Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Architecture and Design
Department of Architecture and Technology

Gaurav Chaudhary

Decoupling the thermal and visual performance in glazing systems: a novel methodology for the numerical investigation of the case of double skin facade systems.

Master's thesis in Sustainable Architecture (M.Sc.)
Supervisor: Dr. Francesco Goia (NTNU, Norway)
Co-supervisor: Dr. Fabio Favoino (Politecnico di Tornio, Italy)
June 2019

NTNU
Norwegian University of
Science and Technology

Gaurav Chaudhary

Decoupling the thermal and visual performance in glazing systems: a novel methodology for the numerical investigation of the case of double skin facade systems.

Trondheim, Spring 2019

Master's thesis for M.Sc. in Sustainable Architecture

Supervisor: Dr Francesco Goia (NTNU, Norway)
Co-supervisor: Dr Fabio Favoino (Politecnico di Torino, Italy)

Norwegian University of Science and Technology
Faculty of Architecture and Design
Department of Architecture and Technology

NTNU
Norwegian University of Science and Technology

Abstract

Dynamic transparent envelope technologies such as double skin facades (DSFs) have been used for a long time as an efficient building envelope system aiming to reduce building energy consumption and indoor air quality. Whereas, the true performance of a highly responsive and dynamic system like DSF depends by a great extent also on the control system which controls the various aspects of such a façade system.

A DSF unit with an inbuilt shading system can be operated in various modes with option to control or regulate many components such as the airflow path and airflow rate in the cavity, shading state and slat angles, eventually, the width of air cavity itself in some newer DSFs. Despite the technological evolutions of the components of the DSF and of the DSF systems, operations and control strategies have not evolved much. In reality, although a DSF can have many working modes, only one or two are selected at each time at the design phase for each building and then used. There is a general lack of understanding of how adjustments in different components of a DSF affect the overall performance because of which it has become difficult to utilize their maximum potential. The ultimate goal of this thesis is to show how by considering a more comprehensive use of the DSF enabled by using different control components in a DSF, can be regulated to decouple, i.e. individually control, its thermal and visual performance; and de-facto change the paradigm of how DSF is used and perceived as a building component.

Different configurations of DSFs (with variations in the type of outer/inner skin, gap width and reflectivity of blinds) were used in this study together with benchmark insulated glazing units (IGUs). These façade systems were tested for thermal and visual performance against a set of realistic boundary conditions and all possible variations of operation modes. The DSFs were modelled and simulated in steady-state conditions in EnergyPlus® whereas purpose-built Python® scripts were used to pre-process, post-process and analyses the thousands of simulation cases and output results. The methodology and procedure for all the work are presented in detail which can be used to replicate the work and results achieved in the thesis.

From the analysis of results, it was seen that “Air Supply” and “Air Extract” modes presented a large range of thermal gain when compared to other airflow paths, whereas, higher airflow rate provided the maximum range which reduced with decreasing value of airflow rate. With the mathematical the model used for DSF in this thesis, no significant differences are measured in the thermal gain range of DSFs when air cavity depth was increased to even highest possible realistic value. The results outlined a trend that both range of performance and degree of freedom of decoupling was lowest when the temperature difference (between indoor and outdoor) was zero and the lowest solar radiation. Both these values increased with the increase of the temperature difference on the both sides of 0 with highest being at -45 °C and 20 °C, whereas the degree of freedom of decoupling reduced as incident solar radiation increased.

This work also includes different applications of the methodology proposed in this study. Different ways are proposed for how the results can be used to effectively design a DSF for a given climate, compare the performance of different DSFs and operate a DSF in the most efficient manner.

Keywords: Double Skin Façade, DSF, Decoupling, Performance, EnergyPlus

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr Francesco Goia and my co-supervisor Dr Fabio Favoino for their continuous support and guidance throughout my thesis. I am grateful for their interest and encouragement; and appreciate their effort for allowing me to explore along with keeping my ideas on track.

Besides my supervisors, I would like to thank all the members of the Technology Energy Building Environment research group in Department of Energy of Politecnico di Torino for hosting me in Italy for 5 months period. I would personally like to thank Ellika Taveres-Cachat, Miren Juaristi Gutiérrez, Francesco Isaia, Stefano Fantucci and Elisa Fenoglio for being my support team all throughout the semester and giving me constructive suggestions.

Table of Contents

List of figures

List of tables

Abbreviations

ηPH	Pre-heating efficiency
ASCII	American Standard Code for Information Interchange
ANN	Artificial Neural Network
CSV	Comma-separated values
DBT	Dry-bulb temperature
DGU	Double Glazing Unit
DSF	Double Skin Façade
EMS	Energy Management System
EP	EnergyPlus
EPW	Energy Plus Weather
HTML	Hypertext Markup Language
HVAC	Heating, ventilation, and air conditioning
IAT	Indoor Air Temperature
IDF	Input Data Format
IGU	Insulated Glazing Unit
RH	Relative Humidity
UDI	Useful Daylight Illuminance
VT	Visible Transmission

