A Block Problem_Group 5

Christoph Geiger
Mengxi He
Alexandra Pittiglio
Harrison Hildebrandt

Contents:

Urban Context Introduction:

In early discussions regarding an appropriate urban context to apply this research towards, Barcelona stood out for its uniquely homogenous grid of superblocks. In a more heterogeneous city, such as Tokyo or London, the stark variation in building height, massing, and architectural style would make it difficult to limit the parametric variability allowed in a model - seemingly anything could be possible in cities such as these. However, Barcelona as an urban context offered existing controls and limited variables, allowing us to focus on parameters that fine-tuned existing conditions.

Using Barcelona as our context, we selected 9 buildings (each approximately 100m x 100m) in a 3x3 cluster, and created a parametric model for optimisation results.

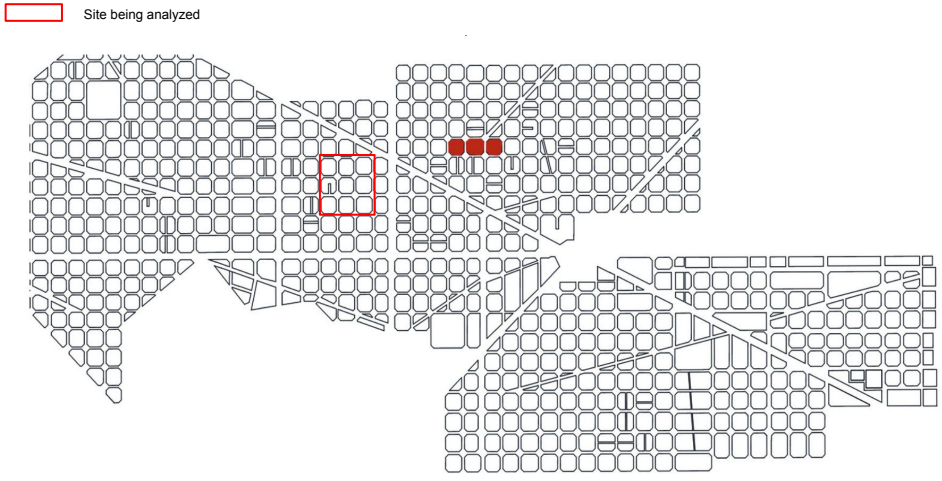


Fig 0_1. Our site

Fig 0_2. Map of Barcelona

Fig 0_3. Common building typology in Barcelona

Experimental Setup:

Each one of our 9 buildings on our selected site has 10 parameters on parametric sliders in a Grasshopper model. This results in a total of 90 variables influencing the resulting building masses.

The parameters are as follows:

(1) Building Morphology
  (a) Type 1
  (b) Type 2
  (c) Type 3
  (d) Type 4
(2) Courtyard Size → X Dimension
(3) Courtyard Size → Y Dimension
(4) Courtyard Positioning → X Location
(5) Courtyard Positioning → Y Location
(6) Courtyard Rotation
(7) Numbers of Storeys per building volume (up to 20 maximum)

With so much surface area for analysis, we experienced extremely long simulation times in our early trials. In order to reduce the analysis time, we only analyzed every 3rd floor in each building. All of the floors were still used in calculating the results but sensor grids were only created for the select floors. Additionally we lowered the sample depth of the analysis from 4096 to 64 and spaced the sensors every 10 meters. We conducted extensive benchmarking to ensure that these changes didn't significantly affect our results and we successfully reduced our simulation time from 15 minutes per iteration to ~1 minute.

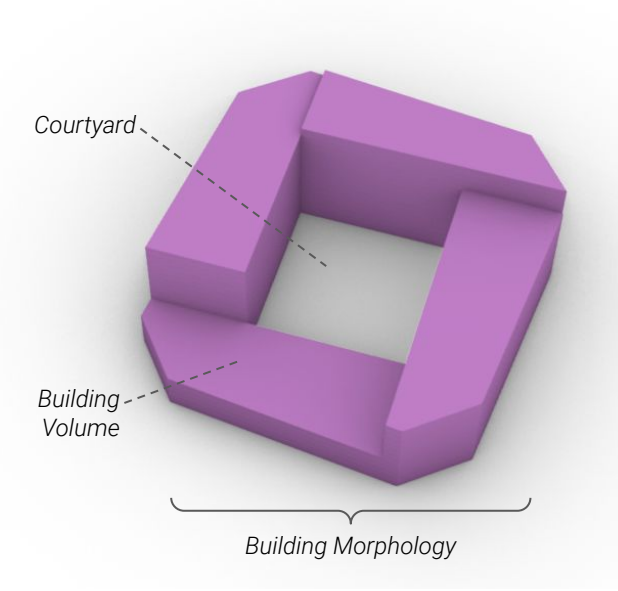


Fig 0_4. Building elements

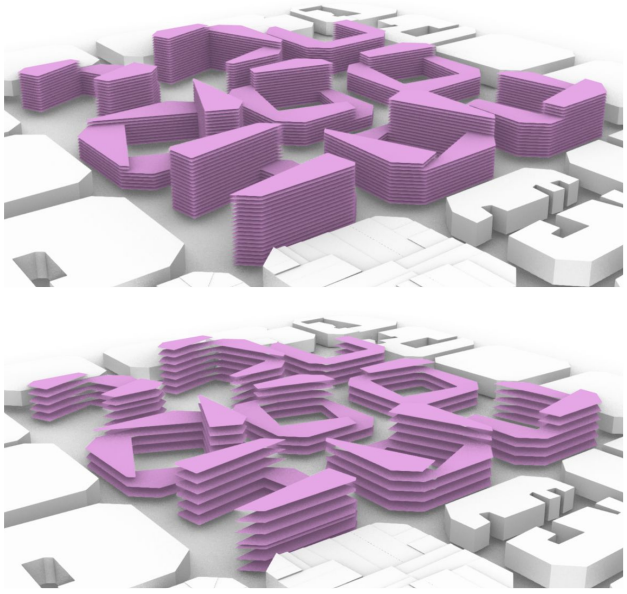


Fig 0_5. Reduction of analyzed surfaces

We used Climate Studio to conduct a solar analysis on our buildings, with the goal of maximizing Useful Daylight Illuminance (UDI) for all of the buildings. In subsequent experiments we compared Floor to Area Ratio (FAR) by simply summing the total floor area. We used the following materials for each of our building surfaces:

- Floor: LM83
- Wall: LM83
- Ceiling: LM83
- Windows
  - Glazing: Solexia Solarban 60
  - Treatment: SheerWeave 2410

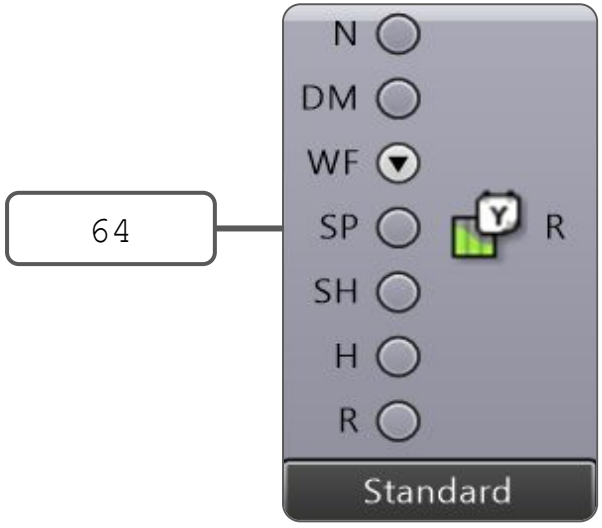


Fig 0_6. Climate Studio's solar analysis component

Reference: *Real Life Composite Barcelona Block*

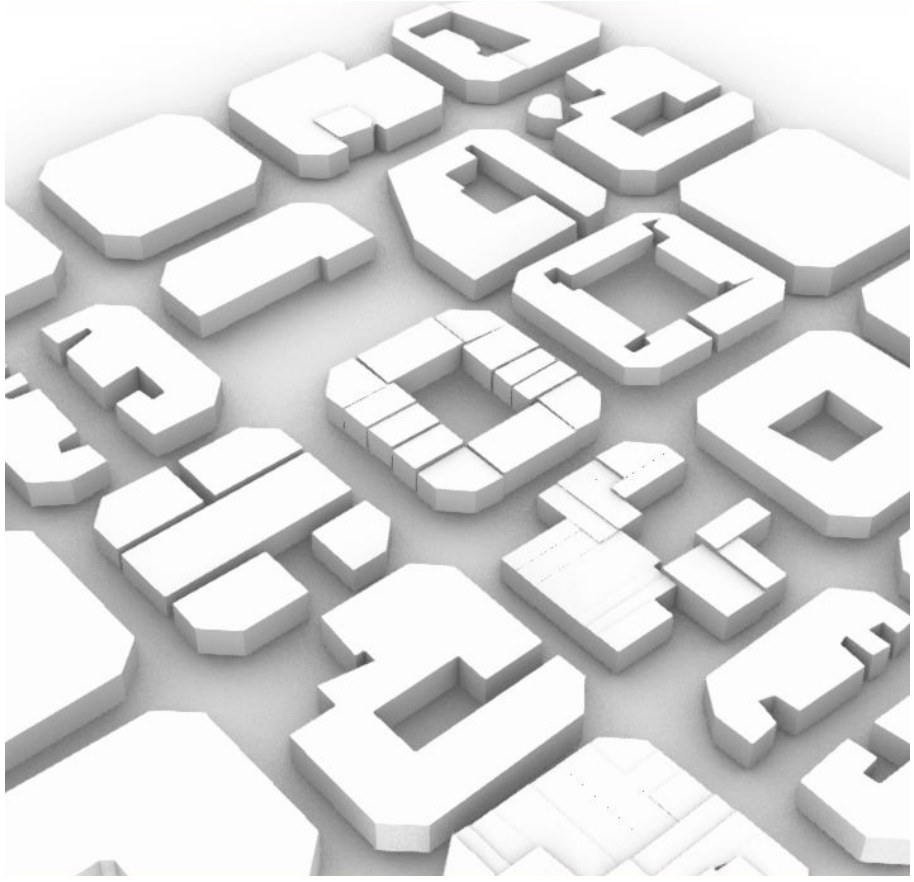


Fig 0_7. Real composite Barcelona superblock

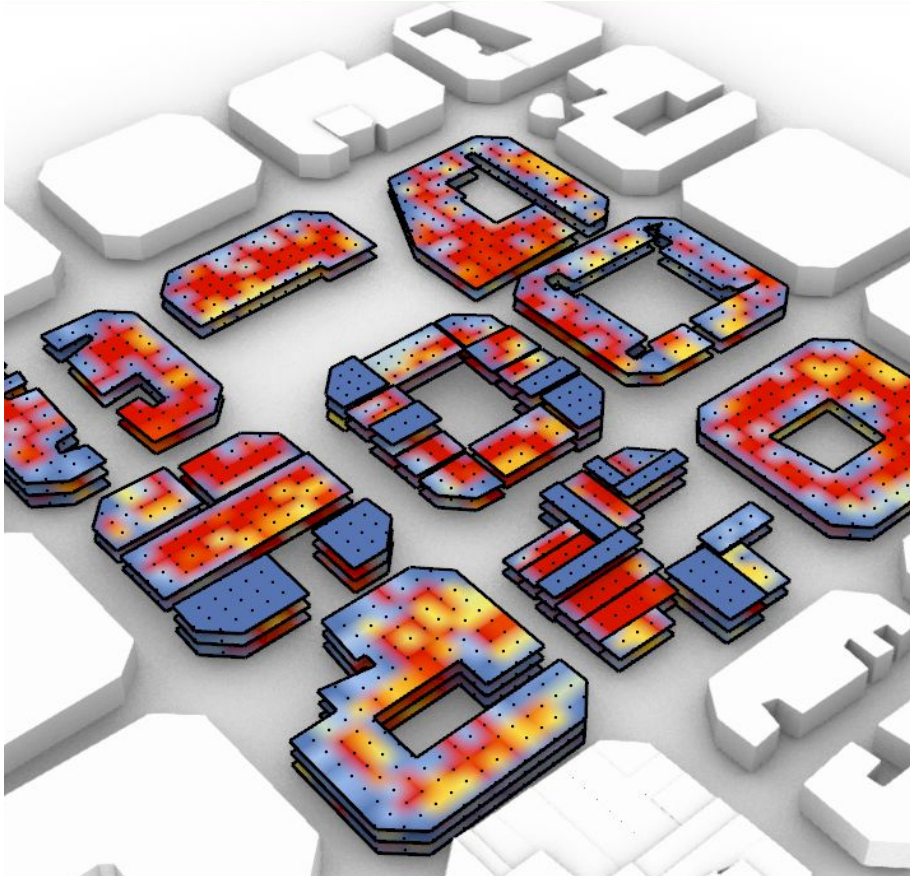


Fig 0_8. Solar analysis of Barcelona superblock

In order to ensure that our parametrically generated buildings yielded results that were at the very least comparable to existing buildings, we performed the analysis on a combination of real Barcelona buildings.

Results:

- FAR: 678518.51 m^2
- 0.2345

Description:

As previously stated, the main metric that we sought to maximize was Useful Daylight Illuminance. UDI is a solar analysis method that calculates the percentage of occupied time when a target range of illuminances at a point in a space is met by daylight. It is evaluated from 0 to 1.

We compared 4 Single Objective Optimization algorithms:

- RBFOpt
- CMA-ES
- NSGA 2
- DIRECT

Initially we ran 500 iterations 3 times, but we found that it was not sufficient for our results to fully converge so we elected to re-run RBFOpt with 1200 iterations. As you will see in the results, with no way of controlling floor area, optimizing UDI often created small buildings with minimal, awkward shaped floors. To combat this without introducing another objective, we decided to add a penalty function to our UDI calculation.

