Nicolas Grobelny

Optimal Camera Placement

CO3201 Interim Report

Contents

[Aims and Objectives 2](#_Toc88129434)

[Aims 2](#_Toc88129435)

[Objectives 3](#_Toc88129436)

[Planning and Timescales 4](#_Toc88129437)

[Methodology 4](#_Toc88129438)

[Scrum 4](#_Toc88129439)

[Why Scrum 4](#_Toc88129440)

[Timescales 5](#_Toc88129441)

[Semester One 5](#_Toc88129442)

[Semester Two 6](#_Toc88129443)

[Justification 6](#_Toc88129444)

[Description of the Prototype 7](#_Toc88129445)

[Technologies Used 7](#_Toc88129446)

# Aims and Objectives

## Aims

This project aims to determine the best solution/solutions to the Optimal Camera Placement Problem by conducting a technical review of the field, attempting to determine whether present algorithms are valid and can produce reproducible result.

Additionally, a web application with an accessible interface would be developed allowing users to find solutions using results of the technical review. The program should allow them to digitalise their surroundings, determine potential camera placements and get an optimal solution based on maximisation parameters, and constraints.

In this way, the research not only improves the field by comparing existing results but also allows users, such as security experts, to place their cameras with greater accuracy whilst using less time. So far, the most common method to place cameras involves previous knowledge and the “rule of thumb”, making it inefficient.

## Objectives

The main objectives required to achieve the project are:

1. Stage One – Problem Research:
   * Conduct research into the Optimal Camera Placement Problem, maximising number of potentially valid algorithms that could be used to solve that problem.
2. Stage Two – Implementation Research:
   * Determine the best Frontend technology for this application, focusing on accessibility and simplicity (HTML5/CSS4/JavaScript).
   * Determine the best Backend technology, focusing on speed and single page applications. (Python-Flask)
   * Determine the best connecting technology, focusing on simplicity and performance. (Web Sockets, more specifically Sockets.io)
3. Stage Three – Environment Implementation:
   * Implement a drawing environment on the client site using JavaScript. The environment should resemble a simple drawing application such as Paint.
   * Implement a single page Backend allowing user to fetch web page and conduct updates to it using Sockets.
   * Implement the socket connection so that user can exchange information with the server in an efficient and secure way.
4. Stage Four – Algorithm Implementation:
   1. Pick a random algorithm from the algorithms researched during stage one.
   2. Determine its validity based on claimed time complexity and difficulty to implement. If the algorithm does not meet criteria, disregard further steps.
   3. Implement and test the algorithm.
   4. Go to step “a” and continue till all algorithms have been considered.
5. Stage Five – Algorithm Comparison:
   1. Get sample data for the Optimal Camera Placement using sources such as GECCO OCP.
   2. Run each algorithm using sample data and collect results.
   3. Determine the best algorithm if possible. Otherwise identify the situation in which the algorithm has the best performance.

Each stage must be completed in the given order. Inside each stage, unordered lists can be finished in any order whereas subtasks from ordered sections need to be executed one after the other. This gives a clear structure and prevents from implementing features in a way that later forces refactoring.

# Planning and Timescales

## Methodology

### Scrum

Scrum is a project management framework based around Agile philosophy. It is based around a sprint – A short, time-boxed period during which a Scrum team attempts to complete a set amount of work. Sprints offer greater flexibility as that project requirements can be changed at the start of every sprint without much of an impact on project deadline.

Another advantage of Scrum is its deliverability – At the end of each sprint, the stakeholder would be able to see finished tasks and determine whether they are of acceptable quality.

### Why Scrum

The Optimal Camera Placement Application appears to be the first project of its type, it is near impossible to visualise all features that should be part of it. Currently, a list of required features tends to evolve naturally, be it from client’s feedback or from inaccessibility of the application itself. For example, an ability to display coordinates of the grid space using both grid space and real-world dimensions was not a planned feature but testing of the application revealed that it is near impossible to draw accurate buildings without it.

