3.2</td>
</tr>
<tr>
<td>Max</td>
<td>0.681, 0.695, 0.668, 0.668</td>
</tr>
</tbody>
</table>

Table 3: Summary of results

We note that two methods achieve significantly better results than the other methods, namely 4B and 5B. Since they achieve almost identical results, we proceed to inspect both of them and attempt to pinpoint more precise optimal values of \( p \) and \( q \).
With these methods, of all of the discrete values tested, \( p = -1 \) corresponds to the highest score. Accordingly, the following two experiments test a more precise range surrounding this value. Similarly, with 4B, \( q = 1.4 \) is optimal, and with 5B, \( q = 0.6 \) is optimal; hence, a surrounding region is tested to precisely determine the optimal value. In order to pinpoint both \( q \) and \( p \) simultaneously, we generate a 2-dimensional plot using 4B and one using 5B, exhaustively searching \( p \in [-2, -0.01] \) and \( q \in [0, 2] \), with a precision of 0.01.

These plots show that the most accurate combinational method is 5B, attaining a kNNT score of 69.9%, with optimal parameters: \( q = 1.32 \) and \( p = -1.41 \). However, this is not an unequivocal victory, as it is only marginally better than the 69.4% achieved with 4B.
With both plots, we note a trend of marginally higher scores in the region of $p = -1$ and for $q$ between 1.0 and 1.5, although the entire tested range of parameters leads to scores between 65% and 70% – not a very wide spread. In Chapter 3, we gave an intuition for why $q = 0.5$ is the expected optimum for the metrics 3, 4, and 5. Although we do note a distinct increase in accuracy around $q = 0.5$ on both of the plots in Figure 57, there is no immediate decrease in accuracy with larger $q$. Similarly, in the graphs generated by all of the other methods (seen in Figures 51 to 56), there a similar increase around $q = 0.5$, but no clear trend other than that the optimal $q$ is positive.

In conclusion to these Chapter 3 experiments, we note that the inverse-sum merging method consistently outperforms the general mean merging method. Furthermore, the refinement of the metrics as described in Chapter 3 is creditable, as later metrics result in higher accuracies than do earlier ones. Although the intuition for the value of $q$ is not justified in practice, the resulting methods are able to generate global distance matrices with average kNNT scores of between 60% and 70% for both datasets.
The goal of the work in Chapter 4 is to take an input global distance matrix, possibly with several undefined elements, and to reconstruct a map of the scene. After the map is reconstructed, it is trivial to generate a distance matrix from it, with which we evaluate the reconstruction method that was used. Two such methods are compared against each other: our proposed Spring Model and MDS-MAP(P), both described in Chapter 4.

We evaluate two aspects of these reconstruction methods: their ability to compensate for noisy distances (variance) and for undefined values (sparsity) in the input distance matrix. In the following experiments, a distance matrix – $D$ – is generated by randomly placing $n$ points on a plane. With $\bar{d}$ as the average distance in $D$, Gaussian noise with mean 0 and variance $(\sigma \bar{d})^2$ is added to the distances, using the absolute value of the result to ensure positive distances. Further, each element is associated with a binary random variable, $\pi_{ij}$ with probability $s_{ij}$, which determines if it is removed, thereby being treated as undefined. The resulting altered matrix is a test matrix – $T$.

Regardless of the variance and sparsity of a distance matrix, all diagonal elements are always equal to 0, as they are the distances between an object and itself. Further, since distance matrices are always symmetric, only the upper triangular elements are altered with the same alterations being applied to the corresponding lower triangular elements.
Hence, a test matrix is defined as follows:

\[
\bar{d} = \frac{1}{n(n-1)} \sum_i \sum_j d_{ij}
\]

\[
t_{ij} = t_{ji} = \begin{cases} 
0, & i = j \\
|d_{ij} + \mathcal{N}(0, \sigma\bar{d})|, & i < j, \quad \pi_{ij} = 1 \\
\text{undef}, & i < j, \quad \pi_{ij} = 0
\end{cases}
\]