Nomenclature

r	Person correlation coefficient [-]
T_{vis}	Visible light transmittance [%]
U-value	Thermal transmittance of glass [W/m²K]
VLT	Visible light transmittance [%]
ε	Emissivity [%]
g-value	Solar factor [-]
F	Airflow rate [m^3/s]
A_{gap}	Gap cross-sectional area [m^2]
h_{cv}	Convective heat transfer coefficient from glass to gap air [W/m^2K]
h_c	Glass-to-glass heat transfer coefficient for non-vented (closed) cavity [W/m^2K]
v	Mean air velocity in the gap [m/s]
X_i	every data point in Thermal Metric array
Y_i	every data point in Visual Metric array
$\bar{X}$	mean of Thermal Metric array
$\bar{Y}$	mean of Visual Metric array
K	Luminous Efficacy [lm/W]
$Q_{sol,SW}$	Directly Transmitted shortwave radiation [W]
$Q_{sol,LW}$	Re-emitted longwave radiation from heat absorbed in glass [W]
Q_{conv}	Convective heat exchange between glass and zone air [W]
$Q_{incident}$	Total incident solar radiation [W]
Q_{air}	Convective heat gain to the zone air due to the gap airflow [W]
Q_{vent}	Extra energy to compensate for heating/cooling energy spent by HVAC [W]
C_p	Heat capacity of the air [J/kg-K]
ΔT	Temperature difference of extra air brought inside the zone [°C]
$C_{p,in}$	Heat capacity of the gap inlet air [J/kg-K]
$C_{p,out}$	Heat capacity of the gap outlet air [J/kg-K]
$T_{gap,in}$	Temperature of inlet stream of air from air cavity [°C]
$T_{gap,out}$	Temperature of outlet stream of air from air cavity [°C]
$\dot{m}$	Air mass flow rate [kg/s]
H	Glazing height [m]
H_o	Characteristic height [m]

ρ	Density of air [kg/m3]
s	Cavity gap width [m]

1. Introduction

1.1. Literature review

1.1.1. Definition of DSFs

The Double Skin Façade is an architectural trend driven mostly by: the desire for an all glass façade for aesthetic reasons; reduction of energy use and need for improved indoor environment for practical reasons. The term "Double Skin Façade (DSF)" refers to a large spectrum of facades. According to the Source book of the Belgian Building Research Institute [1], *"An active façade is a façade covering one or several storeys constructed with multiple glazed skins. The skins can be air tighten or not. In this kind of façade, the air cavity situated between the skins is naturally or mechanically ventilated. The air cavity ventilation strategy may vary with the time. Devices and systems are generally integrated in order to improve the indoor climate with active or passive techniques. Most of the time such systems are managed in semi-automatic way via control systems."*

1.1.2. Classification of DSFs

The cavity in DSFs can be used as a thermal buffer zone, as a ventilation channel or as a combination of the two. It may be naturally ventilated or mechanically ventilated. The natural ventilation in a DSF is driven by two main preserve differences which are caused by thermal buoyancy or directly by wind action. The former, thermal buoyancy occurs when hot air rises and cool air sinks. Air density changes when temperature changes and hence warmer air occupies a greater volume than cooler air and is lighter per unit of volume [2]. Regarding the later, it happens when wind travels from positive pressure to negative pressure. Areas on the windward side, i.e. where the wind hits the building, are characterized by a positive pressure which pushes the air into or against the building. Areas on the leeward side, i.e. the opposite side of windward, have a negative pressure which results in a suction of the air out or away from the building [3].

The cavity width may vary from "narrow cavity", for width upto 40 cm, to "wide cavity", for when width exceeds 40 cm. Narrower spaces may significantly influence airflow and air velocity whereas for the latter wider cavities often imply a higher amount of construction materials which, in turn, increase the embodied energy of the DSF [4]. The limit of 40 cm is determined by the minimum width required in the cavity for maintenance purposes.

Other classifying dimensions of a DSF involve the origin of the airflow and its destination [5], which eventually define the airflow concepts [6], which have been illustrated in Figure 1.

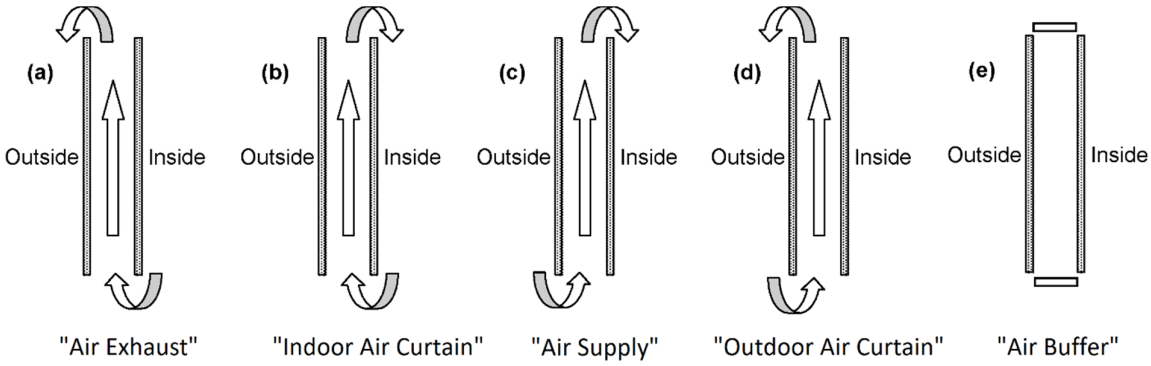


Figure 1:Possible airflows in double skin facades

1.1.3. Energy performances of DSFs

The reduction in heating loads can be seen in many ranges throughout the review. This is achieved by the use of heat trapped in air cavity. Air buffer mode can be used to create to barrier for heat loss, whereas the trapped warmer air in the cavity can also be supplied to indoor spaces (Figure 1, Supply Air and Internal Air Curtain). Baldinelli [7] showed that reductions in heating loads can be as high as 65% for a DSF when compared to single skin façade. Similar results have also been achieved in broader contexts, when DSFs have been compared to advanced single skins [8]. Significant 50% and 40% reductions of heating load due to the greenhouse effect have been found by Pappas [9] and Andjelkovicet et. al.[10] respectively, when DSFs are compared to single skins. Many investigations have concluded a reduction in heating demand ranging from 20% to 30%. Such investigations vary from field experiments in residential houses [11] to the use of DSFs as a renovation strategy for existing ones or new buildings [12].

Cooling savings seen with use of DSF correspond either to the supply of fresh air using Air Supply mode or the extraction of the heat from the occupied spaces through Air Extract mode. Additionally, the DSF can still act as a natural fan which cools off the inner skin using Outdoor Air Curtain mode. Cutbacks in cooling loads as high as 93.3% are seen in investigations done by Baldinelli [7] and 70% as seen in investigations done by Stec and van Paassen [13]. Cooling savings in the range mid-range were seen by Kragh within a 30–40% range in two cases [14,15]. Similar findings show reductions of 37.8% [15] and 38% [17].