As such, I required a framework that can quickly react to changes and can be easily prototyped for client to see. Scrum performs well in this situation as its short sprint window (Two Weeks in this case) allows me to prototype requested features, show them to user and change them if necessary.

## Timescales

### Semester One

#### Done

Stage One – Problem Research **(11th October – 20th October):**

* Conduct research into the Optimal Camera Placement Problem, maximising number of potentially valid algorithms that could be used to solve that problem.

Stage Two – Implementation Research **(11th October – 20th October):**

* Determine the best Frontend technology for this application, focusing on accessibility and simplicity (HTML5/CSS4/JavaScript).
* Determine the best Backend technology, focusing on speed and single page applications. (Python-Flask)
* Determine the best connecting technology, focusing on simplicity and performance. (Web Sockets, more specifically Sockets.io)

#### Started

Stage Three – Environment Implementation **(20th October – 1st December):**

* Implement a drawing environment on the client site using JavaScript. The environment should resemble a simple drawing application such as Paint.
* Implement a single page Backend allowing user to fetch web page and conduct updates to it using Sockets. **(Done)**
* Implement the socket connection so that user can exchange information with the server in an efficient and secure way. **(Done)**

Stage Four – Algorithm Implementation **(3rd November – 21st February)**:

Each individual algorithm should take approximately a week to implement, with enough time left for adjustments.

1. Pick a random algorithm from the algorithms researched during stage one.
2. Determine its validity based on claimed time complexity and difficulty to implement. If the algorithm does not meet criteria, disregard further steps.
3. Implement and test the algorithm.
4. Go to step “a” and continue till all algorithms have been considered. **(Done 1 out of 10)**

### Semester Two

#### Not Started

Stage Five – Algorithm Comparison (21st February – 7th April):

1. Get sample data for the Optimal Camera Placement using sources such as GECCO OCP.
2. Run each algorithm using sample data and collect results.
3. Determine the best algorithm if possible. Otherwise identify the situation in which the algorithm has the best performance.

Stage Six – Testing, Improvements and User Suggestions (7th April – 5th May)

* Ensure that test coverage is at least at 75 %, excluding network functions.
* Implement additional stakeholder suggestions (Such as optional feature or quality of life improvements).
* Add accessibility features and ensure compliance with web standard.

### Justification

Whilst adding exact dates might allow to better track whether the project is progressing at a right rate, the nature of Scrum sprint prevents me from specifying delivery dates within each epic.

Doing so would compromise quality as features will be rushed to meet the expected date. Another option is to ignore exact dates until the task is done, removing the reason for those times to be there in the first place.

Each epic has been chosen in a way to overestimate amount of time needed to implement it. At the current moment the project is progressing faster than expected, allowing to assign a significantly larger duration to Algorithm Implementation and Comparison. In case that time proves to be insufficient to deliver the project, the Stage Six is optional and can be cut out permanently to ensure quality of the deliverable.

# Algorithms and Data Structures

The Optimal Camera Placement Application (OCPA) is implemented as a web application. The flow of information in the application should follow the diagram below (Figure 1):

The Diagram consists of three main parts:

* Client Side
* Connection
* Server Side

Rather than using a MVC model, this application would be based around SPA (Single Page Application) model. Doing so greatly decreases load on the server as the user only sends and receives updates to currently presented content rather than a whole website. The client side is supposed to be responsible for majority of operations, using JavaScript to create a drawing environment that reacts to user’s inputs.

The SPA uses web sockets to communicate, allowing for constant, uninterrupted, and most importantly efficient exchange of information between the user and the server. Once user finishes with designing environment to suit the real-life situation, they can then send the required data to the server to receive optimal camera positions.

The actual server side would be based around Python web framework Flask. As the application requires no security features, except potentially a simple login system, the actual data handlers do not require any complicated features. As such, Flask’s simplicity allows me to quickly implement working data handling and move on to more important features such as algorithms.

