Optimized Building Placement in a Polygonal Bounding Area Considering Natural Hazards

A Special Problem by

Sean Francis N. Ballais
2015-04562
BS Computer Science

Presented to the Faculty of the
Division of Natural Sciences and Mathematics

In Partial Fulfillment of the Requirements
For the Degree of
Bachelor of Science in Computer Science

University of the Philippines Visayas
TACLOBAN COLLEGE
Tacloban City

Month Year

This special problem, entitled "**OPTIMIZED BUILDING PLACEMENT IN A POLYGONAL BOUNDING AREA CONSIDERING NATURAL HAZARDS**", prepared and submitted by **SEAN FRANCIS N. BALLAIS**, in partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE** is hereby accepted.

PROF. VICTOR M. ROMERO II
Special Problem Adviser

Accepted as partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE**.

DR. EULITO V. CASAS JR.
Chair, DNSM

Acknowledgements

First of all I would like to thank the Lord for his guidance during the course of my research and for making this thesis possible. I would like to thank my family who served as my inspiration, and never failed to support me all throughout my studies. Thanks to ...

Abstract

Although the abstract is the first thing that appears in the thesis, it is best written last after you have written your conclusion. It should contain spell out your thesis problem and describe your solution clearly.

Make sure your abstract fits in one page !

Table of Contents

Acknowledgements **iii**

Abstract **iv**

Table of Contents **vi**

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**
1.1 Facility Layout Problem 2
The Basic Mathematical Model 4
Discrete vs Continuous Formulations 5
Static vs Dynamic Facility Layout Problems 5
1.2 Polygonally-Bounded Unequal Area Static Facility Layout Problem . 5

2 Review of Related Literature **6**

3 Statement of the Problem **8**

4 Objectives **9**

5 Proposed Methodology **10**
5.1 Image Preprocessing 10
Corner Detection 10
Region Segmentation 11
5.2 Curve Approximation 12
Point Sequence Generation 12
Point Sequence Curve Approximation 13

5.3 Region Colouring and Merging 20

6 Describing How You Validated Your Approach. 22

7 Stating Your Results and Drawing Insights From Them. 23

8 Summarizing Your Thesis and Drawing Your Conclusions. 24

A What should be in the Appendix 25

Bibliography 26

List of Tables

List of Figures

5.1 The network architecture of the Point Parametrization Network (PPN) 15
5.2 The network architecture of the Knot Selection Network (PPN) . . . 18

Chapter 1

Introduction

In November 8, 2013, Super Typhoon Yolanda (internationally known as Haiyan) made landfall in the Philippines. 14 million people were affected, and a total of $5.8 billion in damages were done [19]. More recently, at the time of writing, in November of 2020, Cagayan Valley, Philippines was wreaked with flooding and landslides which resulting in the loss of lives, and affected local infrastructure [2]. The aforementioned events are just two of the many natural disasters that affected the world. Many more are to arrive in the future.

Despite all the calamities that will be affecting communities around the world, building constructions and development will still take place. As such, developers, architects, and engineers must take into account natural hazards. This consideration for natural hazards is important, especially given today's world's attitude towards sustainability [17]. Sustainability deals with ensuring that the demands today are met, while ensuring that future generations can still meet their own demands. It involves meeting the needs of the environment, society, and economy in a balanced manner [4]. None of the three must be compromised when satisfying one of them. Natural risk hazard mitigation is a part of sustainability. Damages by natural disasters impact

the environment, society, and the economy. Key business and community functions get disrupted, economic losses will be incurred, and resources will be depleted and energy will expended as a result of rebuilding efforts. Also, as the United States's Federal Emergency Management Agency has noted, building in high hazard areas will result in higher costs in the long-term due to disaster recovery when natural disasters strike [23]. Building structures with natural hazard mitigation in mind will result in cheaper costs (environmentally-wise, economically-wise, and socially-wise) during post-disaster recovery as well as minimizing economic and environmental impacts, which results in improved sustainability [17].