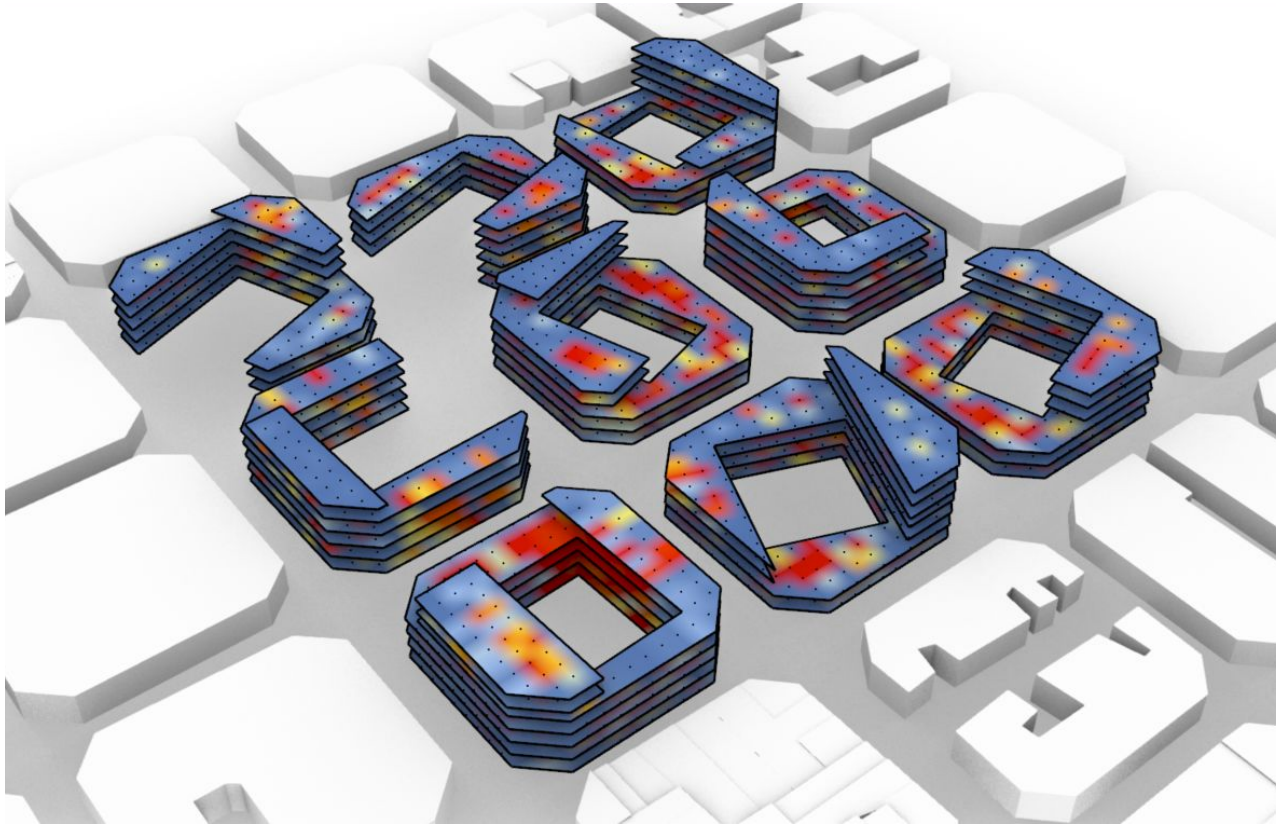


Fig 1_1. Example of UDI analysis on a parametric superblock

Results: *500 Iterations*

Generally, it is good practice to run roughly 50 iterations for every parameter you have. Unfortunately, if we were to follow this rule and run 4500 iterations our optimisation would take over 3 days to complete. For this reason we elected to perform 500 iterations which still took more than 8 hours per run. From our results it was evident that DIRECT performed the worst. With so many parameters, incrementally changing one at a time means that it would need a lot of iterations to ensure that it checks a variety of solutions. Both CMA-ES and CRS2 converged relatively quickly and were relatively robust, but ultimately didn't yield the maximum UDI. That distinction goes to RBFOpt which wasn't as fast or robust as the others but had the highest UDI at 0.4885.

By looking at *Fig 1_2.* it is hard to confidently say whether all of the algorithms converged on their best solutions, particularly when looking at RBFOpt which appears to still be climbing. 500 iterations is likely still too few for the amount of parameters we have.

Fig 1_13 - 1_15. show the morphological results of RBFOpt and, while it has the highest UDI values for each run, its buildings are small and awkward. This led us to the conclusion that simply using the UDI as the sole metric is not enough for creating realistic solutions. Further steps need to be added to the process, whether they be constraints on the model, penalties on calculating the UDI, or post processing with input from a designer.

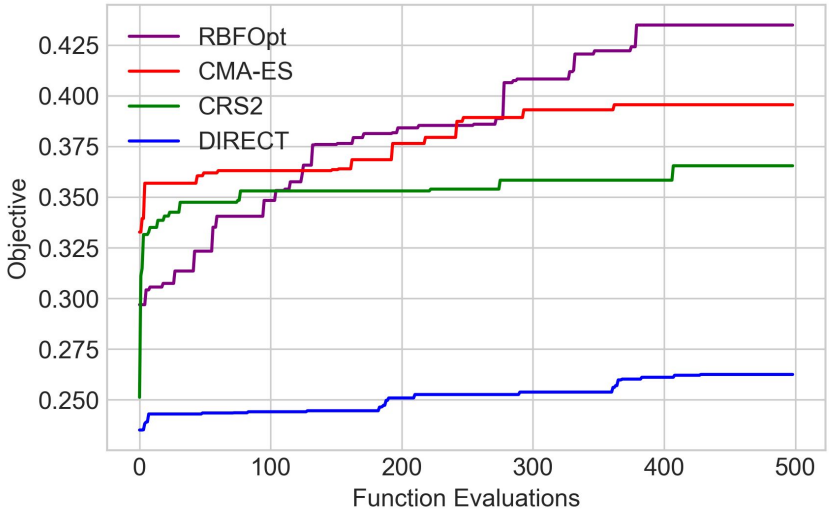


Fig 1_2. Convergence of SOO algorithms

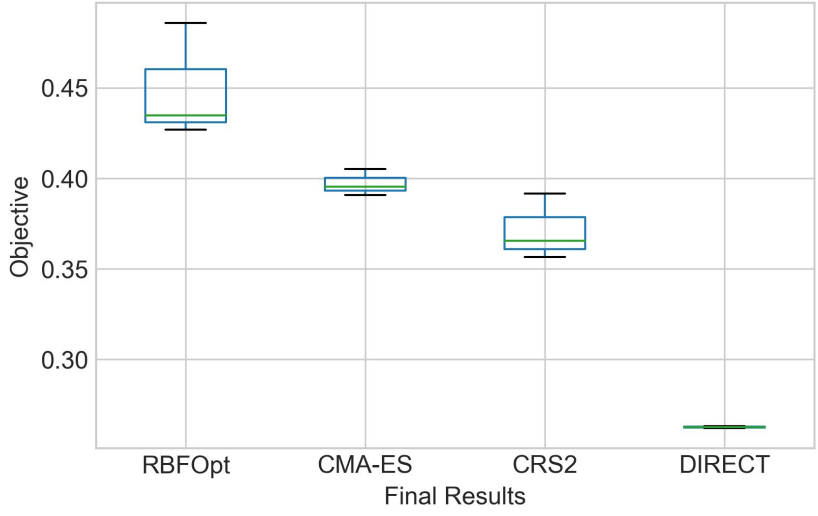
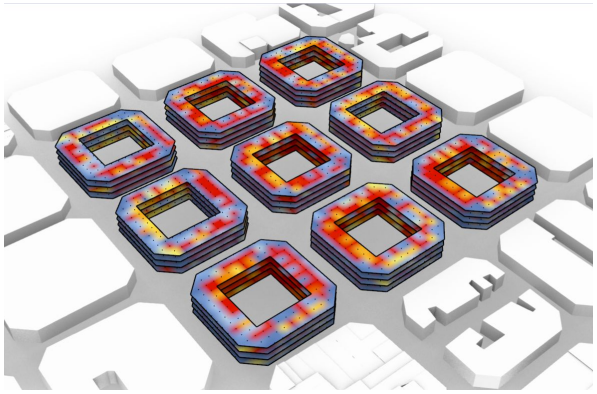