1.1.4. Quantification of performance of DSFs

While the conventional performance metrics like U-value and g-value are sufficient to represent the thermophysical behavior of an IGU, they do not encompass the total thermophysical gain/loss of an adaptive system like a DSF. The conventional metrics cannot be used because the thermophysical behavior of a DSF is too far from the assumptions under which these metrics can be measured or calculated, which is done for specific boundary conditions. There is a need for quantitative information about the performance of systems like a DSF when they are being used in buildings. This information can be used to support the decision-making process and during the comparison with alternative advanced and traditional façade systems. For this, the performance of DSFs can be expressed in terms of indicators which is helpful for the type of analysis or evaluation is being done.

Many studies have highlighted the importance of developing different performance metrics for DSFs. The concept of pre-heating efficiency (ηPH) for transparent double skin façades using the air cavity to pre-heat the supply ventilation air has been used for the first time [18]. The same concepts have been developed further, adopting the dynamic insulation efficiency (ε) for transparent double skin façades, using the cavity air to remove solar loads transmitted through the glazing [19].

1.2. Motivation and aims

Windows plays an important role in buildings. They have to fulfil functional as well as esthetical tasks. In terms of functionality, they provide protection against the weather, provide ventilation to the inside of the building, guarantee a sufficient level of daylight and provide thermal insulation.

IGUs in combination with solar shading devices can became active construction elements which support the conditions of a building. IGUs minimize the heating energy demand of buildings by maintaining a positive energy balance, i.e. higher passive solar gains than transmission heat losses; and also maintain required daylighting levels using shading devices. Shading devices can either be fixed like drapes or curtains; or movable devices like blinds. Fixed devices provide two options of either blocking all solar radiation transmission or keeping all, whereas movable devices can be tweaked to get a desirable amount. These devices can be mounted external, internal or in the inter-

pane space. Externally mounted elements shade the glazing itself and prevent the penetration of solar radiation to the interior. When external shading devices are used there is no significant heat gain from outside except due to the heat transfer due to the temperature difference between inside and outside. Internal elements partly reflect the solar radiation transmitted through the glazing back to the outside. The remaining portion is absorbed and transferred to the interior via convection and infrared radiation.

The largest portion of heat gain through a window is due to the solar gain, i.e. shortwave radiation due to the sun. The short-wave solar radiation (wavelength 380 to 780 nm) is partly reflected and absorbed at the window panes. The remaining fraction of the solar radiation is being transmitted to the building's interior. Since the visible spectrum of light is also within the shortwave range, it leads to the fact that the heat gain and the visible gain are linked. This direct correlation seen between the thermal and visual gain from the windows means when the shading device is used the shortwave transmission is blocked which stops all of visible gain and a major portion of heat gain.

A question naturally arises because of the relationship seen before, "*Are there any types of transparent façade systems where thermal gain and visual gain can be decoupled, i.e. can be controlled separately?*" A double-skin façade, or DSF, seems to be answer to the question. To decouple thermal and visual gain one has two options: one is to act at material levels looking for materials capable of selectively differentiate between the transmission/absorption/reflection of electromagnetic radiation, like electrochromic or thermochromic glass panes; or to act to the absorbed part of the solar thermal radiation and to release it either towards the inside or the outside, according to the needs.

A DSF is an envelope construction composed of two transparent "skins" that are separated by an air cavity. The DSF is a form of an active façade because it employs equipment, like fans or solar/thermal sensors. DSF also employs solar shading devices. For the protection and heat extraction reasons during the cooling period, shading devices are placed inside the cavity. Like in IGUs, shading devices in DSFs can also be used to control the amount of visible light transmission inside to the occupied space. Whereas, the air in the cavity can be used to control the amount of thermal gain to outside. When outside is colder than required inside temperatures the heated air in the cavity can be circulated to the occupied space to offset heating requirements, while in opposite outdoor conditions the air in the cavity can be vented out of the building to mitigate solar gain and decrease the cooling load. The amount of gain or loss can also be controlled with the amount of airflow rate in the cavity. As can be seen, the two components, i.e. the shading devices and the air in the cavity, are independent of each other in terms of functionality and hence in theoretical terms, it can be said that these two components can be controlled individually to control the thermal and visual gain independently.

Although the use of Double Skin Façades has increased radically in buildings all throughout the world, the operation modes which are used is majorly confined to a few combinations. Apart from the type of ventilation inside the cavity, i.e. if natural, fan-supported or mechanical; the origin and destination of the air differ depending mostly on climatic conditions, the use, the location, the occupational hours of the building and the HVAC strategy. Theoretically there are many modes in which DSFs can work, as seen in Figure 1, in reality, only one or two are selected at each time at the design phase for each building and then used. Therefore, it is a choice that limits the potentials of using this archetype, since it is applied only within a limited range of flexibility. There is a general lack of understanding of how adjustments in different components of a DSF affect the overall performance because of which it has become difficult to utilize their maximum potential. The ultimate goal of this thesis is to show how, by considering a more comprehensive use of the DSF enabled by using different control components in a DSF can be regulated to decouple, i.e. individually control, it's thermal and visual performance; and de-facto change the paradigm of how DSF is used and perceived as a building component.

1.3. Research questions and objectives

In line of the aim of this thesis, the research questions of this thesis can be grouped in three part. They are as follows:

Developing different kinds of metrics

1. How to quantify thermal and visual performance of DSF?
2. How to quantify the performance range and decoupling degree of a DSF?

Analysis of the performance of a DSF

3. How big is the thermal and visual performance range of a DSF?
4. How does different operation modes affect the different performance of a DSF?
5. How does performance range of a DSF change with boundary conditions?
6. How decoupled is the thermal and visual performance of a DSF?