Many aspects contribute to the resilience of a building against the forces of nature. Building design and construction quality generally affect how a building withstands against flooding, storms, earthquakes [12], and similar events. However, a key aspect of natural hazard mitigation is the location of buildings. This holds especially true when developers seek to mitigate against flooding, landslides, tsunamis, and storm surges [24]. The problem, however, is determining the locations of buildings in as optimal as possible way. This research work seeks to deal with that problem. There have been many related researches already. All of them, including this problem, is classified under the **facility layout problem**.

1.1 Facility Layout Problem

The problem of arranging a set of facilities and/or machines in a pre-determined area, or a set of possible locations (such as in the work of Farmakis, P., and Chassiakos, A. [6]) is called the facility layout problem (FLP). The facilities and/or machines are arranged in such a way that the resulting layout is in line with some criteria

or objectives and under certain constraints. These constraints, which must not be violated, include shape, size, orientation, pick-up/drop-off points [11], and usable area [10]. Facilities and/or machines must also not overlap. Solutions that satisfy the aforementioned conditions are called feasible solutions [15].

Numerous fields have already been solving specific instances of the facility layout problem. // Continue.

Generally, the facility layout problem is considered to be an **NP-Hard** problem [5]. Hosseini-Nasab, H., Fereidouni, S., and Fatemi, S. have noted in their systematic review of FLP that most researches dealing with the facility layout problem model their problems either as a quadratic assignment problem (QAP) or a mixed integer programming problem [11]. According to Drira, A., Pierreval, H., and Hajri-Gabouj, S., the former is sometimes used in discrete FLP formulations, while the latter is often used in continuous formulations [5]. Discrete and continuous FLP formulations will be discussed later. **Quadratic assignment problems** deal with placing n facilities in n locations in such a way that minimizes the assignment cost. The assignment cost is the sum of all facility pairs's flow rate between each other multiplied by their flow rate [1]. This assignment cost is commonly seen in many FLP researches, as we will discuss later. QAP is also known to be an NP-Hard problem [9]. It should be noted though that *some* instances of QAP are easy to solve [7]. The other modeling framework, **mixed integer programming**, can solve problems with both discrete decisions and continuous variables. An example of such problem is the assignment problem [20], which the FLP can be classified under. In this formulation, a set of integer and real-valued integers are being optimized based on an objective function that is being minimized or maximized, while satisfying constraints which are linear

equations or inequalities [25]. Mixed integer programming, when in the context of optimization, is also known to be NP-Hard [20]. These two formulations being known to be generally NP-Hard proves that FLP is indeed generally NP-Hard.

The fact that FLP is an NP-Hard problem has resulted in many research works that utilize heuristics (such as simulated annealing and genetic algorithms). Note that there are also works that utilize exact methods, which seek to find the *optimal* solution for a problem. However, the NP-Hard nature of the FLP prevents them from finding the solution in large problems within reasonable time [3].

The Basic Mathematical Model

Each problem instances of the facility layout problem naturally will have their own mathematical models tailor-fit for their problem instance. Nevertheless, based on our observations and from readings, most of those models are derivatives of or use (such as in [8], [13], and [16]) what will be calling a basic minimization function, which is defined as:

$$\min F = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} f_{ij} d_{ij}$$

where N is the number of facilities, c_{ij} is the cost of handling materials between locations i and j, f_{ij} is the flow rate between i and j, and d_{ij} is the distance between the centroids of i and j. The distance function may differ from work to work. For example, Liu, J., et. al. uses the Manhattan distance in their work [14], while in the work of Ripon, K. S. N., et. al., Euclidean distance was used [21]. In works that derive from this formula, such as in [6], [22], and [18], it was observed that d_{ij}, or a similar variable or expression, is commonly present in the work's objective function

while c_{ij} and f_{ij} may be present and/or the work uses more or fewer variables.

Drira, A., Pierreval, H., and Hajri-Gabouj, S. note the same observation but showcase a slightly differing formula in their 2007 survey of facility layout problems. Unlike the basic minimization formula above, their formula has f_{ij} and c_{ij} combined. They also note that the function above is typically used in continuous formulations of the FLP. The discrete formulation uses a similar function, but ensures that a facility is only in one location, a location only contains one facility, and makes sure that only pairs of locations that contain facilities contribute to the fitness value of a solution. Additionally, they mention that the function is also subject to the following constraints: (1) facilities must obviously not overlap with one another, and (2) the total area used by the facilities must be equal to or less than the allotted area [5]. These constraints have been observed to be generally in many FLP works.