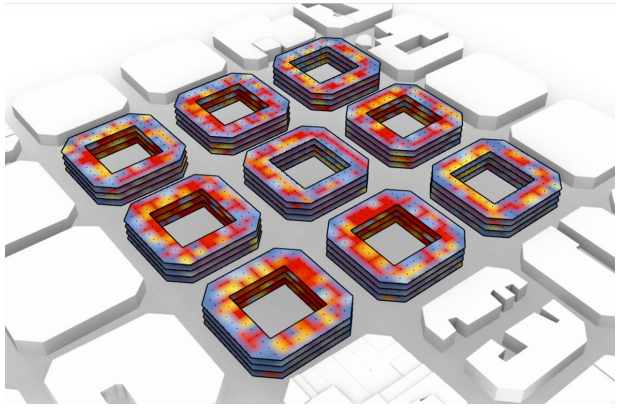
Fig 1_3. Robustness of SOO algorithms

Results: *Max UDI from 500 Iterations*



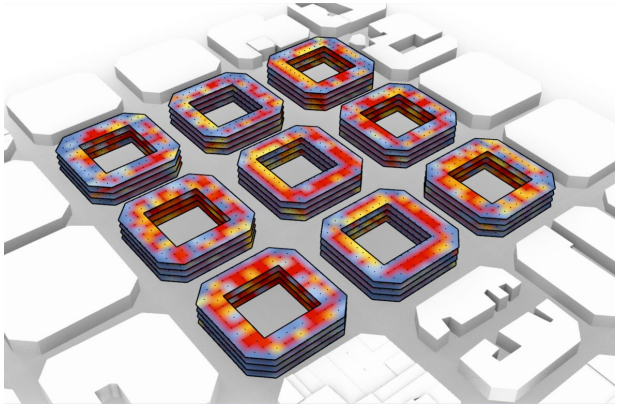
0.2631

Fig 1_4. DIRECT Run 1



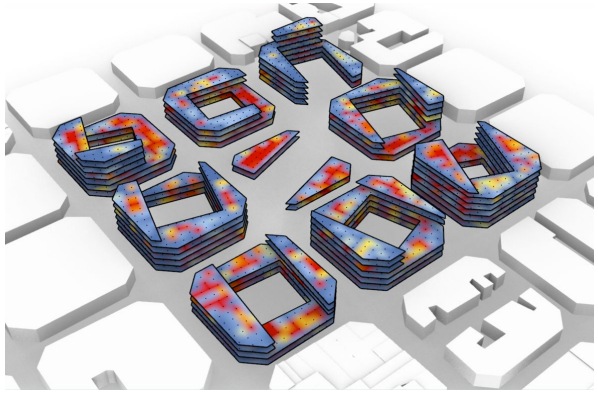
0.2625

Fig 1_5. DIRECT Run 2



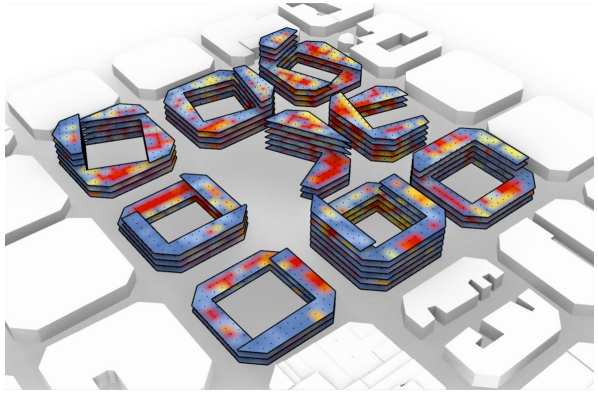
0.2621

Fig 1_6. DIRECT Run 3



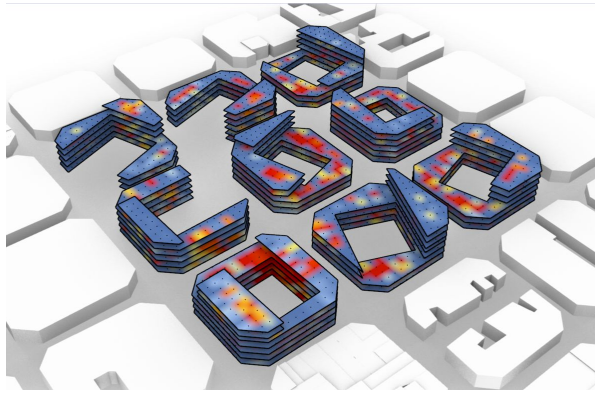
0.3654

Fig 1_7. CRS2 Run 1



0.3565

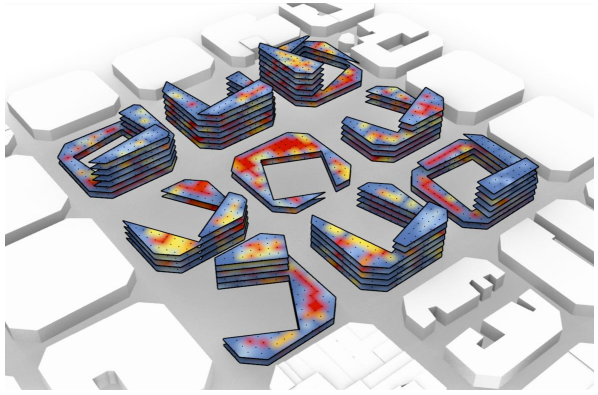
Fig 1_8. CRS2 Run 2



0.3915

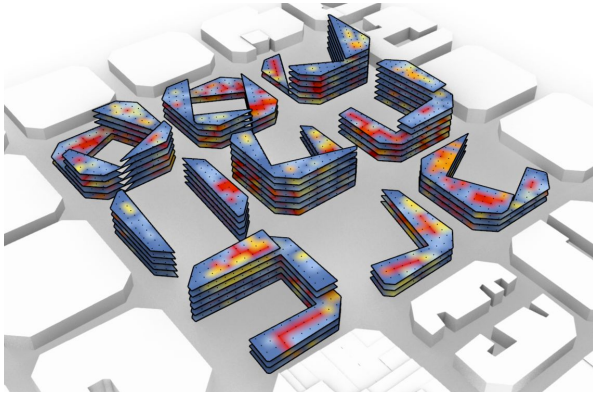
Fig 1_9. CRS2 Run 3

Results: *Max UDI from 500 Iterations*



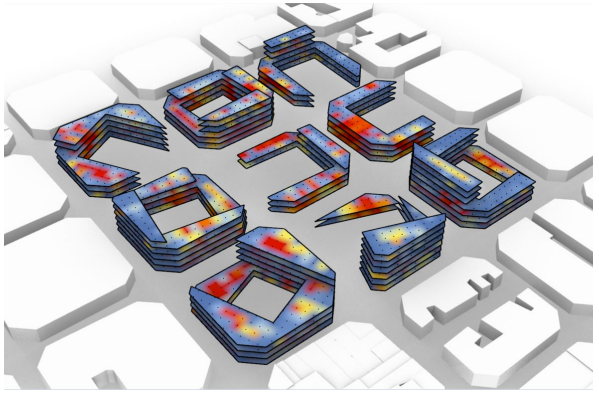
0.4051

Fig 1_10. CMA-ES Run 1



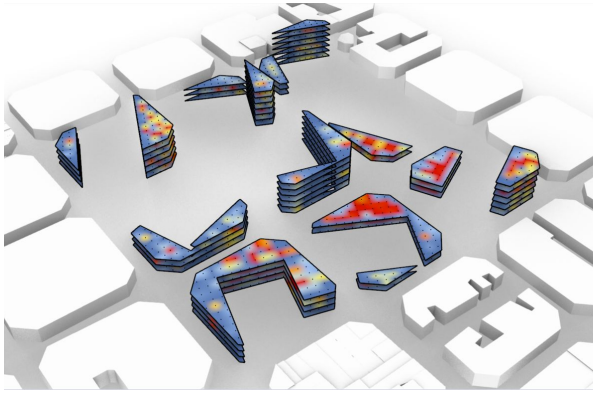
0.3955

Fig 1_11. CMA-ES Run 2



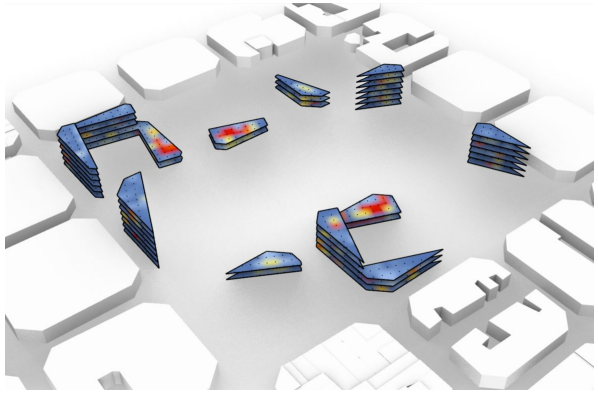
0.3908

Fig 1_12. CMA-ES Run 3



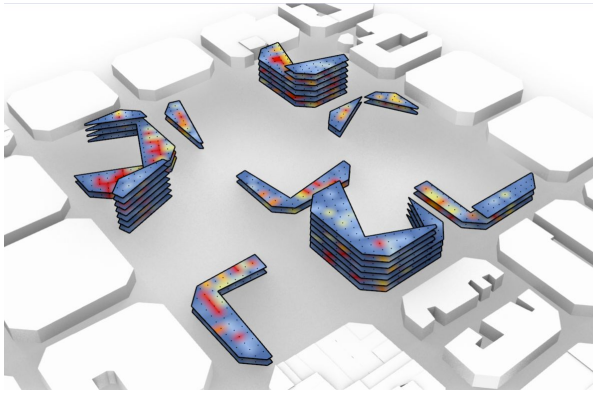
0.4270

Fig 1_13. RBFOpt Run 1



0.4885

Fig 1_14. RBFOpt Run 2



0.4348

Fig 1_15. RBFOpt Run 3

Results: *1200 Iterations*

Since we couldn't confidently say that the results from 500 iterations fully converged, especially for RBFOpt, we decided to repeat the experiment for RBFOpt with 1200 iterations. With the additional iterations the algorithm was able to find larger UDI values, peaking at 0.5068. Again, it suffers from small building footprints and awkward shapes (*Fig 1_16.*) reinforcing the need to add additional layers of logic to make the solutions more realistic. For reference the results of the 1200 iterations are overlaid on the convergence chart with the previous results using 500 iterations as can be seen in *Fig 1_17.*

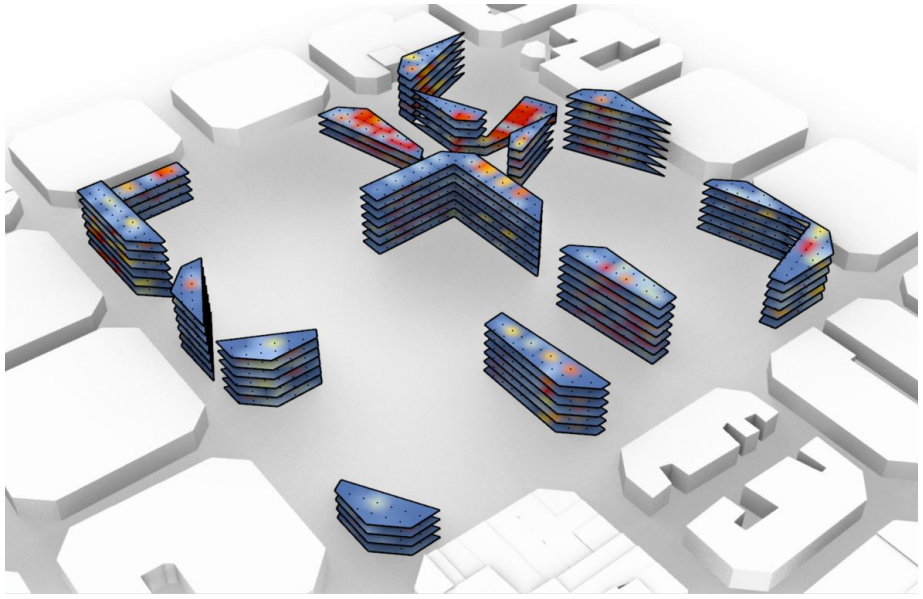


Fig 1_16. Resulting analysis of 1200 iterations, Run 1

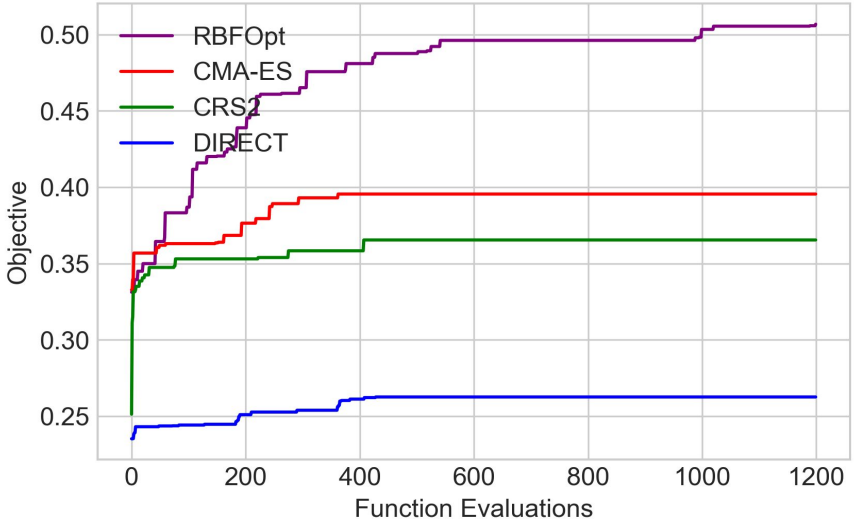


Fig 1_17. 1200 iterations of RBFOpt overlaid on previous results

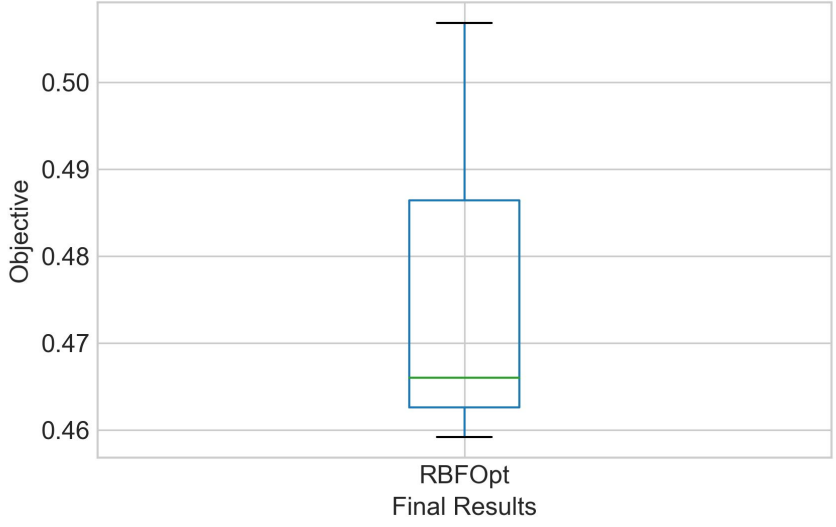


Fig 1_18. Robustness of 1200 iterations of RBFOpt

Description: *Penalty Function*

As previously stated, unchecked optimization of UDI often leads to small, unrealistic building morphologies. In order to yield more realistic solutions we opted to apply a penalty function to the UDI calculation. The penalty function was based on the total floor area of *each* building so that if the building area dipped below a certain threshold the UDI would be negatively impacted. Basically, for each building if the floor area was lower than a user-defined threshold (we used 4 maximized floors as a our metric) then a penalty factor was calculated by dividing the actual floor area by the threshold. Then all of the individual penalty factors were multiplied together to get the combined penalty.

Lastly, the combined penalty was multiplied by the UDI to create the adjusted or penalized UDI. As can be seen in *Fig 1_24 - 1_26*. the penalty function was effective in keeping the buildings more realistic, without significantly decreasing the overall UDI.

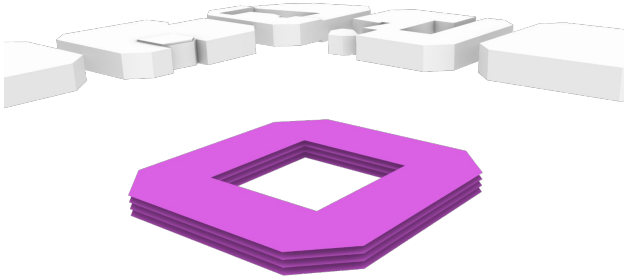


Fig 1_19. Threshold floor area

Pseudo-code:

```
for # of buildings:

    if floorArea >= thresholdFloorArea:
        penaltyFactor = 1
    else:
        penaltyFactor = floorArea / thresholdFloorArea

combinedPenalty = product of all penaltyFactors

adjustedUDI = UDI * combinedPenalty
```

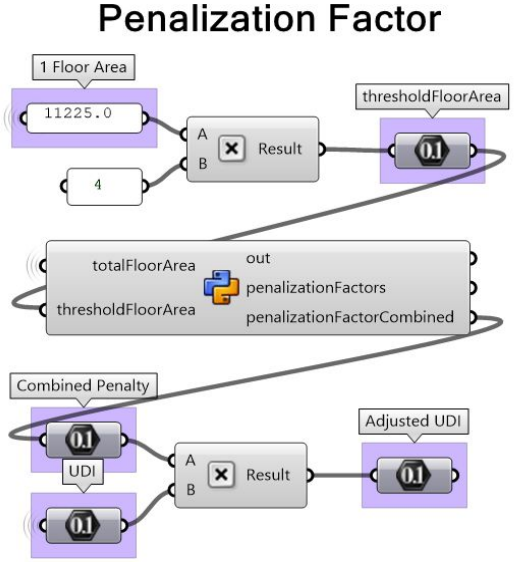


Fig 1_20. Calculation method

Results: *Max UDI from 1200 Iterations with and without Penalty*

0.5068

Fig 1_21. RBFOpt Run 1

0.4660

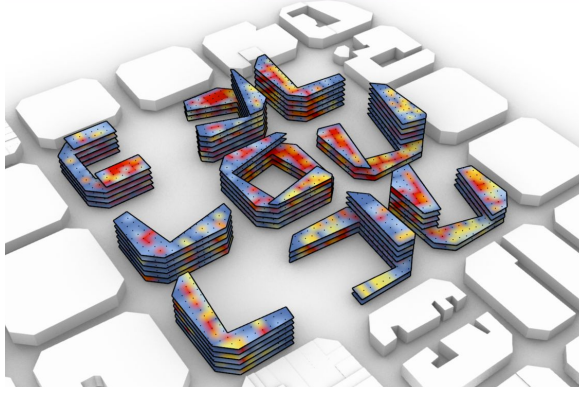
Fig 1_22. RBFOpt Run 2

0.4592 (900 iterations)

Fig 1_23. RBFOpt Run 3

0.4023

Fig 1_24. Penalized RBOpt Run 1



0.4298

Fig 1_25. Penalized RBOpt Run 2

0.4253

Fig 1_26. Penalized RBOpt Run 3

Conclusions:

Through our experiments in single objective optimization we determined that, in general, RBFOpt performed the best. Although it doesn't converge the fastest and it has a slightly higher variance in its results, it consistently yielded the highest UDI. That being said, it is crucial that results are analyzed by looking at more than just the final number they produce. While RBFOpt produced the highest UDI, the resulting buildings were unrealistic. It is important to couple algorithmic optimisation with good designer judgment. To help combat this, we included a penalty function that created more reasonable buildings without significantly decreasing UDI. A penalty function is a smart and effective way to improve the quality of optimisation without adding additional objectives. Despite increasing our iterations to 1200, we cannot confidently say that our results finished converging. This experiment could be improved by increasing the number of iterations and potentially performing a sensitivity analysis to determine whether having so many parameters is really necessary.

Iterations	**Algorithm**	Run			**Avg.**
		1	**2**	**3**	
500	DIRECT	0.2631	0.2625	0.2621	0.2626
	CRS2	0.3654	0.3565	0.3915	0.3711
	CMAES	0.4051	0.3955	0.3908	0.3971
	RBFOpt	0.4270	0.4885	0.4348	0.4501
1200	RBFOpt	0.5068	0.4660	0.4592*	0.4864
	RBFOpt (Penalized)	0.4023	0.4298	0.4253	0.4191

Fig 1_27. UDI results of SOO
**Only ran 900 iterations*

Description:

For our next experiment we introduced the objective of increasing the Floor to Area Ratio (FAR) of our buildings. We simplified this by summing the total floor area of all the buildings as our metric. In theory, FAR is a competing objective to UDI. As seen in our SOO results, UDI increases as the buildings get smaller. The 3 algorithms we compared were:

- RBFMOpt
- NSGA-2
- HypE

Results:

As shown in *Fig 2_1*. RBFMOpt clearly performed the best. Additionally, it had the second best robustness behind HypE.

Looking at the Pareto front in *Fig 2_3*. the relationship between UDI and FAR is inversely linear. As one might expect, the buildings that have the maximum amount of floors with the maximum area are at one extreme of the plot (high FAR) and small, scattered buildings are at the other (high UDI). The most interesting results are in the middle of the plot where you start to get more varied morphologies but that are still in the realm of being a realistic building. Though the relationship between the objectives was predictable, it is still a useful tool to illustrate that relationship and give you a refined list of optimized solutions.