Applications of this work

7. How this work can be used to design and operate a DSF for a certain climate?

In order to answer the research questions, certain objectives were decided for this work. There are as follows:

1. Define performance metrics for thermal and visual gain through the DSF into an occupied zone.
2. Define set of realistic boundary conditions and operation modes for a DSF through which mathematical model of DSF will be tested.
3. Design and implement the pipeline of simulation, data collection, data processing and plotting.
4. Create possible methodologies for future work about applications of work done in this thesis

1.4. Overview of research methodology

To check the performance range of double skin façades same approach is used as is typically used to test the performance of a new technology, i.e. test the technology over all possible modes and conditions it can function on and then analyses how the performance vary with variation in each mode and condition. The steps described below are the core workflow steps which have been used in this thesis. They are as follows:

1. Step 1: Model different kinds of DSFs in a desired building energy simulation tool which has an in-built physical-mathematical model of DSF.
2. Step 2: Run comprehensive steady state simulations to replicate as many configurations as possible (boundary condition and operational modes) for a DSF.
3. Step 3: Gather the output from the simulations, which are in terms of physical quantities in performance metrics dedicatedly developed for this scope of the work
4. Step 4: Analyze the dependency of performance metrics on boundary condition and operational modes, and how the two are intercorrelated.

EnergyPlus version 8.6.0 was selected as the simulation platform to perform steady-state simulations, for various reasons. It is one of the newest and most advanced stand-alone building energy simulation programs. A dedicated component name “Airflow Windows” is present in EnergyPlus to model and simulate DSFs. Because of its popularity and number of people using all over the world, EnergyPlus has good online community support. Whereas the major reason why EnergyPlus was chosen was

because it is a console-based program that reads input and writes output to text files which makes it possible to be run via command line; and process input and output files using a text scripting language.

This work would require multiple simulations in the order of tens of thousands. The management of such a large number of simulations and large sets of output data cannot be done but with the use of automated processing, which takes places several times during the whole thesis work. Automated processing was required for designing simulation input parameters, simulation and output file management, post-processing of output results into usable data, plotting advanced level of visual data representation, and performing statistical analysis of all results. For all automated work, programming and scripting language Python version 3.7.3 was selected, for the reason being it can run EnergyPlus from command line and has the biggest online community support compared to any other programming language.

1.5. Structure of thesis report

The report for this thesis is structured in the following way:

Chapter 2 – Research methodology and materials: This chapter presents the detailed operative workflow of the work; types of IGUs and DSFs studied; simulation settings and mathematical model of DSF; and the data presentation schemes.

Chapter 3 – Results and discussion: This chapter presents and discusses the results for IGUs and DSFs, for how their performance changes with boundary conditions and operation modes

Chapter 4 – Applications: Here the different type of applications of the results and data gathered from this thesis are introduced. A detailed methodology of how to use predictive control algorithm along with the data gathered is introduced as one of the applications.

Chapter 5 – Limitations and discussion: Followed by a discussion of the limitations part of this study, the reason behind why certain aspects of DSF modelling were not considered and how this would be different in a real-time controller.

Chapter 6 – Conclusion: This chapter summarizes the overall work and results of this thesis

Appendix: Includes all the Python scripts developed for the work.

2. Research methodology and materials

This chapter explains the methodology and different components required to do this study. First, the operative workflow for thesis is described in detail followed by simulation settings and input parameters used. The two-performance metrics proposed for this studied have been described after that which is followed by description of different types of façade system studied in this thesis. Finally, the data presentation scheme used in this study have been illustrated and discussed which gives a peek into how the results would look like.

2.1. Description of workflow

The operative workflow for this thesis can be grouped in 4 sequential steps. The four steps are as follows:

1. Modelling: Defining and modelling different types of DSFs which are to be studied in this thesis.
2. Pre-processing: This step involves developing input parameters for simulation cases for different set of boundary conditions and operation modes.
3. Simulation: In this step, the input files for EnergyPlus simulations (i.e. IDF and EPW) were developed and simulations were run.
4. Post-processing: Here, the simulation output from simulation runs is processed to get required performance metrics.
5. Analysis: This step involves further post processing of simulation work and analysis of results.

All four steps have been described in detail below.

2.1.1.Step 1: Modelling

The different types of DSFs to be studied in this thesis were defined and modelled separately on EnergyPlus. The glazing configuration, air cavity depth and shading were varied to make different types of DSFs.

The output of this step was different IDF files for different types of DSFs. The input data file (IDF) is an ASCII file containing the data describing the building components, materials, constructions and HVAC system to be simulated.

2.1.2.Step 2: Pre-processing

This step involved making different input parameters for different simulation cases for each DSF. The list of possible discrete values for each type of input parameter was passed through a Python script which gives the output as a CSV file with all parameters as columns and all cases as rows. For all the parameters of boundary conditions and operation modes selected for this study, a total of 35,700 cases is computed. This is the number of simulation cases just for one of the nine different DSF configurations (in terms of glazing, cavity depth and shading) which were studied in this thesis. The output CSV file here was named as *Cases_In.CSV*.

The Python script made for this task iteratively took 1 possible value from every 7 sets and made one simulation case out of that. Whereas, it added one case with Airflow rate set as 0 every time before cases with variable air speeds were made. Also, when Blind State was selected as “off”, the *Slat angle* and *Reflectivity of Blinds* would be set as null as there are no blinds. These iterative selections were done by multiple nested *For Loops*, *While loops* and combinations of If/Else statements.