Algorithm section is the core of this project. It would consist of approximately ten algorithms, written (where possible) through the usage of C-based Python libraries such as NumPy. This approach ensures high performance whilst retaining simplicity of Python code. As such, the project can be completed much quicker than equivalent C application.

The actual algorithms will be based on the research conducted on the OCPP. The goal is to implement most popular/viable algorithms used to solve this problem and compare them to each other in a variety of situations. Doing so should allow me to review these algorithms, allowing to determine which algorithms have the best performance and which ones should be avoided.

## Client Side

Starting with the client side, the application will resemble a drawing software not unlike Paint. To implement that, I will be using combination of HTML and CSS for the interface following the Material Design style to create an accessible website. JavaScript canvas in combination with mouse events will be used to allow user to draw.

As the canvas can cause performance issues, especially when frequently redrawing objects, all input events would be registered and applied first, then the canvas would actually update. This process should prevent any input lag as well as make the implementation simpler (As the project would guarantee that the updated point would eventually be drawn on the board).

JavaScript classes will be used to create an object-oriented environment that can be updated without connection to the server. This approach of limited communication has two advantages. First one is limited load on the server, as it does not need to handle constant requests from possibly hundreds of users at the time. Second advantage is ability to operate in areas with bad connection. The data is stored locally and no changes to it are made by the server until server receives the whole data. As such, if the user is disconnected from the server at any point, they can just reconnect at a later point without data loss.

## Connection between server and user

The connection between server and user consists of two parts. Despite stating beforehand that the application does not use the MVC model, there still needs to be a simple Controller responsible for serving the main page. Fortunately, because the project does not use models, the controller only needs to serve the client-side with a view. Whilst Flask only allows to handle one request at a given point, this is the only operation that is a single, synchronous process. As such, it is unlikely to become a bottleneck for the performance, especially because more concurrent workers can be added.

The second part of the system is a Web Socket JavaScript library called Socket.io. The Web Socket has been chosen in this case as MVC and AJAX were not suitable due to reasons below:

MVC – Model-View-Controller requires a large number of conversions and data transfers, especially when transferring model data to and from client. Whilst implementing this framework type would drastically simplify client-side code, the performance drawback is too great for this project.

AJAX – Asynchronous JavaScript and XML works similarly to the Web Socket technology in a way that a user can send and receive data after the web page has loaded. The main difference is that whilst AJAX is more secure, Web Socket has a significantly better performance as it does not need to establish a connection with server every time it sends data. As such, it is a better choice in the situation where security is not a concern.

The Web Socket has been selected due to its advantages over other candidates. Whilst it is possible that there are other technologies out there that do outperform Web Sockets, the time needed to find them would outweigh potential benefits.

## Server-Side Algorithms

When user requests for new cameras to be placed, the client-side socket would send a request to the server containing user’s request and the Environment in the JSON form.

Upon arriving, the Python side would generate an Environment using NumPy data types (For efficiency) and create a Solver class.

The Solver class aims to contain all of the camera solving algorithms within it. This allows a development of helper functions, such as one for generating all coordinates between two grid points for visibility checker (Bresenham's line algorithm), to reduce code redundancy.

This approach should be viable for as long as the algorithms stay relatively simple. If the actual class starts to become buggy and cumbersome, each algorithm can easily be converted into separate classes (Each inheriting form the parent with all of the helper functions) and adapt execution using Polymorphism. Whether this decision is necessary will be determined once several algorithms are implemented, but the project is developed with that in mind.

These Backend algorithms will be based around NumPy (C library in Python). This is done to ensure performance as C is approximately forty-five times faster than regular python code. Whilst other libraries such as TensorFlow or SciPy are likely to be used in Computational Intelligence algorithms, the NumPy will be the core of the project, with all data types and calculations being done in that library (Where possible).

Once the algorithm executes correctly, the program will return camera results back to the user using a Web Socket by targeting a unique User Id. The User Id is generated automatically once user connects to the socket.

# Description of the Prototype/Current Development

Whilst previously I have discussed the technologies intended to be used within prototype, this section would focus on the current progress of the project as well as include feature list for each application module.