It is worth mentioning that the objective function and constraints in FLP works may have additional modifications to the basic minimization function to suit their problem instances.

Discrete vs Continuous Formulations

Static vs Dynamic Facility Layout Problems

1.2 Polygonally-Bounded Unequal Area Static Facility Layout Problem

Chapter 2

Review of Related Literature

The polygonal bounding area building placement problem (PBABPP) deals with the arrangement of buildings within a polygonally-shaped area. As far as the authors know, no previous research has been conducted for the problem. More so with the fact that this research also takes into account areas prone to natural hazards, namely flooding and landslides. Nevertheless, PBABPP is still an extension of the facility layout problems (FLP), which delves with determining the placement of various assets in a facility. Many techniques utilized in solving FLP instances can be adapted in PBABPP. As such, this literature review will mostly consist of prior works that attempt to solve facility layout problems. Majority of the FLP works included here utilize approximation methods since this work uses one. FLP researches are also easy to find thanks to the fact that it is NP-Hard. Being NP-Hard resulted in numerous researches being done for the field [5].

All research works in the literature produce layouts with varying degrees of fitness and performance. The process of layout generation differ from proposal to proposal based on the specific FLP instance they are working on. Various techniques are used to solve the FLP. Exact methods have been used, but stochastic-based methods like

local search and population-based evolutionary algorithms are popular.

The instance of the facility layout problem that is most related to this work is the unequal area static facility layout problem (UA-SFLP). The works dealing with the UA-SFLP (and even the unequal-area dynamic facility layout problem) use a rectangle to mark the bounds of the area where assets can be placed. This is unlike the problem we are solving here where a polygonal area is used instead.

Chapter 3

Statement of the Problem

Raster images are composed of a pixel matrix. This makes their data representation simple. However, this reduces the amount of detail and quality they have. This limitation is clearer when scaling raster images. This motivates the use of an alternative form of representing images. One form is vector images, which uses mathematical equations to represent an image. The process of converting a raster image to a vector image is called *vectorization*. Semi-structured imagery, such as those used in graphic designs, is one of the classes of images that would benefit from vectorization. This process will allow such images to be easily scaled without sacrificing quality nor detail.

There have been numerous works that tackles image vectorization for semi-structured images. Many of these works primarily use a curve optimization algorithm such as variants of NEWUOA and conjugate gradient. A machine learning approach has been used in one of the works, but as a preprocessing step only [**?**]. Deep learning have been used to solve problems in multiple domains such as natural language processing (NLP), object detection, and playing board games. No other known work has applied deep learning as a core step in image vectorization. This study will deal with such application.

Chapter 4

Objectives

This study is primarily aims to apply deep learning via artificial neural networks to the problem of vectorizing semi-structured imagery. However, there are still some key objectives that this study seeks to accomplish:

1. To develop an approach that considers Gestalt psychology.
2. To evaluate the effectiveness of using deep learning for image vectorization.
3. To evaluate the accuracy of results obtained from the proposed approach to the target raster inputs.
4. To evaluate and compare the results of the proposed approach to that of previous semi-structured image vectorization methods.
5. To evaluate and compare the speed of the proposed approach compared to previous semi-structured image vectorization methods.

Chapter 5

Proposed Methodology

The proposed methodology will be based on the frameworks proposed by Hoshyari, S., et. al. [**?**], Yang, M., et. al. [**?**], Xiong, X., et. al. [**?**], and Laube, P., et. al. [**?**]. Additionally, human perception will also be taken account. Thus, the Gestalt psychology principles of accuracy, simplicity, continuity, and closure, as taken from Hoshyari, S., et. al., will be taken into account as well.

5.1 Image Preprocessing

Before a raster input image can be fitted with curves, it must preprocessed to simplify the vectorization procedure and align it with .

Corner Detection

The first step in the approach is detecting the corners in the input image. This is an important step as it will allow us to enforce the simplicity principle in Gestalt psychology and make sure the resulting vectorization be C^0 continuous should the raster input be as such.