2.1.3.Step 3: Simulation

For every DSF type simulation case template of IDF and EPW (EnergyPlus weather) files were made. IDF and EPW files are used as input files for EnergyPlus simulation. The template files were used to make all simulation input files of cases designed in Step 1. These files had parameters values replaced with a distinctive keyword, for example, $@DBT@$ for Dry-bulb temperature in EPW file. To summarize, Dry-bulb temperature and Global diffuse solar radiation values were replaced as a distinctive keyword in EPW file. Whereas, the thermostat setpoint, Airflow rate, Airflow path, Slat angle, Reflectance of slats, and the shading control for the state of blinds were replaced as a distinctive keyword in IDF file.
Energy Plus is a dynamic simulation program which is contrary to the steady-state simulation required for this thesis. Special settings and tricks were used in the simulation settings and both input files to assure that steady state conditions were achieved.

A Python script was made which took text data of the template IDF file, replaced distinctive keywords with corresponding data from *Cases_In.CSV* file and saved the text data as *Case_x.IDF* with x as a case number. It then performed a similar process for template EPW file and saves each case input EPW file as *Case_x.EPW*. The result of this Python script was fully functional IDF and EPW files for each 35,700-simulation cases.

Another Python script iteratively took *Case_x.IDF* and *Case_x.EPW* with the same x and launched EnergyPlus simulation for each case. To save time, this script runs multiple simulations over all available processors on the computer at the same time and will take care that no simulation is run twice. In every EnergyPlus simulation at least 15 output files are developed out of which only *Output.CSV* file and *Output.HTML* is useful. As the number of simulations is really high all the useless output files have to be deleted once each simulation is finished. This is done to save storage and prevent the computer from crashing mid-simulation. The same Python script deleted every useless output file and renamed *Output.CSV* and *Output.HTML* as *Case_x.CSV* and *Case_x.HTML* respectively after every simulation was finished. The final result of this Python script was an output CSV and HTML file for each 35,700-simulation cases.

2.1.4.Step 4: Post-processing

Each Output CSV files had hourly data for 21st June, i.e. Summer solstice. Being under steady state conditions for simulations, results for each timestep are identical and de-facto independent. Hence, any of the data time-rows could be selected and time-row for 12:00 was selected. Performance metrics were calculated from the data physical quantities retrieved from this time-row. The four physical quantities are as follows: *Gap Convective exchange from airflow*, *Convective exchange with zone air*, *Directly Transmitted shortwave radiation* and *Re-emitted longwave radiation.*

A Python script was then used to gather all four physical quantities from all the output files and save it in one file. This script iteratively opened every *Case_x.CSV* file, copied 4 quantities mentioned above and computed the 5th component if it was required. With these 5 components, the script then computed *Thermal Metric* and *Visual Metric* separately according to Airflow path type. After all the 35700 cases had been computed the script opened *Cases_In.CSV* file from Step 1 and merged 5 components and 2 metrics side by side for every case and saved it as *Cases_In_Out.CSV*. This file had all parameters of boundary condition /operation mode and computed data from simulations and Step 3.

2.1.5.Step 5: Analysis

This step was the broadest step work wise. Here all the data collected in *Cases_In_Out.CSV* was used to perform different kinds of analysis and visualizations. As mentioned in the previous step

Cases_In_Out.CSV had both input parameters, i.e. boundary condition /operation mode, and output parameters, i.e. performance metrics. In this step, several 2D and 3D graphs were made using data and more post-processing was done for further analysis. For all this work plotting and scripting capability of Python was used extensively. Description to some of the major Python scripts used for data analysis and visualization is given here below.

A Python script was made to plot all data points on X, Y-axis as Thermal and Visual metrics respectively, whereas the data point marker was changed according to input parameters for that data point. The markers were changed in the following manner: increasing temperature difference (between indoor and outdoor) was shown as increasing opacity of the color of marker, increasing incident solar radiation was shown as increasing size of marker whereas different airflow paths, airflow rates and slat angles were individually shown with a different color of marker. The data points which were for "Off" blind state were marked as black dot changing size and opacity as described above. Several levels of nested For loops and If/Else condition were used on the data read from *Cases_In_Out.CSV*, whereas all the plotting was done using the Matplotlib library of Python.

Another type Python script used was designed to visualize the performance of DSF at a specific combination of boundary condition. In this script, the data from *Cases_In_Out.CSV* was taken and plotted similarly as done in the previous script but keeping a combination of temperature difference (between indoor and outdoor) and incident solar radiation value constant. For every combination, for example, -20C and 200 W/m2, programmatically boundary of the scatter plot was made by finding min and maximum vertices on every axis. This polygon represented the spread and possibility of thermal and visual performance of that DSF for certain boundary condition, whereas the area of that polygon represented a comparative value of the performance range. The data points in this area were collected to calculate Pearson correlation coefficient, *r*. The value of r gave the degree of correlation between the two metrics. If value of *r* is closer to zero, it means there is less correlation whereas value of r near to -1 and +1 represented negative and positive correlation respectively. These coefficients have been described in next sub-section in detail. This Python would plot 2D scatter plot with the boundary marked for every combination of boundary condition and also saves the values of Pearson, *r*, area of the polygon and its vertices in a separate CSV file named *dT_SR_data.CSV*. This data would be bi-linearly interpolated to get these coefficients and vertices of polygon for more possibilities of boundary conditions and in practicality for any possibility of temperature of inside/outside and incident solar radiation.

Figure 2 shows the 5 steps described above with a brief description of Python scripts used everywhere.