## Application Appearance and Functionality

At the time of writing this report, the application looks like in the figure below:

Table

Description automatically generated with medium confidence

Whilst the interface can be considered as ugly, it is functional allowing user to adjust size and draw basic schematics:

A picture containing graphical user interface

Description automatically generated

As well as sample the space automatically using the sampling slider on the left side:

A picture containing icon

Description automatically generated

Or even effectively place cameras:

A picture containing treemap chart

Description automatically generated

Whilst not perfect, the program is achieving its purpose. The user can draw a basic schematic, sample it, and get semi-optimal camera placements. The project from now onwards would add onto the framework, increasing number of options and quality of life features, as well as polishing the user experience by improving interface and granting them more tools.

## Features

This section focuses on all features for this application. It is by no means a completed list as features would get added and subtracted based on development, testing and client’s feedback. Therefore, this list should be used as guidelines where each of the features will be at least considered but might not end up in the final version of the project.

For simplicity word **User** will be used throughout these features/user stories. In this context User means a security company employee responsible for determining camera positions.

This project is also meant to serve as a technological survey of the Optimal Camera Placement Problem. As such, it also includes user stories aimed at **Scientists** from that field.

### Done

#### Front-End

- As a User, I want to be able to draw the examined building, so that I can receive optimal camera positions for my situation.

- As a User, I want to be able to place obstacles and other line of sight blockers, so that I can receive a realistic coverage of the building.

- As a User, I want to be able to automatically sample the drawn space, so that I can spend more time on determining viable camera positions or drawing a more accurate layout.

- As a User, I want to be able to manually sample the drawn space, so that I can better protect crucial areas of the building.

- As a User, I want to be able to resize the environment, so that I can make plans for all kinds of structures, including large outdoor areas.

#### Connection

- As a User, I want to be able to operate as much of the program offline, so that I can operate in areas without internet such as forests.

- As a User, I want to be able to easily reconnect to the program without losing data, so that I can operate even if the connection is temporarily disconnected.

- As a User, I want the program to transfer as little data as possible, so that I can operate in areas with poor/limited internet.

#### Back-End

- As a User, I want to be able to get optimal camera positions, so that I can secure the examined area.

- As a User, I want to be able to get camera positions fast, so that I can easily experiment with different parameters to suit my needs.

### In Development

#### Front-End

- As a User, I want to have a clear, accessible interface, so that I do not require extra training to know how to use the program.

- As a User, I want to be able to tell where each camera is looking and what samples it sees, so that I can validate results of the program.

#### Connection

- As a User, I want to know the status of the connection, so that I can know whether program is operational.

#### Back-End

- As a User, I want to be able to choose from variety of algorithms, so that I can get the most optimal result for my situation.

### Backlog/TODO

#### Front-End

- As a User, I want to have additional drawing tools such as Fill or bigger brush, so that I can draw the building plan more effectively.

- As a User, I want to be able to insert real-life dimensions, so I can know each camera position without needing to do conversions.

- As a User, I want to have an option to change camera parameters, such as cost, distance, or number of them, so that I can maintain constraints imposed onto me by the client.

- As a Scientist, I want to be able to insert environment data from the file, so that I can test algorithms without needing to redraw the environment.

#### Back-End

- As a User, I want to be able to have a way of determining the best algorithm for me, so that I can choose one without needing to know how they work.

- As a User, I only want to have a choice of optimal algorithms, so that I don’t waste time and resources using inefficient solutions.

- As a Scientist, I want to be able to see information about each solving iteration (Such as average score), so that I can compare algorithms easily.

### Optional/Nice to Have

#### Front-End

- As a User, I want to be able to save data to a file and share it with other people using the same program, so that I can easily exchange plans between people without losing data.

- As a User, I want to be able to use the program on a tablet or similar mobile device, so that I can use this program in any situation.

#### Back-End

- As a User, I want to have access to more algorithms (More than ten), so that I can achieve even better efficiency when securing an area.