This stage will be based from the corner detection classifier of the work by Hoshyari, S., et. al. [**?**]. A random forest classifier is used in the said work. Supervised learning is used as corners are manually annotated. Annotated corners will not be specific pixels. Rather, corners will be between at least two pixels. Training data is available publicly provided by the researchers. However, such data is limited only to quantized data (i.e. aliased data). The target raster input is expected to be anti-aliased data. As such, the training data will have to be built from scratch to support anti-aliased data.

Region Segmentation

In line again with the simplicity principle, the input image must be divided into regions. This will result in simpler curves being used in the final vectorization output. The additional benefit of segmenting the input into multiple distinct regions is the possibility for parallelism to be used in the vectorization approach. Since each region is distinct and independent from one another, multiple regions can be vectorized at the same time. Thus, speeding up the vectorization process.

Due to the nature of semi-structured imagery where each region will only contain a single colour, a scanline-based approach can be used in this stage. The scanline algorithm will be based off of the scanline algorithm used for boundary pixel detection in the work of Xiong, X., et. al..

For each line l_i, where $0 \leq i < h \mid h$ is the height of input image, in the raster input, each pixel p_i will be assigned to a pixel set r_{ij}, where j is the index to a set in l_i. Each pixel set will contain horizontally adjacent pixels that have the same or near-same colours. We include pixels whose colours are within a certain threshold k from the colour of the pixels that is most prominent in the set. This threshold is

necessary due to the fact that certain parts of a regions may contain an anti-aliased pixel. Each l_i will have a pixel set vector r_i containing all pixel sets of l_i:

$$r_i = (r_{i0}, r_{i1}, \ldots, r_{i(j-1)}, r_{ij})$$

Consequently, a region vector r will contain all pixel set vectors.

$$r = (r_0, \ldots, r_i)$$

Note that there is an opportunity to utilize parallelism, as shown in the work of Xiong, X., et. al. [**?**], during this step due to the independent nature of every line in the raster input.

Once we obtain all the pixel sets for each line, we iterate through r, $(h-2)$ times. For each iteration, we process r_i and r_{i+1}. If there are any r_{ij_α} whose pixels are vertically adjacent and have the same colour (or within k) to another pixel set $r_{(i+1)j_\beta}$, then those two pixel sets are merged into one. By the end of the iterations, we have obtained a set of regions that we can individually vectorize.

5.2 Curve Approximation

The core step of the proposed approach is curve approximation. This stage fits curves to the region boundaries of the raster input. This stage is based on the work by Laube, P., et. al. on curve approximation on point sequences using deep learning [**?**].

Point Sequence Generation

The work of Laube, P., et. al. takes a point sequence as input. As such, for this proposed approach, we must generate point sequences from the region we will be

vectorizing. For every region we ought to vectorize, we treat the center of boundary pixels and the detected corners of each region as a point sequence. However, we must also take into account the fact that certain portions of a region may be a corner where the curves in such segment would have C_0 continuity. As such, the point sequence we generate must take into account corners. This would implore us to take note of the following cases during point sequence generation:

1. For regions with no corners, a random pixel will be selected as both start and end point of the point sequence. The expectation is that there will be no difference in the resulting curve from choosing a different start and end point.

2. For regions with a single corner, the corner point will be selected as the start and end point of the point sequence. This is to ensure that the resulting vector output will have a corner at that point.

3. For regions with two or more corners and assuming n is the number of corners, the point sequence will be divided at those corners into separate point sequences. This will result in $(n + 1)$ new point sequences. Each new point sequence will have their start and end points be the corner points they are adjacent to.

Each point sequence will then be passed to the next stage to be parametrized and have a curve approximated for.

Point Sequence Curve Approximation

The point sequences obtained from the previous step are now to be fitted with curves, specifically B-splines. As provided by the framework by Laube, P., et. al., two neural networks will be used in this stage and some preprocessing will be performed on

the point sequences. The two neural networks are a point parametrization network (PPN), which approximates parametic values to point sequences, and a knot selection network (KSN), which predicts new knot values for knot vector refinement.