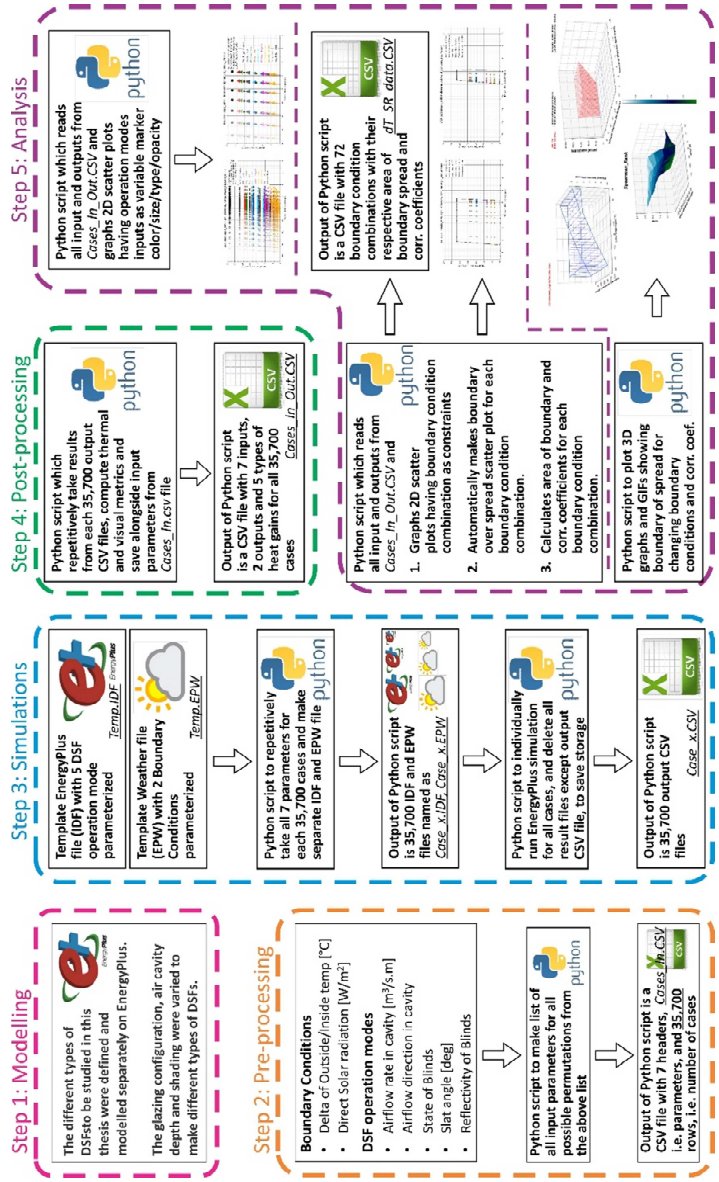


Figure 2: Illustration for four step process of workflow for this study

2.2. Simulation settings and input parameters

2.2.1.Simulation settings

The different façade systems selected for this study were simulated against different boundary conditions with different operation modes of a DSF. Each façade system was placed on the wall facing south side of a 5m x 5m box with 3.5m high walls. All the six surfaces, i.e. four walls, one floor and one roof, are well insulated which makes heat gain/loss from these surfaces almost zero. Also, no internal loads were added in the zone which results in, heat gain/loss happening from only the façade. The inside temperature of the zone was kept constant by using an Ideal air load HVAC system. The thermostat setpoint was varied to 20 °C and 25 °C; which represented winter and summer mode respectively. The main objective behind selecting these simulation settings was to keep every heat gain/loss which was “not because of façade” equal to zero, as the aim of simulations was to record the heat exchange only between façade and the building.

Steady-state simulations were done for each permutation of boundary conditions and operation mode to get the values of all 5-heat gain/loss components as shown in Figure 5, which were then processed to get 2 performance metrics, i.e. Thermal metric and Visual metric. Steady-state simulations were performed instead of dynamic simulations because only then the stabilized effect of a particular operation mode at a particular boundary condition can be seen. It was done by keeping boundary conditions same for an extended period which allowed dynamic systems to reach an equilibrium state.

EnergyPlus version 8.6.0 [20] was selected as the simulation platform to perform these steady-state simulations. It is one of the newest and most advanced stand-alone building energy simulation programs capable of modelling the hourly energy consumption of a building subject to user-specified construction, internal loads, schedules, and weather. A dedicated component name "Airflow Windows" is present in EnergyPlus to model and simulate DSFs. This component can model only forced/mechanical airflow between glass panes for which the airflow rate needs to be given as input. It can run in five different modes, i.e. Air *supply*, Air *exhaust*, *Indoor air curtain*, *Outdoor air curtain* and *Air buffer*.

In this simplified model of airflow windows [21], the convective heat transfer coefficient for heat transfer inside faces of glass in cavity to cavity air is calculated as follows:

$$h_{cv} = 2h_c + 4v \tag{1}$$

where,
h_{cv} = convective heat transfer coefficient from glass to gap air (W/m^{2}K)
h_c = glass-to-glass heat transfer coefficient for non-vented (closed) cavity (W/m^{2}K)
v = mean air velocity in the gap (m/s)

The air velocity is determined by the gap cross-sectional area and air flow rate which is the user input value in the IDF file:

$$v = \frac{F}{A_{gap}} \text{ (m/s)} \tag{2}$$

where,
F = airflow rate (m^3/s)
A_{gap}= gap cross-sectional area (m^2)

The outlet air temperature of gap is calculated as function of average temperatures of inner faces of glasses around cavity and inlet gap temperature given by following expression:

$$T_{gap,out} = T_{ave} - \left(T_{ave} - T_{gap,in}\right)e^{\frac{-H}{H_o}} \tag{3}$$

where,
H = glazing height (m)
$T_{gap,in}$= gap air inlet temperature (Indoor temperature if the airflow source is indoor air, Outdoor temperature if the airflow source is outside air) (K)
T_{ave} is as follows:

$$T_{ave} = \frac{T_2 + T_3}{2} \tag{4}$$

T_2 , T_3 = Inner face temperatures of glasses around cavity where subscript number represents face number counting from outside
H_o = characteristic height (m), given by:

$$H_o = \frac{\rho C_p s}{2 h_{cv}} v \tag{5}$$

ρ = density of air (kg/m^3)
C_p = heat capacity of air (J/kg-K)
s = cavity gap width(m)