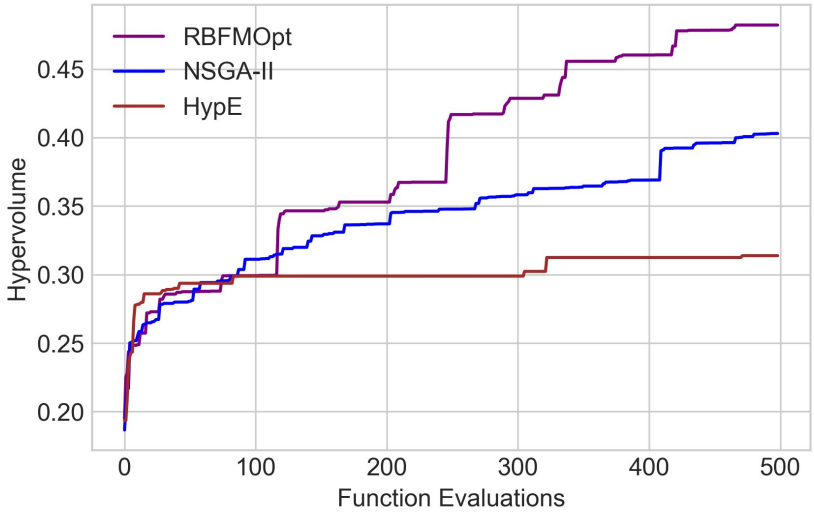


Fig 2_1. Convergence of hypervolume values

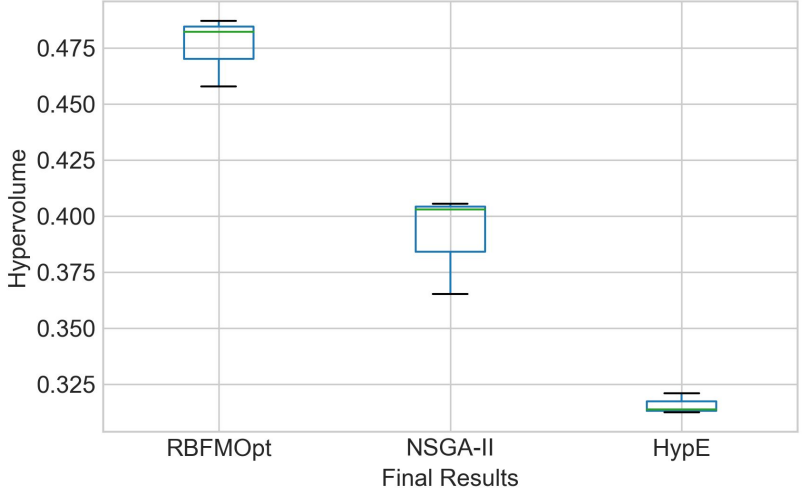


Fig 2_2. Robustness of hypervolume values

Results: *500 Iterations*

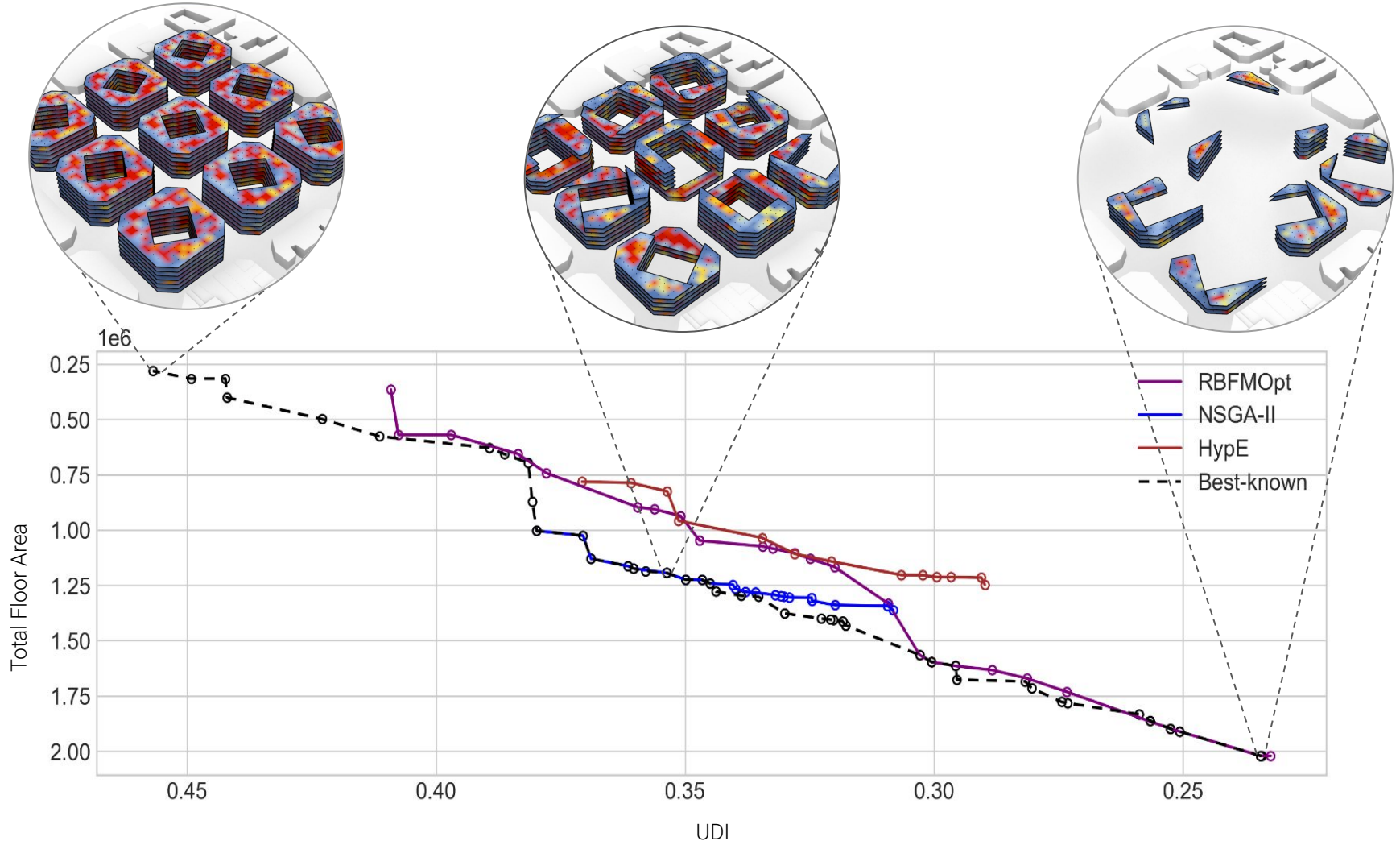


Fig 2_3. Pareto front

Results: *1200 Iterations*

In order to test whether more iterations would improve our hypervolume calculations, we increased the number of iterations to 1200 and repeated the tests (but with only 1 run each). The results of the 1200 iterations were basically the same as the 500. It slightly increased the hypervolume of the RBFMOpt, but it actually decreased the hypervolume results of NSGA-2 and Hype. That being said it isn't a robust comparison since we only did 1 run each for the 1200 iteration tests.

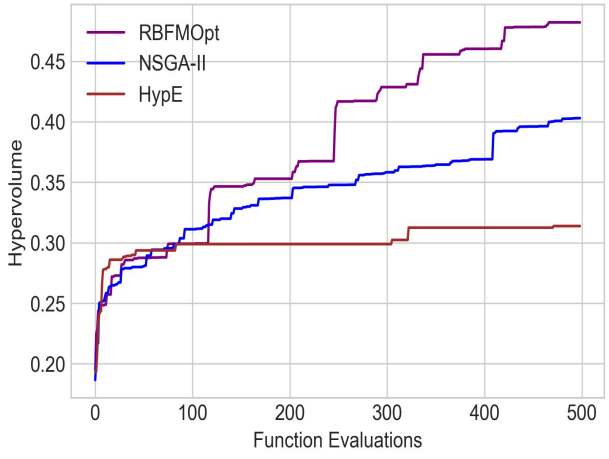


Fig 2_4. 500 iterations

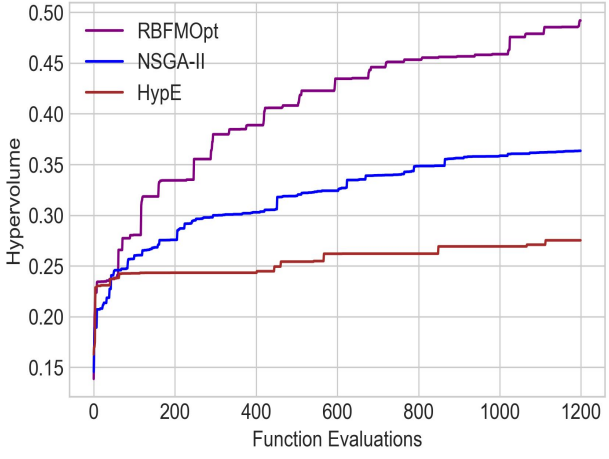


Fig 2_5. 1200 iterations

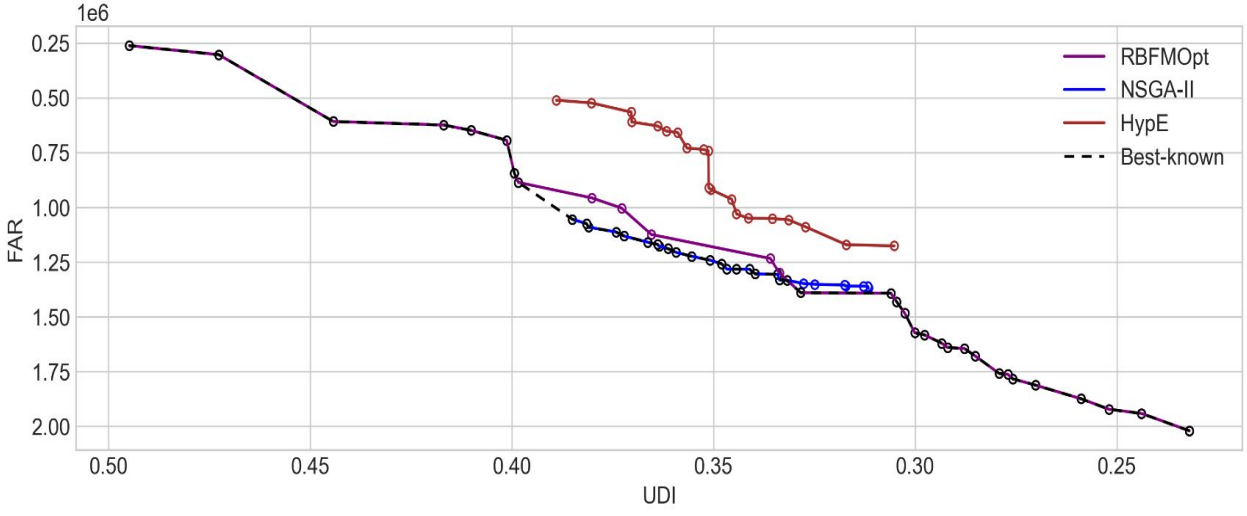


Fig 2_6. Pareto front after 1200 iterations

Conclusions:

The results of multi-objective optimization were relatively straightforward. We confirmed that UDI and FAR are inversely proportional. We found that by and large RBFMOpt was the best performing algorithm. It is unclear whether increasing the number of iterations used improves the results. This test could be improved by increasing the number of iterations even higher and by making sure that a sufficient number of runs are conducted. MOO is effective at providing a refined list of solutions that maximize the combine totals of UDI and FAR. The most interesting results are those in the middle of the Pareto front because they create the buildings that are varied in their morphologies but are still in the realm of possibility to be built in real life. That being said, as was shown in our SOO results, you can achieve effective results by smartly incorporating the second objective into the calculation of a single objective—making an MOO problem into an SOO problem.

Iterations	***Algorithm***	***# of Runs***	***Hypervolume***
500	HypE	3	0.314
	NSGA-2	3	0.403
	RBFMOpt	3	0.482
1200	HypE	1	0.275
	NSGA-2	1	0.363
	RBFMOpt	1	0.492

Fig 2_7. Hypervolume results of MOO

Description:

After the completion of the generative runs of the different optimization algorithms the next step was to get a better overview of the results.
Clustering and a graphic representation of the data could be helpful for the further selection of design options.
For that we decided to use K-Means clustering and afterwards show the clusters represented by meshes of the design options with colors assigned relating to the clusters they belong to.

The first attempt was to cluster the results laying on the pareto fronts of the three best performing algorithms, which were RBFMOpt, NSGA-2 and HypE.
Since for each of the algorithms all results on the pareto front are equivalent from an optimization standpoint, a spatial and visual representation of the results each labeled with the two objectives used for optimization.

After using principal component analysis for dimensionality reduction we first clustered the results of each of the three optimisation results by their objectives, as can be seen exemplary in (*Fig 3_1*.).
With only the two objectives used as features, the results of the clustering could be easily arranged on a two dimensional field using the feature values as axes.
This way of visual representation is chosen to simplify the design option selection by still providing the relevant information to the designer.

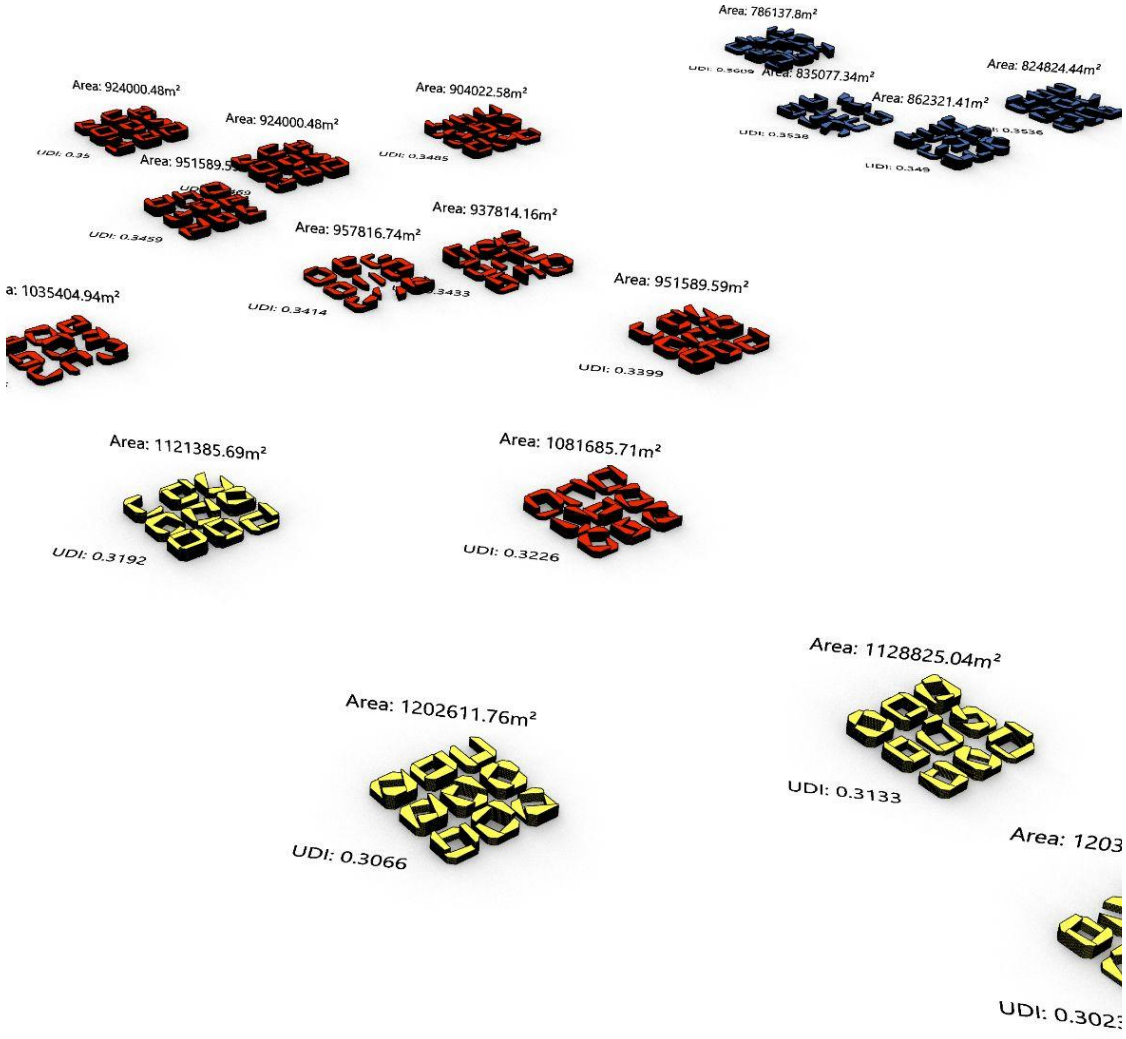


Fig 3_1. Partial perspective - results at the pareto front of RBFMOpt, NSGA-2 and HypE *clustered by objectives*