Sequence Segmentation

The input point sequence must first be split. This is to ensure that real data and training data match in terms of complexity. Let us define a function $\hat{k}(p)$ that measures the complexity of a point sequence p, where k_i is the curvature at point p_i, given its total curvature:

$$\hat{k}(p) = \sum_{i=0}^{m-1} \frac{(|k_i| + |k_{i+1}|)||p_{i+1} - p_i||_2}{2}$$

A point sequence p is split into point sequence segments $p^s, s = 1, \ldots, r$ at the median, if $k\hat{(}p) > \hat{k_t}$ for a threshold $\hat{k_t}$. This $\hat{k_t}$ will be set, as per the original authors have done, to the 98th percentile of $\hat{k}(\cdot)$ of the training set. This process is performed $r - 1$ times, until each p^s satisfies $k\hat{(}p) > \hat{k_t}$.

Sub/Supersampling and Normalization

To be able to approximate parametric and knot values using the PPN and KSN, the number of points per segment p^s must equal the input size l of the aforementioned networks. As such, all segments p^s are either subsampled or supersampled.

If a segment p^s has a number of points greater than l, then p^s is subsampled. This process involves drawing points in p^s such that the drawn indices i are equally distributed and include the first and last point. If, on the other hand, the number of points in p^s is less than l, then *temporary* points are linearly interpolated between consecutive points p_i^s and p_{i+1}^s. This interpolation is performed until the number of

points equal l.

The sampled segments are then normalized to $\bar{p}^s$, which consists of the points

$$\bar{p}_i^s = \frac{p_i^s - min(p^s)}{max(p^s) - min(p^s)}$$

where $min(p^s)$ and $max(p^s)$ are the minimum and maximum coordinates of p^s respectively.

Parametrization of Point Segments

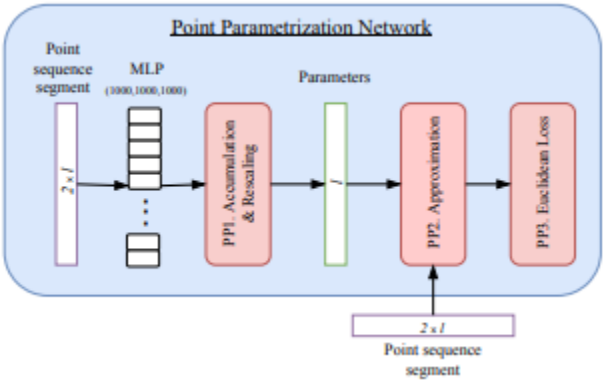

Figure 5.1: The network architecture of the Point Parametrization Network (PPN)

.

The PPN will be responsible for parametrization of point segments. For every $\bar{p}^s$, the PPN generates a parametrization $\bar{t}^s \subset [0, 1]$, which will be rescaled to $[u_{s-1}, u_s]$ and adapted to sampling of $\bar{p}^s$.

In supersampled segments, the parameters t_i^s of temporary points are simply removed from $\bar{t}^s$. In a subsampled p^s, for every point p_i that was removed from the segment, a parameter $\bar{t}_i$ is inserted to $\bar{t}^s$.

$$t_i = t_\alpha^s + (t_\beta^s)\frac{chordlen(p_\alpha^s, p_i)}{chordlen(p_\alpha^s, p_\beta^s)}$$

chordlen is the length of the polygon defined by a point sequence. In the sub-sampled segment, with parameters t^s_α and t^s_β, p^s_α and p^s_β are the closest neighbours of p_i.

The initialization of the parametric step requires an initial knot vector. We first define $u_0 = 0$ and $u_n = 1$. For each segment (except the last one), one knot u_i is added.

$$u_i = u_{i-1)} + \frac{chordlen(p^s)}{chordlen(p)}, i = 1, \ldots, r-1$$

This yields a start and end knot for every point sequence segment.