Hence, the convective heat gain to the zone air due to the gap airflow is:

$$\dot{q}_v = \dot{m}\left(C_{p,out}T_{gap,out} - C_{p,in}T_{gap,in}\right)(\text{W}) \quad (6)$$

where,
$C_{p,in}$ = heat capacity of the gap inlet air (J/kg-K)
$C_{p,out}$ = heat capacity of the gap outlet air (J/kg-K)
and where the air mass flow rate in the gap is:

$$\dot{m} = \rho F(\text{kg/s}) \quad (7)$$

When a shading device is installed in the air cavity, EnergyPlus assumes the shading device is in the center of cavity and airflow, F, is divided equally. The convective heat gain to the zone air due to the airflow through the two gaps is now calculated as:

$$\dot{q}_v = \dot{m}\left(C_{p,ave,out}T_{gap,ave,out} - C_{p,i}T_{gap,in}\right)(\text{W}) \quad (8)$$

where the average temperature of the two outlet air streams is:

$$T_{gap,ave,out} = \left(T_{gap,1,out} + T_{gap,2,out}\right)/2 \quad (9)$$

and
$C_{p,ave,out}$ = heat capacity of the outlet air evaluated at $T_{gap,ave,out}$ (J/kg-K)

Although whole building energy simulation programs like IDA ICE and TRNSYS can model and simulate DSFs, the major reason why EnergyPlus was chosen for this study was its ability to be easily programmed to run thousands of simulations with changing parameters. EnergyPlus is a console-based program that reads input and writes output to text files. This enables us to programmatically pre-process the input files, launch simulation via command line and post-process result files all with a scripting program like Python. Although IDA ICE provides capability of running parametric simulations, it does not provide smooth flow of workflow the way it was intended in this study.

EnergyPlus is a dynamic energy simulation program but it can be tricked to perform steady-state simulations. It can be done by changing the input EnergyPlus Weather file (EPW) to keep boundary conditions, i.e. Dry-bulb temperature (DBT), Relative Humidity (RH), Dew-point temperature (DPT) and Global diffuse/direct solar radiation, same all throughout the year; while setting coordinates and altitude level of building such that Solar altitude angle is always same which keeps incident solar radiation constant. For this purpose, Global Direct solar radiation was kept as zero; EPW file's earth coordinates were set at the North Pole, i.e. 90°N, 0°W; and altitude level was kept at highest possible of 20,000 m such that incident solar radiation on façade system which is not influenced by reflections due to the surroundings and ground, which EP calculate automatically. Dry-bulb temperature (DBT) and Global Diffuse solar radiation were the two conditions parametrized for the simulations.

2.2.2.Input parameters

The input parameters against which the simulations were performed can be grouped in two parts. The temperature difference of inside and outside, i.e. (DBT-Inside Air Temperature) and Global Diffuse radiation value was defined as "**Boundary Conditions**". Whereas different operation modes of DSF, i.e. Airflow path, Airflow rate; and of Blinds, i.e. State of blinds, Slat angle; was defined as "**Operation modes**". Different possibilities of these parameters used in this study are listed below:

Boundary Conditions

i. Temperature difference (between indoor and outdoor) = -45, -40, -35, -30, -25, -20, -15, -10, -5, 0, 5, 10, 15, 20 [°C]
    - Outside temperature = -20, -15, -10…….35, 40 [°C]

- Inside Temperature = 20, 25 [°C]

ii. Solar radiation = 0, 200, 400, 600, 800, 1000 [W/m^2]

DSF operation modes

i. Airflow rate in cavity = 0, 0.0027, 0.0055, 0.011, 0.022, 0.044, 0.088 [m^3/s.m]
ii. Airflow direction in cavity = Supply (SU), Exhaust (EX), Indoor air curtain (IC), Outdoor air curtain (OC), Air Buffer (AB)
iii. State of Blinds = on/off
iv. Slat angle = 0, 30, 45, 60, 90, 120, 135, 150 [deg]
v. Reflectivity of Blinds = High, Low

2.3. Description of Performance metrics

Research questions answered here:
"How to quantify thermal and visual performance of DSF?"

To quantify the thermal and visual performance of a façade system there is a need to define metrics which can accurately describe the thermal gain/loss and visual gain/loss of a dynamic façade system like a DSF. First, it is important to understand the metrics/coefficient used to evaluate the performance of Insulated Glazing Units (IGUs) which will then modify to quantify the performance of DSF. For IGUs, for example, a Double/Triple glazing unit, g-value is the coefficient used to represent the thermal energy transmittance while VT/T_{vis} is the coefficient used to represent the visual light transmittance.

G-value of an IGU is defined as the ratio of total solar heat transmittance to the total solar heat incident on the surface of IGU. It can be expressed as follows:

$$g - value = \frac{(Q_{sol,SW} + Q_{sol,LW} + Q_{conv})}{Q_{incident}} \quad (10)$$

where,
$Q_{sol,SW}$ = Directly Transmitted shortwave radiation
$Q_{sol,LW}$ = Re-emitted longwave radiation from heat absorbed in glass
Q_{conv} = Convective heat exchange between glass and zone air
$Q_{incident}$ = Total incident solar radiation

The incident short-wave solar radiation (wavelength 380 to 780 nm), termed as $Q_{incident}$, is partly reflected and absorbed at the window panes. The remaining fraction of the solar radiation is being transmitted to the zone's interior, termed as $Q_{sol,SW}$. The heat transfer from the interior to the inner pane occurs via long-wave infrared radiation (wavelength 8 to 12 mm), termed as $Q_{sol,LW}$; and convection, termed as Q_{conv}. The heat energy absorbed at the inward surface of the inner pane is being delivered to the outward surface of the inner pane via heat conduction. In the inter-pane space, the heat transfer occurs via convection and longwave radiative exchange, as it is also the case between the outward surface of the outer pane and the surrounding.