The two datasets used in the Chapter 3 experiments led to the selection of Method 5B as the optimal combination to generate a global distance matrix. The matrices generated using 5B for each of the respective datasets have an associated sparsity and variance, as given in Table 4. Since any given global distance matrix is generated with no regard to overall scale, to estimate its variance, a scaling factor is optimized such that the variance between the scaled matrix and a reference matrix is minimized. This final value is reported as the variance of the given matrix.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dispersed</th>
<th>Condensed</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>67</td>
<td>19</td>
</tr>
<tr>
<td>Sparsity</td>
<td>0.193</td>
<td>0.509</td>
</tr>
<tr>
<td>Variance</td>
<td>0.364</td>
<td>0.524</td>
</tr>
</tbody>
</table>

Table 4: Comparison between the dispersed and condensed datasets
With our experiments in this section, we compare the Spring Model and MDS-MAP(P) under these sparsity and variance conditions. Consequently, we conduct two experiments, one with $n = 67$, and the other with $n = 19$. In each experiment, a test distance matrix $T$ (which is an alteration of $D$ as previously discussed) is given to the two reconstruction methods whose outputs are then compared against $D$ using kNNT. To inspect a region that surrounds these two datasets, variance and saturation are incorporated into the test matrices with the discrete values of $\sigma \in \{0, 0.2, 0.4, 0.6\}$ and $s_1 \in \{0.1, 0.2, \ldots, 1\}$.

For each value of $\sigma$ and $s_1$, 10 test matrices are generated. The average kNNT scores over these 10 test reconstructions are recorded for each of the methods. Figures 58 and 59 show the results of the experiments. In the figures, the thicker blue bars correspond to the average kNNT scores achieved by MDS-MAP(P) and the thinner red bars correspond to the average kNNT scores achieved by the Spring Model.

Figure 58: kNNT scores achieved using MDS-MAP(P) and the Spring Model for $n = 67$
Figure 59: kNNT scores achieved using MDS-MAP(P) and the Spring Model for $n = 19$

For ease of visualization, Figure 60 consists of the same graphs as shown in the two figures above, overlaying the results achieved with MDS-MAP(P) and the Spring Model for both datasets.

Figure 60: Overlays of the above graphs for $n = 67$ and $n = 19$
From these graphs, we note that in general, MDS-MAP(P) outperforms the Spring Model only in the case of low variance, and achieves only slightly more accurate results. With higher variance however, the Spring Model leads to better results regardless of sparsity, and further, the loss of accuracy with increased variance is greater in MDS-MAP(P) than it is in the Spring Model. Hence, for an input distance matrix with a large level of variance, the Spring Model is a better choice of reconstruction method.

In the $n = 67$ experiment, sparsity does not have a noticeable effect on either of the reconstruction methods until it is reduced to a level of about 0.3, whereas in the $n = 19$ experiment, a noticeable effect occurs at 0.5, a greater saturation level. This is a result because with a larger input matrix, even with reduced sparsity, there are still enough defined elements to perform reconstruction. For sparsity levels lower than the knee points given above, the Spring Model again outperforms MDS-MAP(P).

In conclusion to the Chapter 4 experiments, we note that the Spring Model outperforms MDS-MAP(P) in the regions of variance and sparsity expected in our input distance matrices. Using methods 4B and 5B followed by the Spring Model, we perform a full reconstruction of both of our test datasets. Table 5 gives the kNNT score for a random object placement (as described earlier in this section), the expected score as extrapolated from the graphs in Figure 58 and Figure 59, and the actual score obtained with our reconstructions using methods 4B and 5B.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dispersed</th>
<th>Condensed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>67</td>
<td>19</td>
</tr>
<tr>
<td>Random kNNT Score</td>
<td>7.7 %</td>
<td>27.3 %</td>
</tr>
<tr>
<td>Expected kNNT Score</td>
<td>38.7 %</td>
<td>54.2 %</td>
</tr>
<tr>
<td>kNNT Score with 4B</td>
<td>40.0 %</td>
<td>62.1 %</td>
</tr>
<tr>
<td>kNNT Score with 5B</td>
<td>46.9 %</td>
<td>66.3 %</td>
</tr>
</tbody>
</table>