Results:

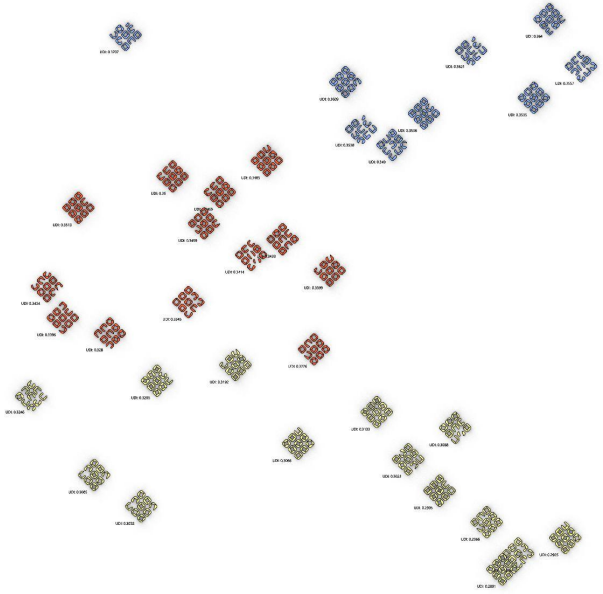


Fig 3_2. Top view - K-Means Clustering - HypE pareto front

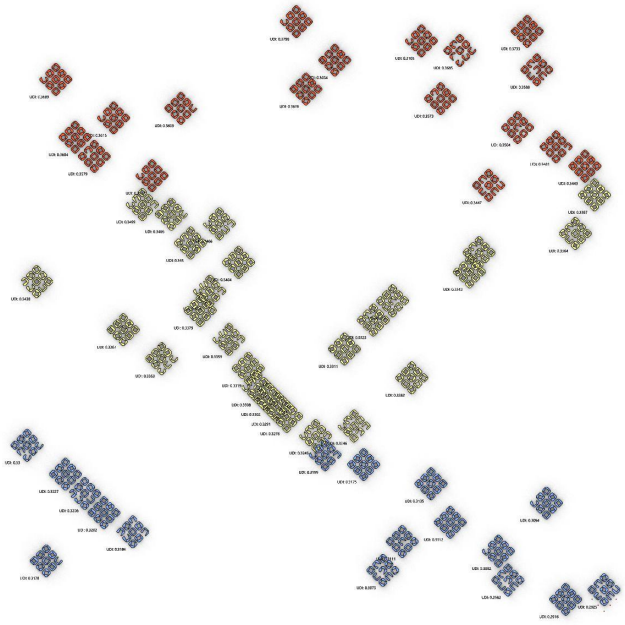


Fig 3_3. Top view - K-Means Clustering - NSGA2 pareto front

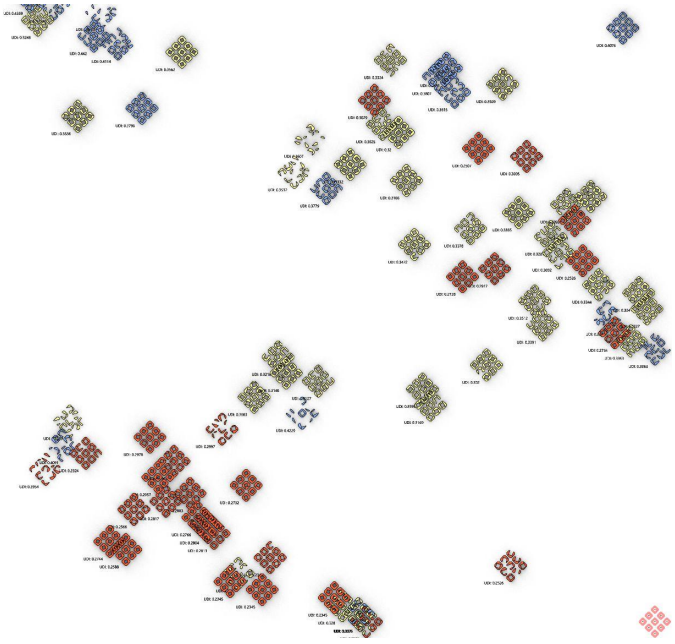


Fig 3_4. Top view - K-Means Clustering - RBFMOpt pareto front

Results:

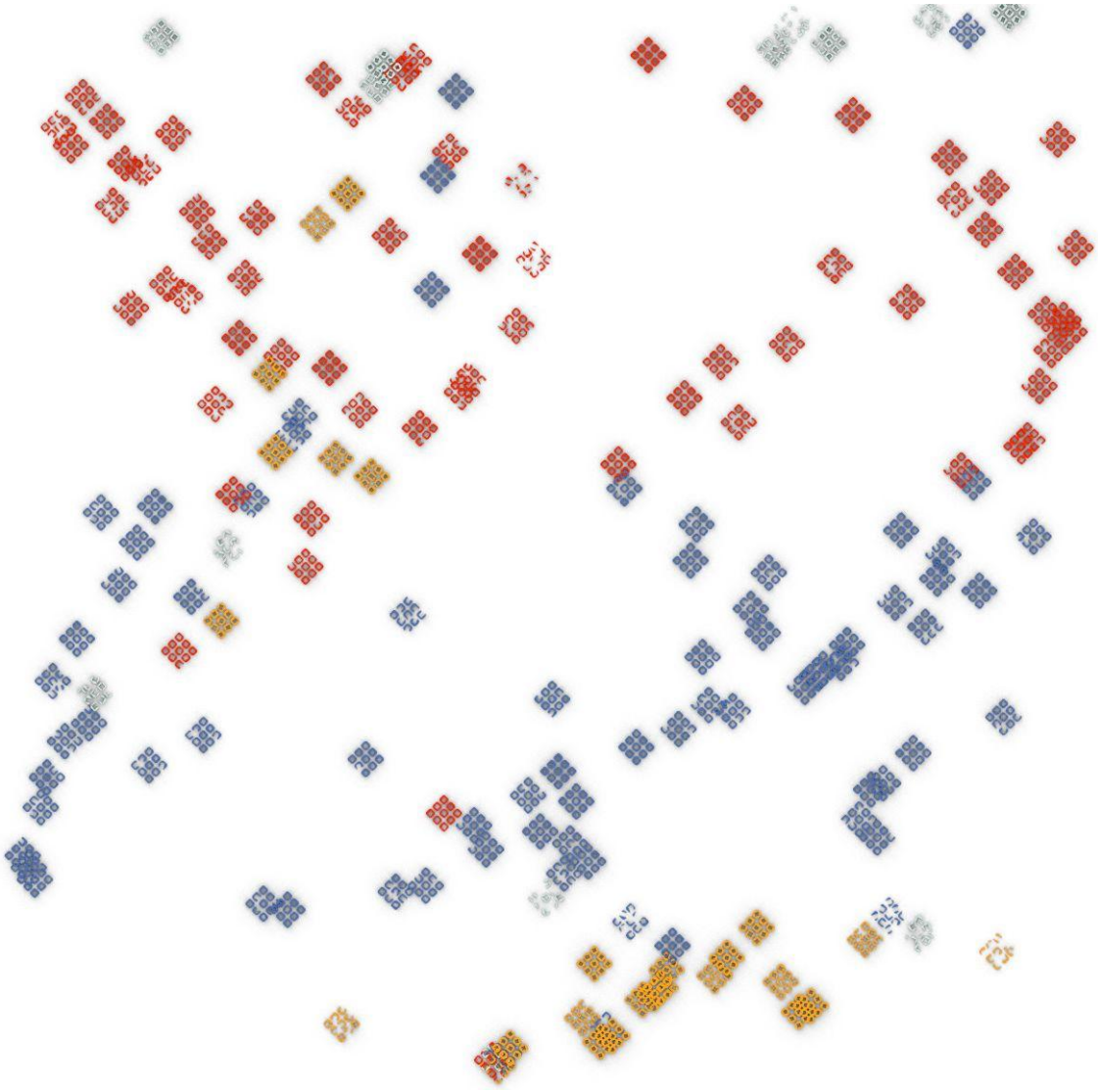


Fig 3_5 - Top view - All results at the pareto front of RBFMOpt, NSGA-2 and HypE *clustered by objectives*

Conclusion:

The clustering only by two objectives did not turn out to be very insightful since the achieved results could be also easily analysed by more traditional ways of statistical analysis and did not inherently provide the designer with a better insight of different types of design option., Also the use of only two parameters basically made the use of dimensionality reduction obsolete, so we decided to use clustering also for more than the objectives.

One approach was to use K-Means for clustering our results by some of the parameters used, as for example in the case shown in (*Fig 3_6*.). Here the input parameters are used to create a grouping of meshes by a slight editing of the parameters. The first parameters is the variance of the building heights for each block. Although this was not used as an objective for the optimisation from a design standpoint this seems definitely of interest. So the variance of all height parameters for each superblock are calculated and used as the first feature.
A second feature the area of the courtyards of the blocks that cannot contain buildings is chosen, so an other feature directly influencing more the design expression than the objectives of optimisation.

In the shown example the pareto front values of the RBFMOpt algorithm are used.

'STATISTICAL DETAILS'

	0	1
count	79.000000	79.000000
mean	2.484070	0.071166
std	0.856481	0.048574
min	1.079077	0.000000
25%	1.922789	0.028422
50%	2.400069	0.078124
75%	2.892216	0.098038
max	4.875204	0.244444

'FEATURE CORRELATIONS'

	0	1
0	1.000000	0.349153
1	0.349153	1.000000

'VARIANCES'

	0	1
0	0.674576	0.325424

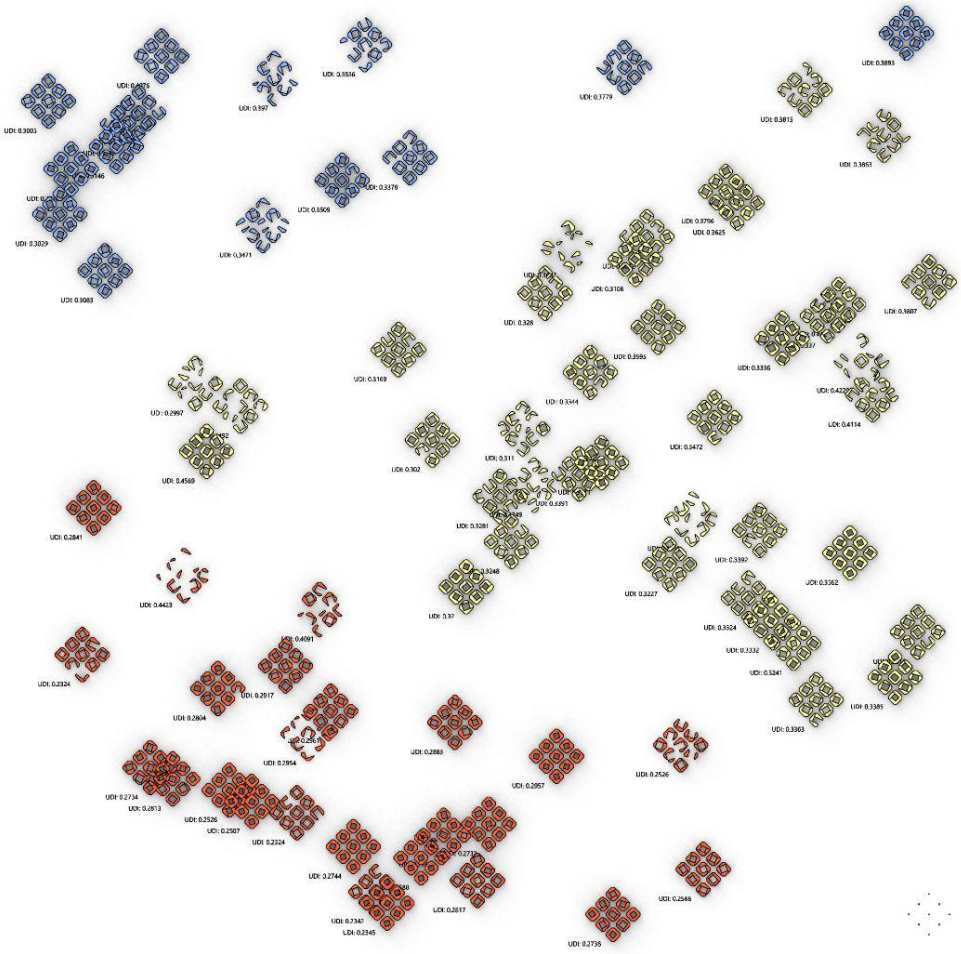


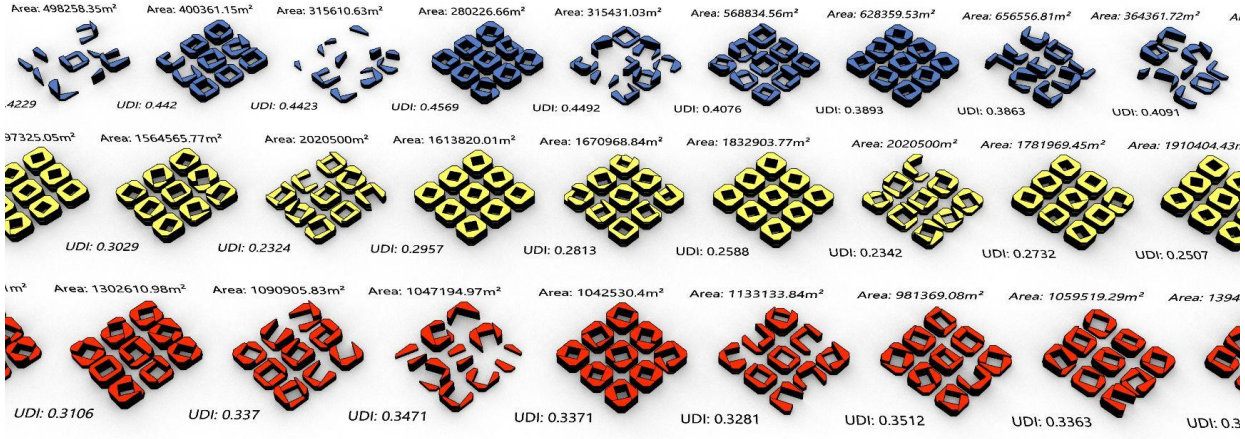
Fig 3_6. RBFMOpt best solutions clustered variance of the building heights and courtyard sizes

For the sake of higher complexity in the clusters and bringing a further viewpoint upon the best solutions of the algorithms this was further edited by combining these two “design” variables of - the variance of the height in each solutions and the size of the courtyard - with the two objectives of the optimisation runs.