The PPN Architecture The PPN, as stated earlier, takes in an input of segments p, which can be written as $p = (x_0, \ldots, x_{l-1}, y_0, \ldots, y_{l-1})$. The parameter domain is defined as $u_0 = t_0 = 0 and u_n = t_{l-1} = 1$. For a sequence of points p, a parameter vector $t = (t_i)_i$, is defined as $t_i = t_{i-1} + \Delta_{i-1}$. The task of the PPN is to predict missing values $\Delta = (\Delta_0, \ldots, \Delta_{l-2})$ with

$$\Delta_s ubi > 0, i = 0, \ldots, l-2$$

such that $t_0 < t_1$ and $t_{l-2} < t_{l-1}$. We apply a multilayer perceptron (MLP) to the input data p, yielding as output a distribution for parametrization $\Delta^{mlp} = (\Delta^{mlp}_0, \ldots, \Delta^{mlp}_{l-2})$ of size $l-1$.

The PPN further contains additional layers PP1, PP2, and PP3, which will be discussed next.

PP1. Accumulation and Rescaling The output Δ^{mlp} is used to compute a parameter vector t^{mlp} with $t^{mlp}_0 = 0$ and

$$t_i^{mlp} = \sum_{j=0}^{i-1} \Delta_j^{mlp}, i = 1, \ldots, l-1$$

Since t_{l-1}^{mlp} is usually not 1, rescaling t^{mlp} yields the final parameter vector t with

$$t_i = \frac{t_i^{mlp}}{max(t^{mlp})}$$

The MLP layer in the PPN uses a softplus activation function defined to be:

$$f(x) = ln(1 + e^x)$$

PP2. Approximation B-spline curve approximation is included directly into the PPN as a network layer. The input points p and their parameters t are used for an approximation with knot vector $u = (0, 0, 0, 0, 1, 1, 1, 1)$ for $k = 3$. The approximation layer's output $p^{app} = (p_0^{app}, \ldots, p_{l-1}^{app})$ is the approximating B-spline curve evaluated at t.

PP3. Euclidean Loss A loss function, which is a Euclidean loss function, is to be used in the PPN. The Euclidean Loss function is defined as

$$\frac{1}{l} \sum_{i=0}^{l-1} ||p_i - p_i^{app}||_2 \tag{5.2.1}$$

Parametrization Refinement

In some cases, the approximated parametrization of a p^s may have errors. The approximation error of a p^s is computed by using the Hausdorff distance to the input data p.

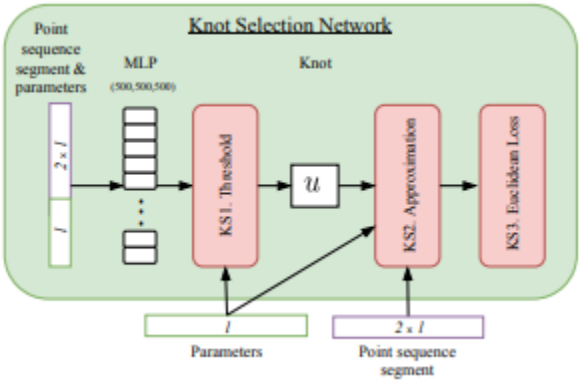

Figure 5.2: The network architecture of the Knot Selection Network (PPN)

.

The segments p^s that have a large approximation error are computed a new knot using the KSN. For $\bar{p}^s$ and $\bar{t}^s$, the KSN generates a new estimated knot $\bar{u}_s \in [0, 1]$. The new knot $\bar{u}_s$ is mapped to the actual knot value range $[u_{s-1}, u_s]$ by

$$\tilde{u}_s = u_{s-1} + \bar{u}_s(u_s - u_{s-1})$$

Instead of $\tilde{u}_s$, the parameter value t_i closest to $\tilde{u}_s$ is inserted into u. u can be further refined until the desired curve approximation error threshold is satisfied.

The KSN Architecture This stage utilizes a KSN, as mentioned earlier. The KSN predicts a new knot u to the interval $(0, 1)$ for a given segment p and parameters t, of which were previously approximated by the PPN. This network uses an MLP, which transforms the input to a single output value u^{mlp}. The RELU function is used as the activation function for the MLP, except for the output layer where the Sigmoid function is used instead.

Similar to that of the PPN, the KSN has three additional layers: KS1, KS2, and KS3.