Table 5: Results of the reconstructions using our finalized approach

From these results, we see that our reconstruction approach is able to generate maps from roughly-tagged images with kNNT scores that are about 40% better than the scores of random matrices, and about 10% better than the expected performance level as extrapolated from the results of the Chapter 4 experiments. For both $n = 67$ and $n = 19$, Method 5B performs slightly better than Method 4B. Hence, the optimal two-stage approach is Method 5B followed by the Spring Model. In the following chapter, we apply this approach on several sample datasets, getting a qualitative visual sense of the reconstructions accuracies.
6 SAMPLE APPLICATIONS

In this chapter, we apply the finalized approach determined in the previous chapter to reconstruct maps from various sample datasets. Three sample datasets are explored: a small set if images taken of a graduate student office, a larger set of images taken of a shopping centre, and a set of images taken of a group of people at a wedding. The photographs from these samples were given as input to Photosynth, but it was unable perform any reconstruction from them.

6.1 OFFICE MAP RECONSTRUCTION

The first sample dataset consists of 5 photographs taken of a graduate student office, consisting of 8 student desks, a whiteboard, and a window, for a total of 10 tags. The corresponding tagged photographs are shown in Figure 61.
Figure 61: Tagged photographs of a graduate student office

Figure 62 shows the floor plan and seating arrangement of the office as well as the output map generated by our reconstruction method.

After the appropriate rotation, translation, and potential reflection is applied to the reconstructed map, it is superimposed onto the floor plan as shown in Figure 63.
We qualitatively see from the above superposition that although the exact locations of the tags are inaccurate, their relative positions do represent the outlying structure of the actual floor plan.

6.2 SHOPPING CENTRE MAP RECONSTRUCTION

The second sample dataset consists of 41 photographs taken of a shopping centre, with various stores, seating sections, and attractions tagged in the photographs for a total of 53
tags. Due to this large number of tags in this dataset, to avoid visual confusion, the tag names have been changed into numbers. The tagged photographs are shown in Figure 64.
Figure 65 shows the official map of the shopping centre as found on the mall directory as well as the output map generated with our reconstruction method.

After the appropriate rotation, translation, and potential reflection is applied to the reconstructed map, it is superimposed onto the directory as shown in Figure 66.

Figure 65: Mall directory and corresponding map reconstruction

Figure 66: Superposition of the map reconstruction onto the mall directory
This same superposition is shown again in Figure 67, but this time is also marked with four regions grouping the tags that lie in the same arm of the mall.

Figure 67: Superposition of the mall map onto the directory

In this reconstruction, there is not as clear of a representation of the true layout of the mall as there was in the previous sample reconstruction. However, amidst the visual chaos, we are still able to distinguish four denser regions of tags that correspond to the four arms of the mall. The tags within these regions are not accurately positioned, but do belong within that respective arm of the mall. Overall, map reconstruction on this sample dataset is able to qualitatively suggest the high-level structure of the mall – that it has four branching arms at approximately right angles to each other – and in which of these branches the tags are found.
6.3 WEDDING MAP RECONSTRUCTION

The final dataset consists of 28 photographs taken at a wedding, with 17 tagged guests, shown in Figure 68.

![Figure 68: Tagged photographs of a wedding](image-url)
In the above photographs, the subjects were not stationary as in the two previous sample datasets; hence, the generated reconstruction is not expected to be a geographical map, but instead an indication of the relationships of the attendees. The “map” reconstruction is shown in Figure 69.

![Figure 69: Reconstruction from photographs of a wedding](image)

In this reconstruction, we see that there is somewhat of a split between the families and friends of the bride and groom, with the groom’s guests noticeably placed in the top half, and the bride’s placed in the bottom. At the centre, we find the priest – a neutral party. Immediately surrounding the newlyweds are their parents, and further away are the rest of the members of the bridal party and more distant relatives.
When given datasets with tags that are not stationary, map reconstruction is in essence just performing dimensionality reduction on whatever type of data is actually being represented by the extracted distances. It is interesting to note that in the case of a wedding, this process leads to an intuitive visual representation of the family, centered around the bride and groom.