So that might be a possibility to give the designer a better view on the solutions Evaluating by the edited parameters might be able to say something about the design in relation to the optimisation aims.
In this case the cluster has four features and cannot be shown so easily on a two dimensional plane as before, except one would choose only two features for this representation, as seen on the pages before.

But it may be more plausible in relation to the more dimensional feature correlation, if the solutions are simply lined up in their clusters, that there is no spatial relation pretended, that is not visible in the data.
Still it seems important to show the values of the objectives of the single solutions in the graphical representation, since they were already chosen before is most relevant for the design.

Again the pareto front values of the RBFMOpt algorithm are used.



'STATISTICAL DETAILS'

	0	1	2	3
count	79.00000	7.900000e+01	79.000000	79.000000
mean	0.32729	1.214250e+06	0.071166	2.484070
std	0.05276	4.615598e+05	0.048574	0.856481
min	0.23240	2.802267e+05	0.000000	1.079077
25%	0.29355	9.337458e+05	0.028422	1.922789
50%	0.32480	1.130081e+06	0.078124	2.400069
75%	0.35470	1.616781e+06	0.098038	2.892216
max	0.45690	2.020500e+06	0.244444	4.875204

'FEATURE CORRELATIONS'

	0	1	2	3
0	1.000000	-0.982664	0.703514	0.311771
1	-0.982664	1.000000	-0.754674	-0.300816
2	0.703514	-0.754674	1.000000	0.349153
3	0.311771	-0.300816	0.349153	1.000000

'VARIANCES'

	0	1	2	3
0	0.70079	0.209588	0.086113	0.003509

Fig 3_7. RBFMOpt best solutions clustered variance of the building heights and courtyard sizes and objectives

Description: Sensitivity Analysis

Because we had so many parameters, we performed a sensitivity analysis to determine how much each parameter actually contributed to the overall results. This was done for both a single building and for an entire superblock. We didn't find any one distinct parameter that was the main contributor to either of the objectives. There were parameters that contributed more than others, but they weren't really consistent across the 9 buildings. A more detailed analysis needs to be conducted to identify any patterns in the the effects of the parameters.

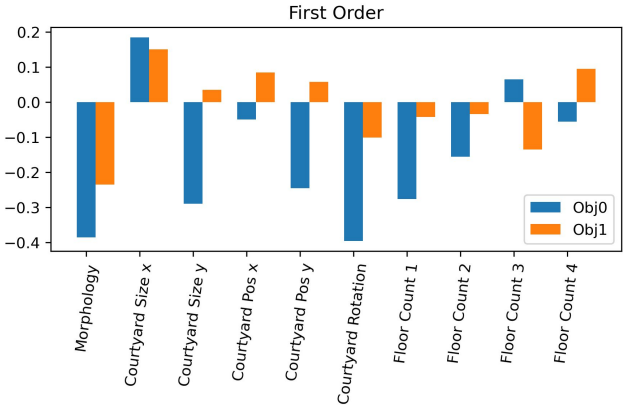


Fig 3_8. 1st Order SA of one building

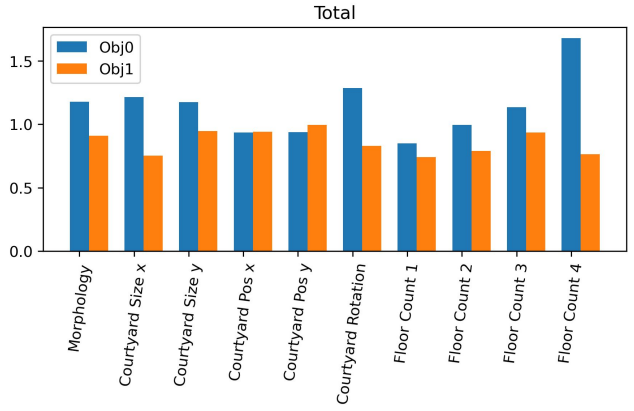


Fig 3_9. Total SA of one building

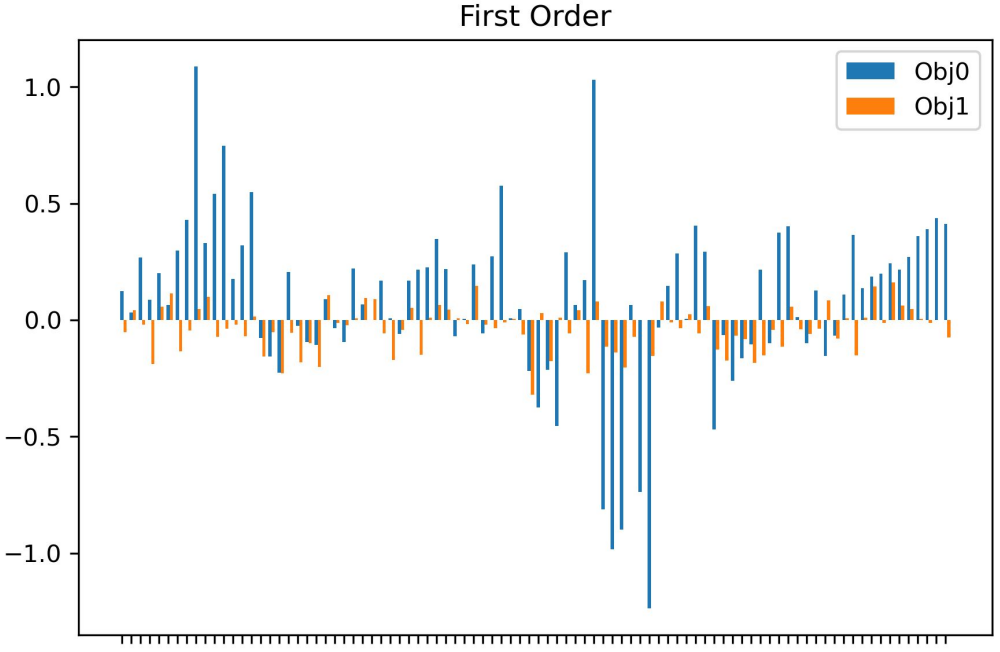


Fig 3_10. 1st Order SA ofall 9 buildings

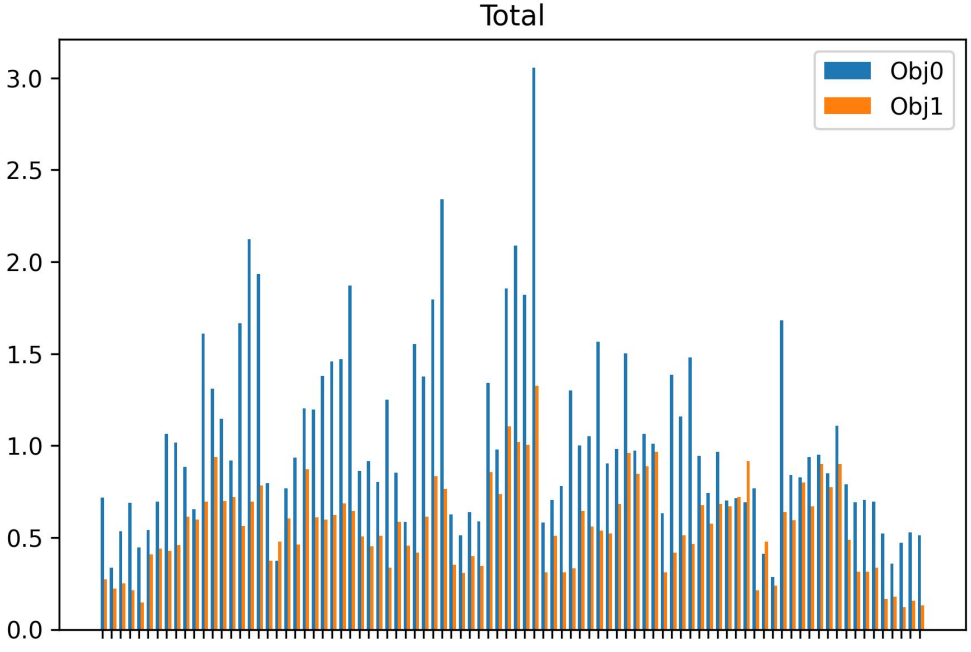


Fig 3_11. Total SA of all 9 buildings

Description:

Performing the solar analysis of so many surfaces, even with our reduced settings still is very computationally expensive and time consuming. In order to investigate more efficient ways of calculating accurate results we attempted to train a supervised machine learning algorithm so that we could generate the UDI without using Climate Studio

After a few unsuccessful attempts to effectively train an entire superblock, we decided to simplify our experiment by only analyzing one building. Initially we created a surrogate model to calculate both UDI and FAR, but we realized that there is no need to do it for FAR because FAR isn't computationally expensive to calculate on its own. We repeated the experiment with just UDI and generated similar results.

The training data was generated using Latin Hypercube sampling to create roughly 1800 near-random data values.

Results:

After testing many different training and test sizes, polynomial regressions, support vector regressions, and polynomial degrees, cross validations, and kernel types we found that a polynomial regression of degree 2 with 10 cross validations using a 66% - 33% split between training and test data created the highest cross validation score. The final settings and results are shown in Fig 4_2.

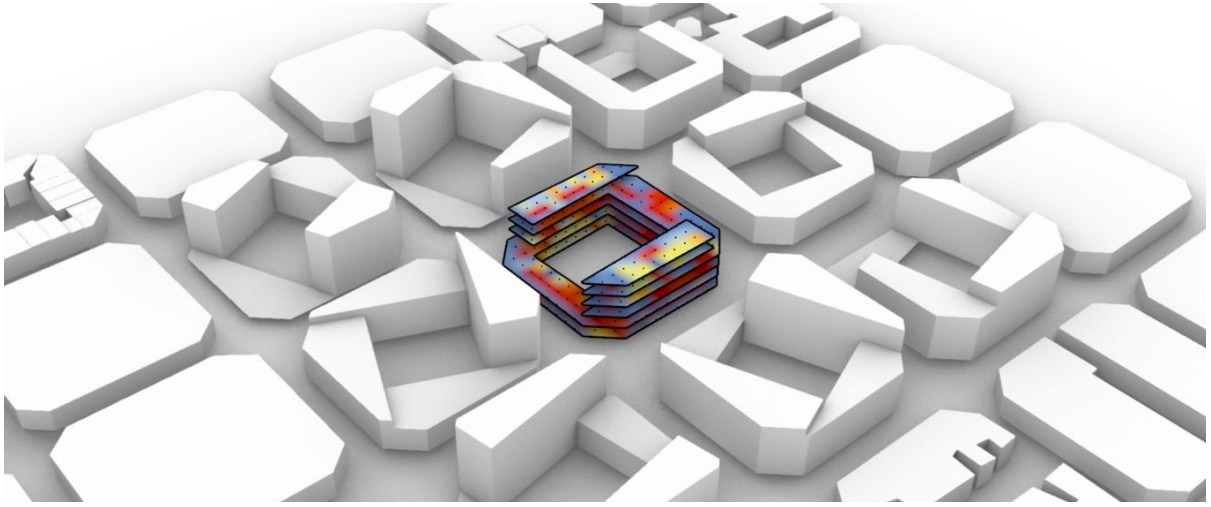


Fig 4_1. Example of UDI analysis on a single building

Model Type	Poly / Linear	SVR
Training data samples	1800	1800
Training Size	66%	66%
Test Size	33%	33%
Type / Kernel	Polynomial	Polynomial
Degree	2	2
Cross Validations	10	10
Cross Validation Score	0.68	0.14

Fig 4_2. Example of UDI analysis on a single building

Results:

In order to compare the accuracy of the surrogate model to the simulation we took 100 samples of randomly generated parameters and their resulting UDI calculations. From each sample we calculated the percent difference between the ground truth (simulation) and prediction (surrogate model). Out of these results the average percent difference between the two methods was 4.105%. It is worth noting however that there were some concerning outliers that got as high as 41.5% difference.

Out of curiosity we lowered the number of parameters that our training script was solving for and found that as the number of parameters decreased, the cross validation score increased. This is not a valid metric or comparison because the samples used to train the model still had 10 parameters, but it may suggest that we need many more samples to get a sufficiently accurate model for the number of parameters we have.

Conclusions:

This experiment could be further improved by getting a larger data set to train with. It would also be beneficial to do a sensitivity analysis to check how impactful each parameter is. In a later experiment we could use deep learning to generate a (potentially) more accurate model.