KS1. Threshold Layer The new knot u computed has to satisfy the following conditions: $u \in (0, 1)$, $t \cap [0, u] \neq \theta$, and $t \cap [u, 1] \neq \theta$. Satisfying these conditions will will require us to use a threshold layer which maps u^{mlp} to

$$u = \begin{cases} \epsilon & , \text{if } u^{mlp} \leq 0 \\ 1 - \epsilon & , \text{if } u^{mlp} \geq 1 \\ u^{mlp} & , \text{otherwise} \end{cases}$$

Introducing a small $\epsilon = 1e - 5$ makes sure that the knot multiplicity at the end knots stays equal to k.

KS2. Approximation Approximation in the KSN is generally similar to that of PPN. The only different is that of the knot vector. The knot vector in the KSN is defined to be $u = (0, 0, 0, 0, u, 1, 1, 1, 1)$. For backpropagation, the derivative of the B-Spline basis functions with respect to u is required.

KS3. Euclidean Loss The loss function for the KSN is the same as that of the PPN. See 5.2.1.

Network Training

The training of the PPN and KSN will be based from the work of Laube, P., et. al. [**?**]. The input size of the network will be $l = 100$.

The data set generated by the original authors consists of 150,000 curves. This data was synthesized from B-spline curves. Random control points c_i were generated using a normal distribution μ and variance δ to define cubic ($k = 3$) B-spline curves with $(k + 1)$-fold end knots and no interior knots. The y-coordinates are given the configuration: $\delta = 2$ and $\mu = 10$. For the x-coordinates, $\delta = 1$ and $\mu = 10$ are used for

the first control point. All consecutive points have μ increased by $\Delta\mu = 1$. Curves with self-intersections are discarded, because the sequential order of their sampled points is not unique, and point sequences are usually split into subsets at the self-intersections. Smaller δ for the x-coordinates of control points reduces the number of curves with self-intersections. To closely match the target input as much as possible, we also include curves that have been manually fitted to raster images. These curves can be obtained from vector images available online.

For each curve, l points $p = (p_0, \ldots, p_{l-1})$ are sampled. These curves then to have increasing x-coordinates from left to right. As such, index-flipped versions of the point sequences of the dataset are added, resulting in 300,000 point sequences. 20% of the sequences are used as test data in the training process.

The PPN is trained first since the KSN requires point parametrizations t, which is obtained from the PPN. After training, the PP2 and PP3 layers are discarded and PP1 becomes the output layer of the PPN. The parametric values t are computed for the training dataset by applying the PPN and train the KSN on the combined input. After training, KS2 and KS3 are discarded, with KS1 becoming the network output layer. The MLPs of the PPN and KSN will consist of three hidden layers with sizes (1000, 1000, 1000) and (500, 500, 500) respectively. Dropout is applied to the MLP layers. The network is trained using the Adam optimizer.

5.3 Region Colouring and Merging

Once the curves have been approximated, the vectorization of the region will be filled with the colour prominent in the raster version of the region. The regions will be plotted unto their locations in the original raster input. Once all the regions have

been plotted, they will be grouped into a single vectorization. This will now be the vectorization output of the raster image.

Chapter 6

Describing How You Validated Your Approach.

Chapter 7

Stating Your Results and Drawing Insights From Them.

Chapter 8

Summarizing Your Thesis and Drawing Your Conclusions.

Appendix A

What should be in the Appendix

What goes in the appendices? Any material which impedes the smooth development of your presentation, but which is important to justify the results of a thesis. Generally it is material that is of too nitty-gritty a level of detail for inclusion in the main body of the thesis, but which should be available for perusal by the examiners to convince them sufficiently. Examples include program listings, immense tables of data, lengthy mathematical proofs or derivations, etc.

Bibliography

[1] Quadratic Assignment Problem — NEOS.

[2] Massive flooding hits Cagayan Valley, Philippines, CARE and partners ready to respond, 2020.

[3] Ali Derakhshan Asl, Kuan Yew Wong, and Manoj Kumar Tiwari. Unequal-area stochastic facility layout problems: solutions using improved covariance matrix adaptation evolution strategy, particle swarm optimisation, and genetic algorithm. *International Journal of Production Research*, (August 2015):0–25, 2015.

[4] Alexandre C. Dimian, Costin S. Bildea, and Anton A. Kiss. Sustainability Analysis. *Computer Aided Chemical Engineering*, 35:679–702, 2014.