Overall, the map reconstructions of these sample datasets have shown that although our approach is not able to very accurately reconstruct the map of a scene, it is capable of registering the scene’s higher-level structure. This is a good start, considering that Photosynth was not able to generate anything with the given photographs.
7 CONCLUSIONS

In this thesis, we have proposed a method to reconstruct the map of a scene when given tagged photographs of that scene. In order to easily compare our approach with a previously existing approach (MDS-MAP(P)) from the field of wireless sensor positioning, we split our approach into two stages. In the first stage, we proposed several metrics with which individual distance matrices are generated from the tag data associated with the individual input photographs, and two merging methods with which the individual distance matrices were merged into a single global matrix. All possible combinations of metrics and merging methods were compared against each other. The optimal combination was found to be xy-variance adjusted Euclidean distance metric followed by the inverse-sum merging method.

In the second stage, we proposed the Spring Model – a reconstruction method based on the spring-mass system. We compared our approach with MDS-MAP(P), and found that for the type of roughly-tagged input photographs that are inherent to this problem, the Spring Model produced better results. In all of our experiments, we used kNNT scores as
our evaluation methods, and used two test datasets, one with a total of 67 tags, and the other with a total of 19. The optimal overall reconstruction method was able to produce results with kNNT scores of 47% and 66% respectively for the two test datasets.

We then applied this optimal reconstruction method to three sample datasets: a small office with a few desks, a shopping centre with many stores, and a wedding with non-stationary subjects. The overall qualitative results of these reconstructions proved to be fairly accurate in the case of the office dataset. With the larger shopping centre dataset, the resulting map was adequate; not completely accurate, but representing of the general layout of the mall. Further, it correctly identified within which regions of the mall each tag was located. With the wedding dataset, we found that although the subjects were not stationary, the map reconstruction was able to output a relationship representation, showing a clear distinction between the families of the bride and groom. As well, at the centre were the newlyweds, and immediately surrounding them were their parents followed by the remaining guests.

In conclusion, the proposed map reconstruction has the potential to generate both a quantitatively and qualitatively accurate map of a scene from tagged photographs of that scene, providing a good basis upon which to build on for future work.
7.1 SUMMARY OF CONTRIBUTIONS

The following contributions were made with this work:

- A metric and merging method was identified that transforms the tag data associated with input photographs into a global distance matrix.

- A superior method was proposed that transforms a global distance matrix generated from a set of roughly tagged photographs into a map.

- When these two stages follow one another in order, the resulting novel approach is able to reconstruct a map when given tagged input photographs.

7.2 FUTURE WORK

Future work should be done in several areas. In the first stage, we only inspected five metrics and two merging methods. Many other metrics and merging methods exist, leading to potentially higher kNNT scores for the global distance matrices. In particular, the estimation of zoom can be attempted using hidden variables for example, optimized for the minimization of overall variance in the distances between two tags in every image in which they appear.
In the second stage, our Spring Model is guaranteed to converge at a local minimum. If we were able to ensure that it reach a global minimum, the reconstruction accuracies and then robustness of the method would be increased. Further, an investigation can be done on the effects of the order in which nodes are sequentially added.

In this work, we limited our maps to within two dimensions. Scenes are 3-D, however; thus, generating 3-D spatial reconstructions would be a logical future direction. Furthermore, combining image-processing with tag-based map reconstruction would also be synergetic. Using map reconstruction as an initial basis for the layout of a scene would greatly reduce the amount of image-processing required to analyze the scene.

Lastly, a future direction is using our map reconstruction technology to develop the various applications that gave motivation to this work. Especially, starting up a database of tagged images of various scenes (which already exist online, and need only be added) and displaying their corresponding generated maps – using an infrastructure that enables the end goal that is mapping the entire world.
BIBLIOGRAPHY