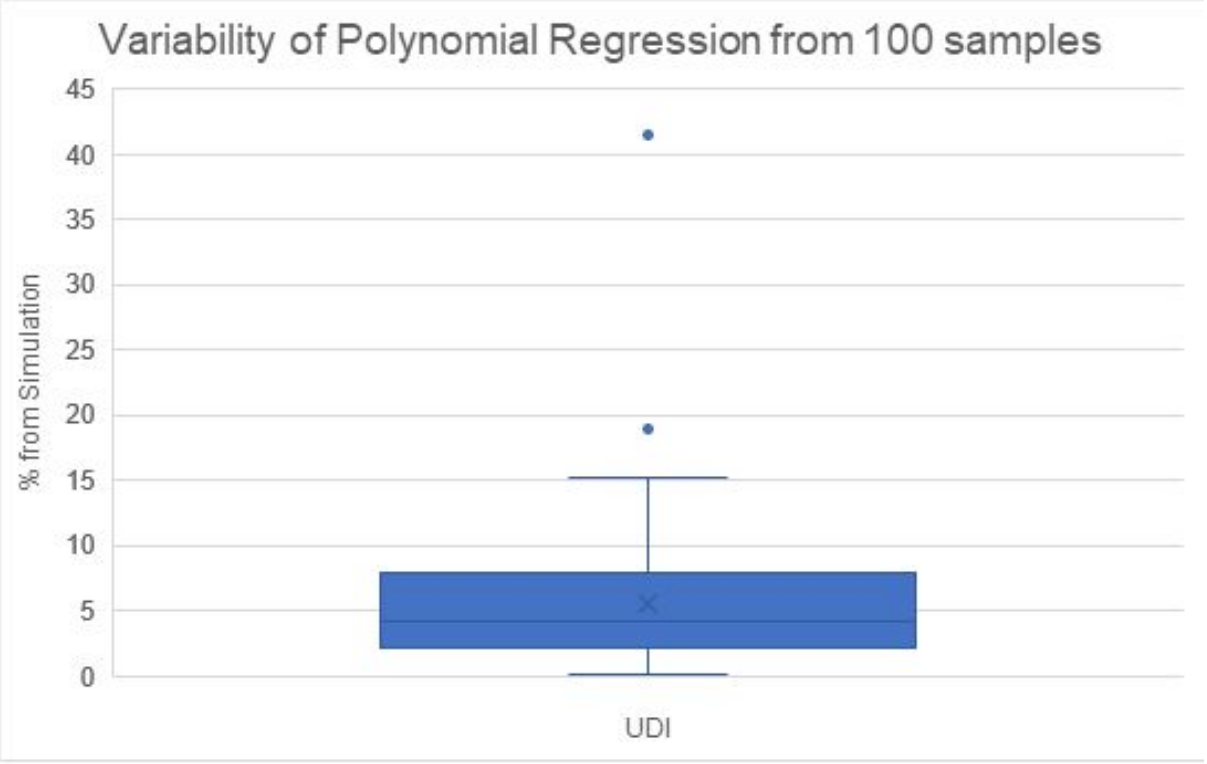


Fig 4_3. Percent difference from true Climate Studio simulation

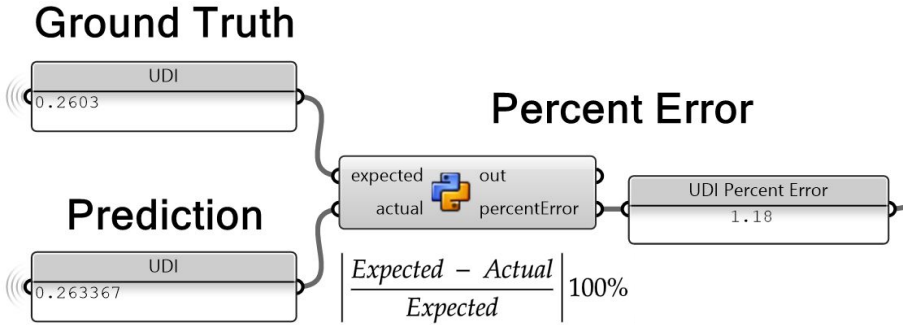


Fig 4_4. Percent error calculation

Description:

For generative deep learning, we were interested in applying image style transfer with two approaches; one more technical and street layout-focused, and one centred around artistic style. Unlike many unsupervised learning tasks in machine learning, image style transfer doesn't require a massive input of source material, making it feasible for the scope of this experimentation.

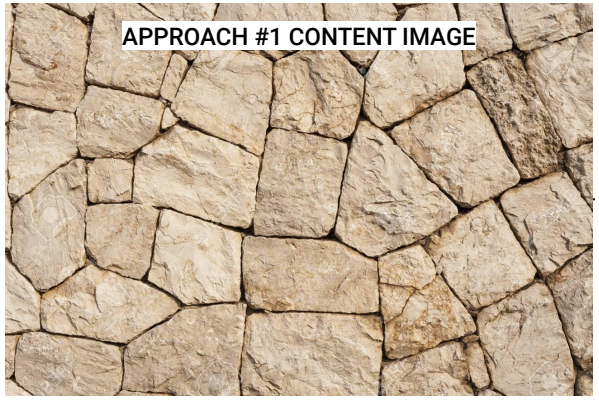
We experimented with the following two ideas:

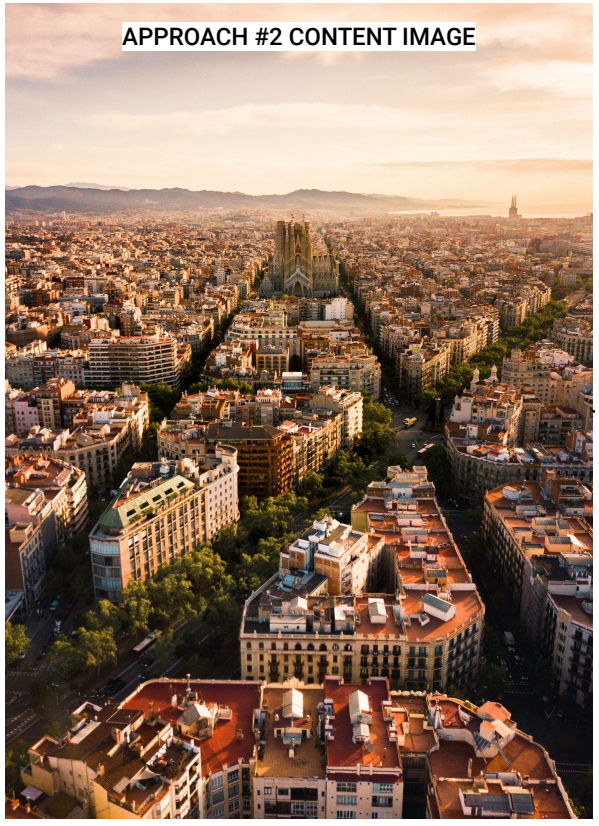
Approach 1:
Re-styling the urban layout of Barcelona streets through image style transfer.

Approach 2:
Transferring the artistic style of a Barcelona photo to that of a notable artist.

For each experiment, we toggled with style and content weight factors in the algorithm, resulting in style transferred images that more so reflected the style images, seen on the right, or the content images, seen left, respectively. We found the most interesting results to occur at a style weight equal to 1e-2, and content weight equal to 1e4. All experiments involved 50 epochs and 100 steps per epochs, resulting in 5000 total iterations.

For each experiment, we plotted the results for comparison and analysis. The analysis process involved plotting the loss graphs, including the Total Loss, Style Loss, Content Loss and Variation Loss.





Approach 1:

CONTENT IMAGE
Floor Tiles

STYLE IMAGE
Barcelona Grid

TRANSFERRED IMAGE

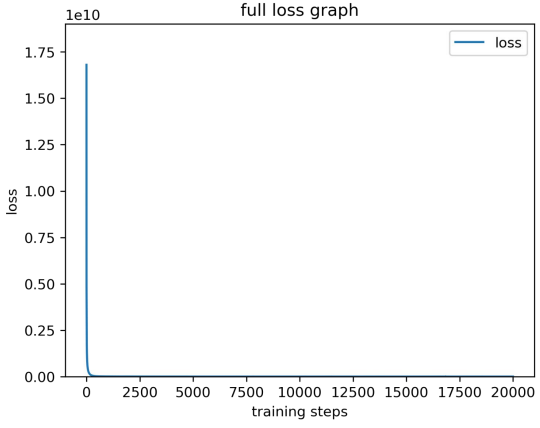


Fig 5_1. Full Loss Graph

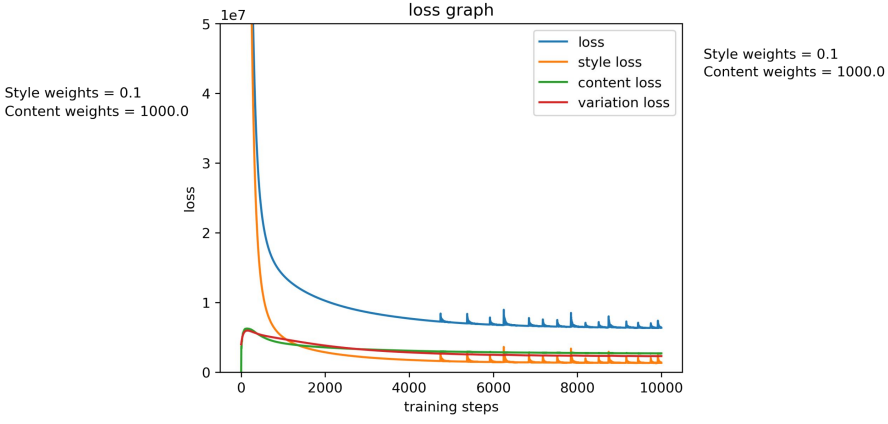


Fig 5_2. Loss Graph

Approach 1:

CONTENT IMAGE
Floor Tiles

STYLE IMAGE
Barcelona Grid

TRANSFERRED IMAGE

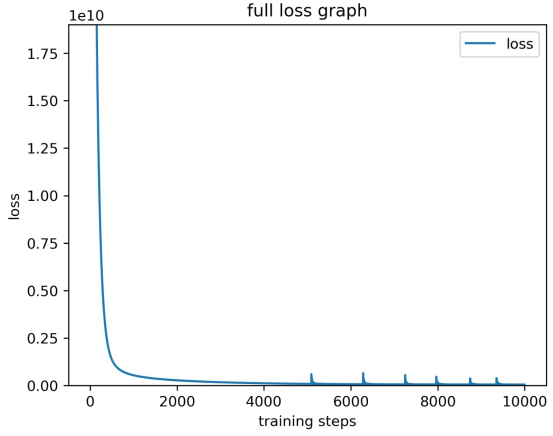


Fig 5_3. Full Loss Graph

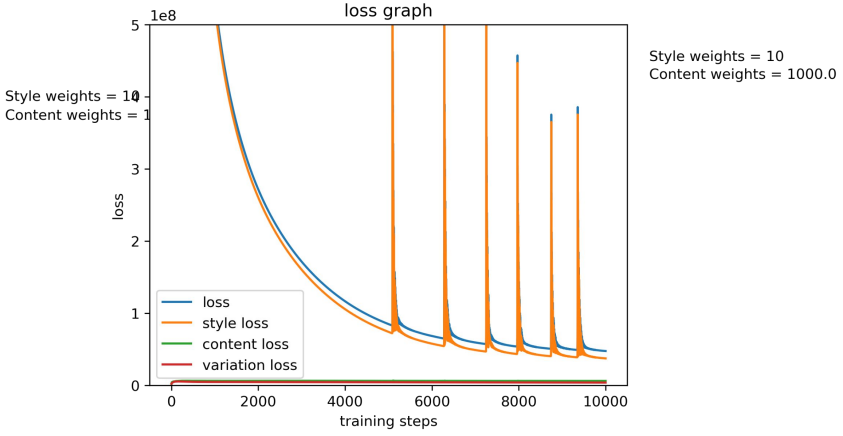
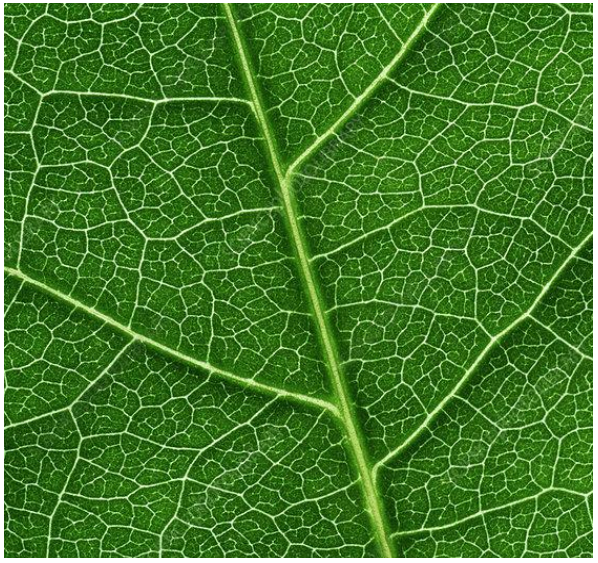
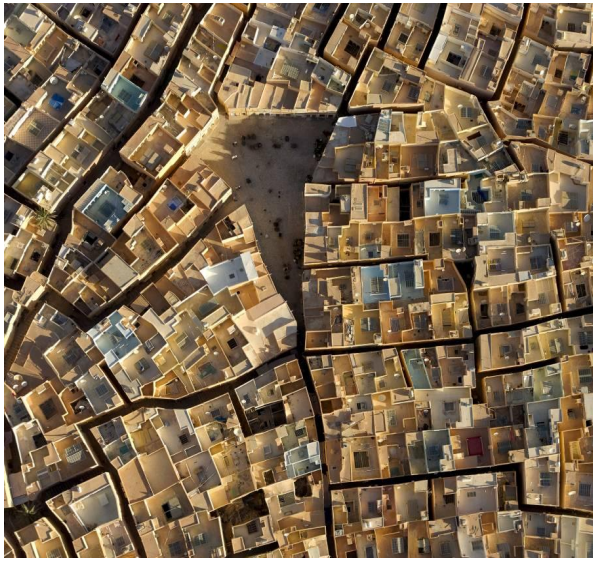


Fig 5_4. Loss Graph

Approach 1:



CONTENT IMAGE
Floor Tiles



STYLE IMAGE
Ghardai - Algeria

TRANSFERRED IMAGE

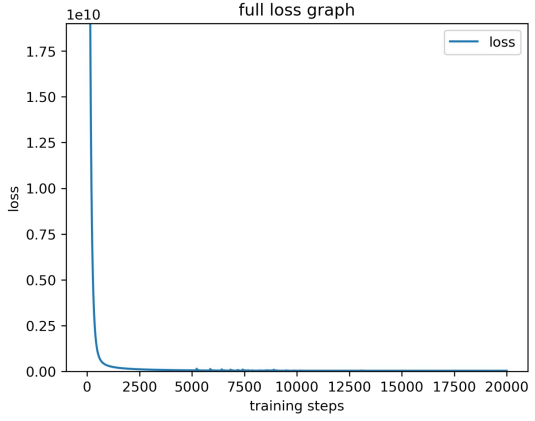
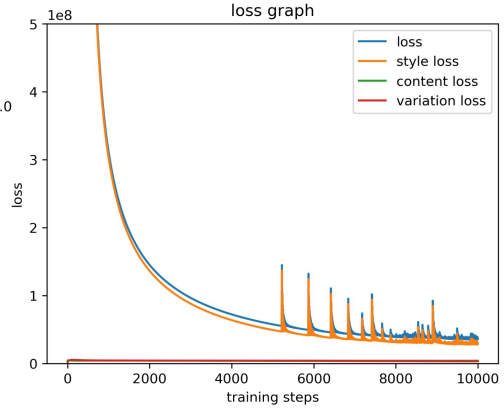


Fig 5_5. Full Loss Graph

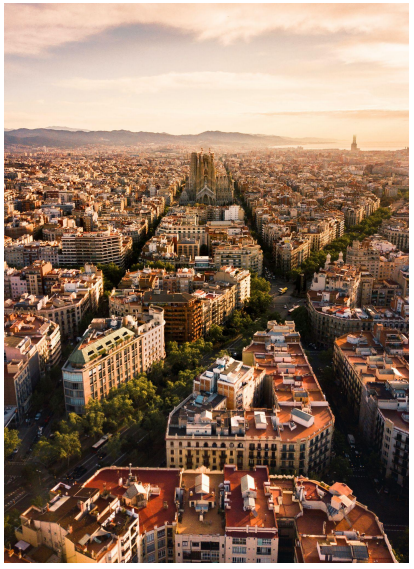
Style weights = 10
Content weights = 1000.0