[5] Amine Drira, Henri Pierreval, and Sonia Hajri-Gabouj. Facility layout problems: A survey. *Annual Reviews in Control*, 31(2):255–267, 2007.

[6] Panagiotis M. Farmakis and Athanasios P. Chassiakos. Genetic algorithm optimization for dynamic construction site layout planning. *Organization, Technology and Management in Construction: an International Journal*, 10(1):1655–1664, 2018.

[7] Mohammad Javad Feizollahi and Hadi Feyzollahi. Robust quadratic assignment problem with budgeted uncertain flows. *Operations Research Perspectives*, 2:114–123, 2015.

[8] Laura Garcia-Hernandez, Antonio Arauzo-Azofra, Lorenzo Salas-Morera, Henri Pierreval, and Emilio Corchado. Facility layout design using a multi-objective interactive genetic algorithm to support the DM. *Expert systems (Print)*, 32(1):94–107, 2013.

[9] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, 1979.

[10] José Fernando Gonçalves and Mauricio G. C. Resende. A biased random-key genetic algorithm for the unequal area facility layout problem. 246:86–107, 2015.

[11] Hasan Hosseini-Nasab, Sepideh Fereidouni, Seyyed Mohammad Taghi Fatemi Ghomi, and Mohammad Bagher Fakhrzad. Classification of facility layout problems: a review study. *International Journal of Advanced Manufacturing Technology*, 94(1-4):957–977, 2018.

[12] J. Lewis. Housing construction in earthquake-prone places: Perspectives, priorities, and projections for development. *The Australian Journal of Emergency Management*, 18(2), 2003.

[13] Zhang Lin and Zhang Yingjie. Solving the Facility Layout Problem with Genetic Algorithm. *2019 IEEE 6th International Conference on Industrial Engineering and Applications, ICIEA 2019*, pages 164–168, 2019.

[14] Jingfa Liu, Huiyun Zhang, Kun He, and Shengyi Jiang. Multi-objective particle swarm optimization algorithm based on objective space division for the unequal-area facility layout problem. *Expert Systems with Applications*, 102:179–192, 2018.

[15] R. D. Meller and Y. A. Bozer. A new simulated annealing algorithm for the facility layout problem. *International Journal of Production Research*, 34(6):1675–1692, 1996.

[16] Maricar M. Navarro and Bryan B. Navarro. Evaluations of crossover and mutation probability of genetic algorithm in an optimal facility layout problem. *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 8-10 March:3312–3317, 2016.

[17] Jamie E. Padgett and Citlali Tapia. Sustainability of Natural Hazard Risk Mitigation: Life Cycle Analysis of Environmental Indicators for Bridge Infrastructure. *Journal of Infrastructure Systems*, 19(4):395–408, 2013.

[18] Yunfang Peng, Tian Zeng, Lingzhi Fan, Yajuan Han, and Beixin Xia. An Improved Genetic Algorithm Based Robust Approach for Stochastic Dynamic Facility Layout Problem. *Discrete Dynamics in Nature and Society*, 2018, 2018.

[19] K. Reid. 2013 Typhoon Haiyan: Facts, FAQs, and how to help.

[20] Arthur Richards and Jonathan How. Mixed-integer programming for control. *Proceedings of the American Control Conference*, 4:2676–2683, 2005.

[21] Kazi Shah Nawaz Ripon, Kyrre Glette, Kashif Nizam Khan, Mats Hovin, and Jim Torresen. Adaptive variable neighborhood search for solving multi-objective facility layout problems with unequal area facilities. *Swarm and Evolutionary Computation*, 8:1–12, 2013.

[22] Maghsud Solimanpur and Amir Jafari. Optimal solution for the two-dimensional facility layout problem using a branch-and-bound algorithm. *Computers and Industrial Engineering*, 55(3):606–619, 2008.

[23] United States Federal Emergency Management Agency. Planning for a Sustainable Future: The Link Between Hazard Mitigation and Livability. Technical report, 2000.

[24] WBDG Secure/Safe Committee. Natural Hazards Mitigation.

[25] Laurence A. Wolsey. Mixed Integer Programming. In *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc., Hoboken, NJ, USA, sep 2008.