Style weights = 10
Content weights = 1000.0

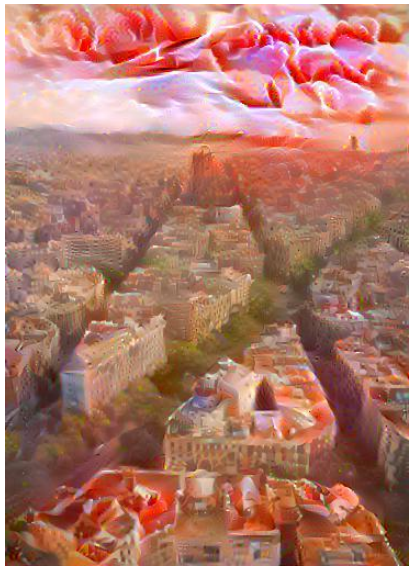
Fig 5_6. Loss Graph

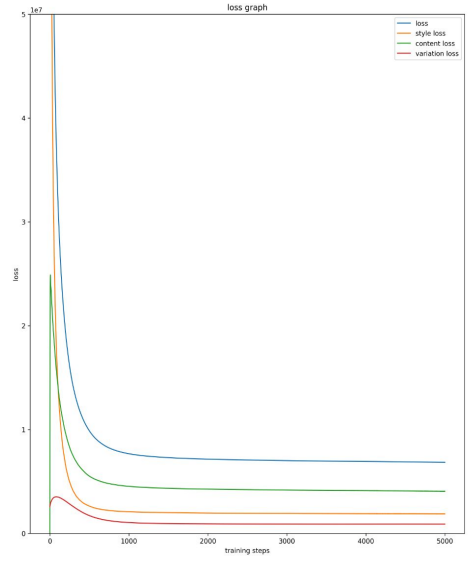
Approach 2:



CONTENT IMAGE
Barcelona Perspective

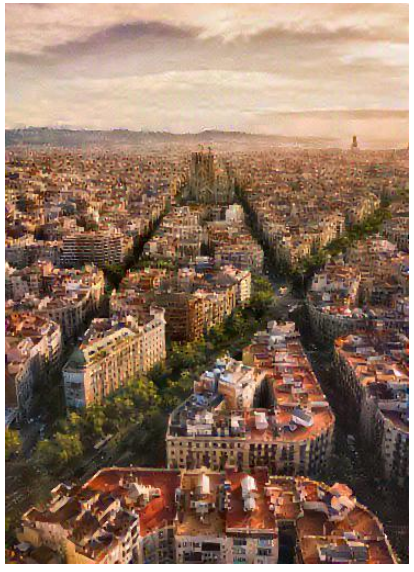
STYLE IMAGE
Strawberry Cream Skies

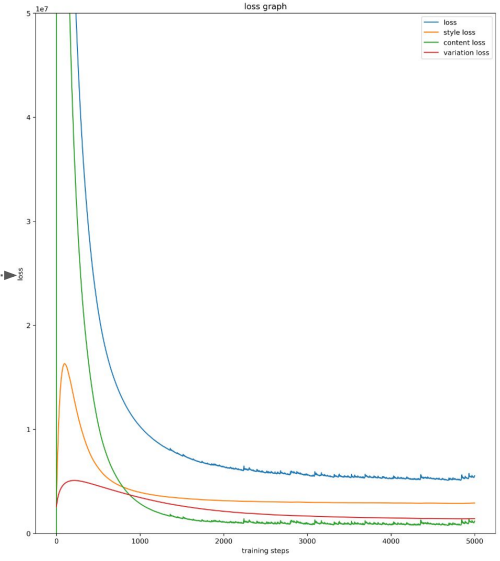




STYLE WEIGHT = 1e-2 **CONTENT WEIGHT = 1e4**

Fig 5_7. Loss Graph

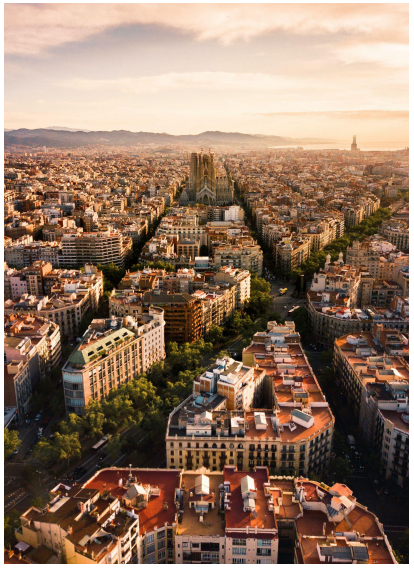




STYLE WEIGHT = 1e-4 **CONTENT WEIGHT = 1e6**

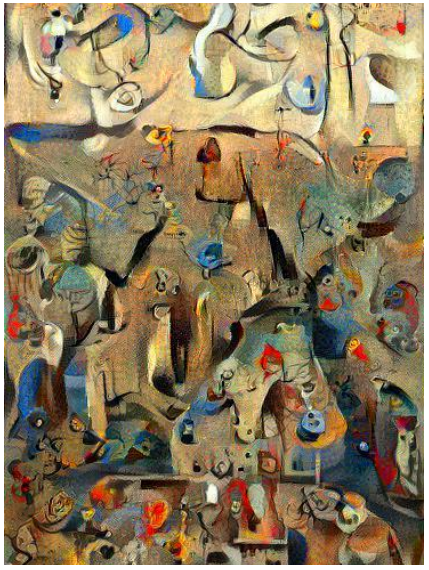
Fig 5_8. Loss Graph

Approach 2:



CONTENT IMAGE
Barcelona Perspective

STYLE IMAGE
Joan Miro, Harlequin's Carnival



STYLE WEIGHT = 1e-2 **CONTENT WEIGHT = 1e4**

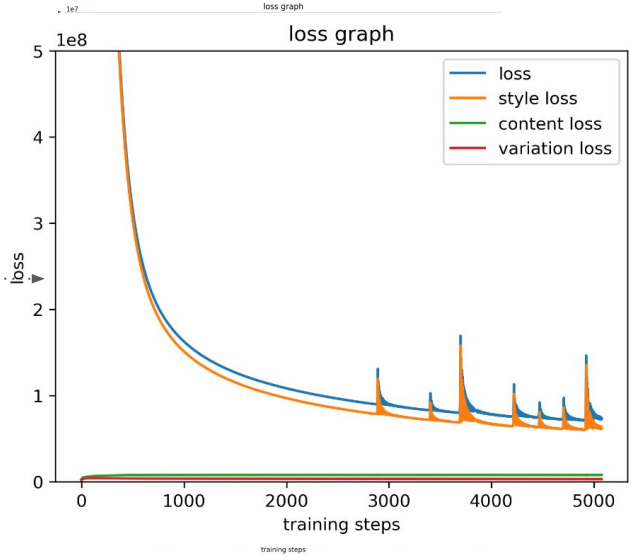
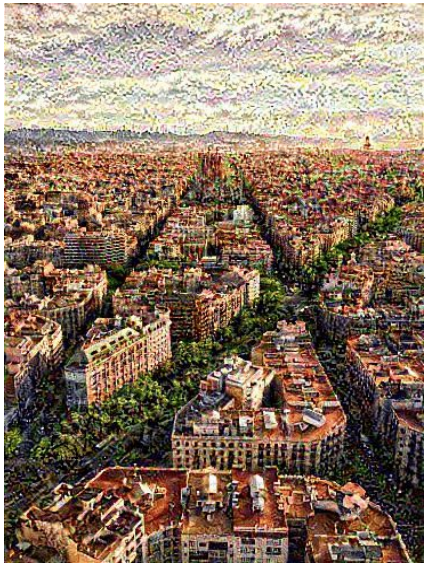


Fig 5_9. Loss Graph



STYLE WEIGHT = 1e-4 **CONTENT WEIGHT = 1e6**

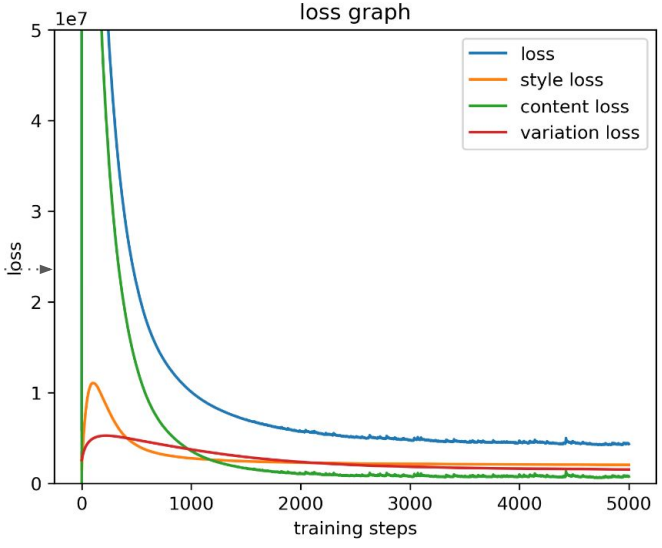
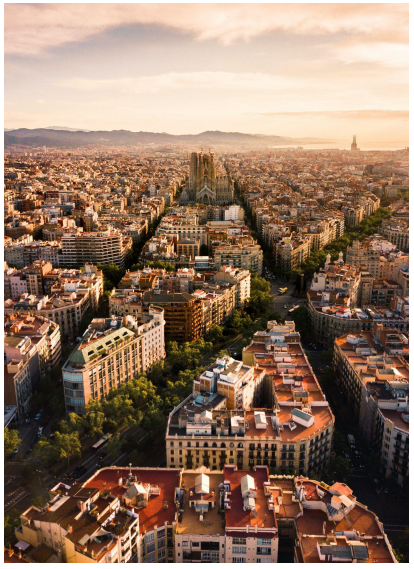


Fig 5_10. Loss Graph

Approach 2:



CONTENT IMAGE
Barcelona Perspective

STYLE IMAGE
Piccaso Guernica

STYLE WEIGHT = 1e-2 **CONTENT WEIGHT = 1e4**

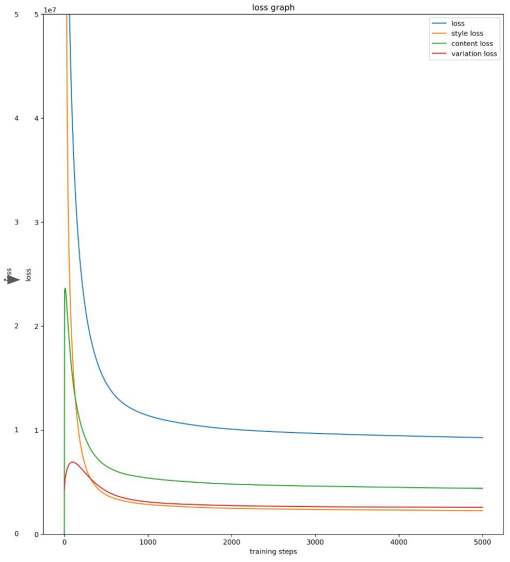
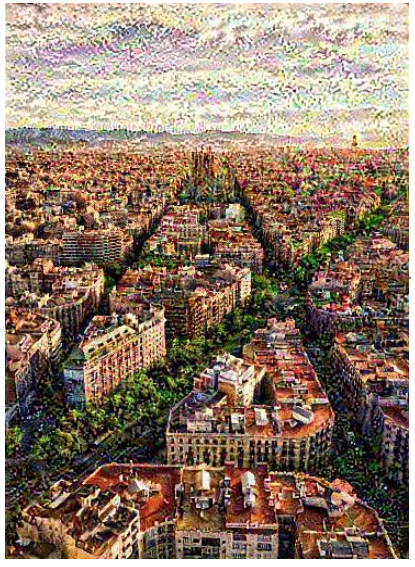


Fig 5_11. Loss Graph



STYLE WEIGHT = 1e-4 **CONTENT WEIGHT = 1e6**

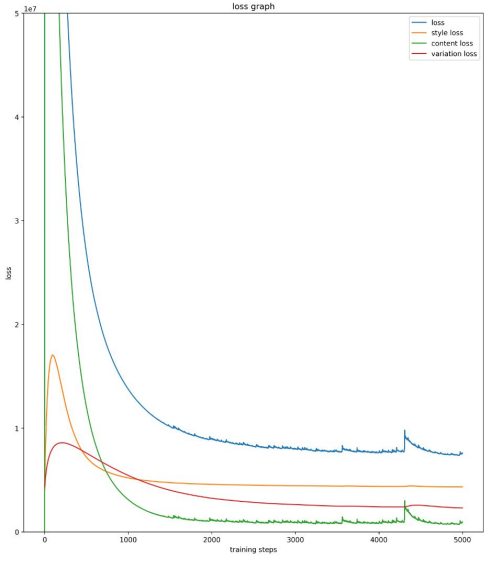


Fig 5_12. Loss Graph

Conclusions:

To complete this experimentation with neural style transfer as our generative deep learning study, we utilized TensorFlow learning resources to generate code and run training loops for optimising the above-mentioned content images. The goal was to integrate style image features into content images in a visually interesting way. Experimenting with the weight factor assigned to style versus content image allowed us to benchmark results, comparing style, content, variation, and overall loss graphs. While these studies were visually interesting from a stylistic perspective, we didn't find significant overlaps between this work and the topics covered in the above SOO, MOO, and supervised and supervised machine learning sections of the report. These above-mentioned sections of work focused on technical improvements to the design layout of the urban landscape. In contrast, the generative deep learning section taught us, from a more visual perspective, how the steps taken through iterative and generative learning algorithms build towards the final result. Additionally, this experimentation provided visual feedback on the impact of algorithmic weight factors.

This section of our experimentation, while aesthetic-centric, was valuable in understanding the generative algorithmic process. It would be really interesting to extend this experiment in to 3 dimensions. We could foresee this process applied towards Virtual Reality or cinematic experiences, in which one is able to experience a world or space with a stylistic filter or lens with which their favourite artist, director, etc., would view it.

Fig 5_13 Generative Deep Learning Experimentation

Conclusions:

Ultimately, this research exercise gave us a great introduction into the world of computational exploration in architecture. We learned the basics of many techniques and got the chance to apply them to a holistic architectural scenario. We feel that we covered a lot of ground and produced a lot of good work throughout the semester. However, we recognize this is just the beginning of the potential of these tools. Now that we have a base knowledge of how these areas work we can apply it to a wide variety of problems and, hopefully, come up with some meaningful and impactful results. If we were to do this over from the beginning, there would be a lot of changes we would make to our initial parametric model and problem setup, as well as many changes to the decisions we made throughout. But that is part of the learning process and overall we are very happy with our work and the outcome of the class.

Of course we would like to give special thanks to Professor Thomas Wortmann, Zuardin Akbar, and Lior Skoury whose excellent instruction and constant assistant contributed immensely to the success of this project.