Visible transmittance (VT) or T_{vis} of an IGU is defined as the ratio of the visible light (approximately 380 to 780 nanometers within the solar spectrum) entering the space through the IGU to the incident visible light. It can be expressed as follows:

$$T_{vis} = \frac{Q_{sol,SW}}{Q_{incident}} \quad (11)$$

Figure 3 and 4 describes the heat and energy flow through an IGU into a zone. While the conventional performance metrics like U-value and g-value are sufficient to represent the thermophysical behavior

of an IGU, they do not encompass the total thermophysical gain/loss of an adaptive systems like a DSF. The conventional metrics cannot be used because the thermophysical behavior of a DSF is too far from the assumptions under which these metrics can be measured or calculated, which is done for specific boundary conditions.

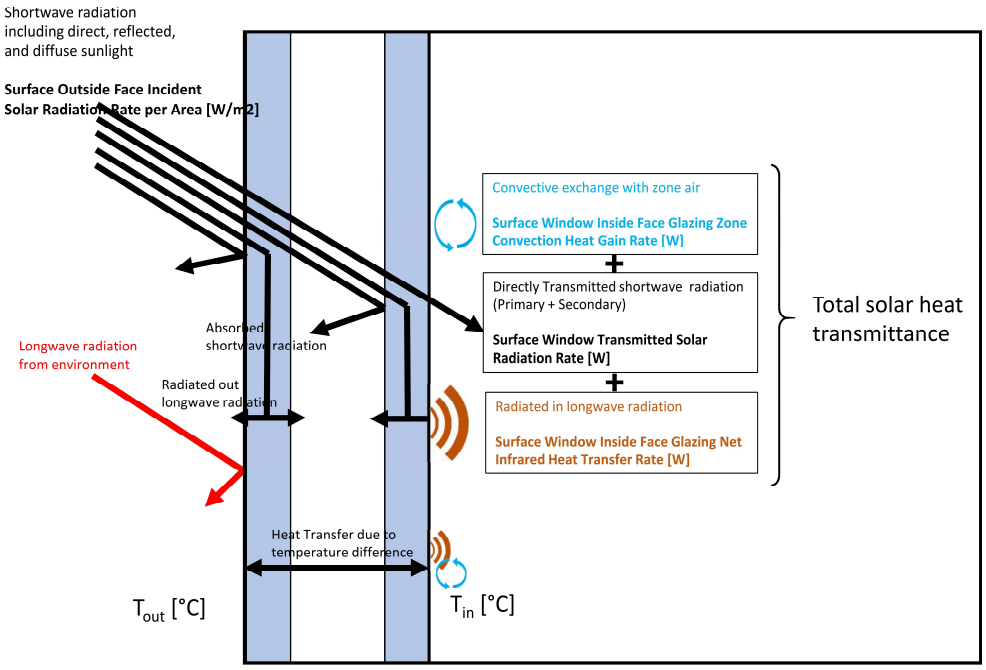


Figure 3:G-value of an insulated glazing unit is defined as ratio of total solar heat transmittance to total incident shortwave solar radiation

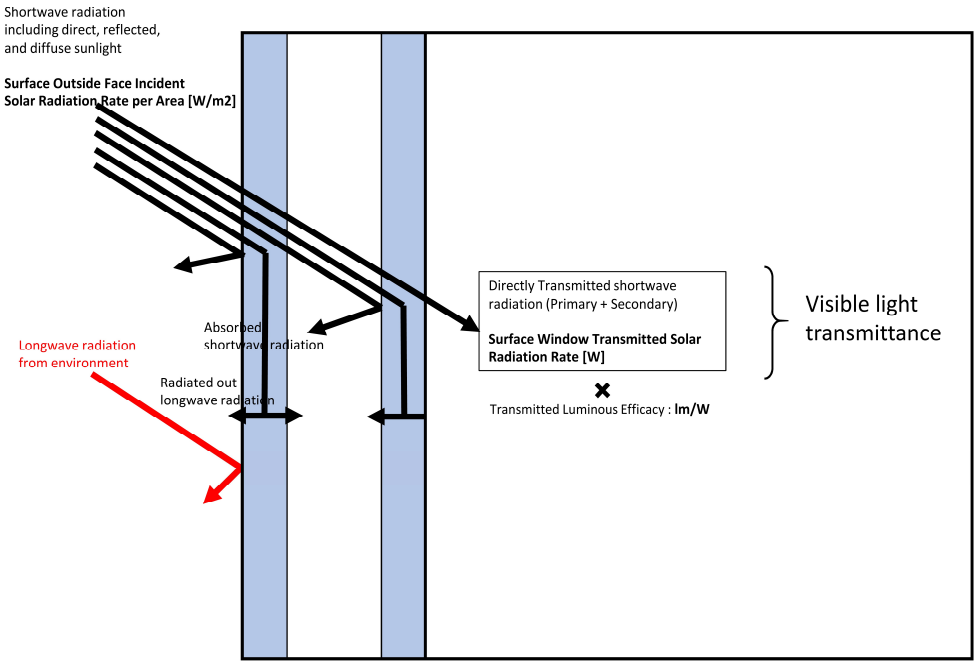


Figure 4: VT or Tvis of an insulated glazing unit is defined as ratio of total visible light transmitted to total incident visible light

2.3.1.Thermal metric

There would be two extra physical quantities for a DSF, besides the three physical quantities previously seen in g-value's schematic illustration (Fig. 3). The heat gain/loss from a DSF as the effect of airflow can expressed in two extra terms:

- Q_{air} = convective heat gain to the zone air due to the gap airflow
  This is the convective heat exchange from the airflow in the DSF cavity to the zone air as discussed in sub-section 2.2 by equation 16
- Q_{vent} = Extra energy to compensate for heating/cooling energy spent by HVAC
  This is a fictitious quantity that is not transferred through the component in terms of heat transfer but associates a heat loss/gain occurring with a mass transfer to the thermal zone. Depending on the airflow path, this amount of energy is accounted or discounted from the heat transfer through the façade in order to keep steady state conditions on the indoor side, i.e. maintain air balance and heat balance in the zone. It accounts for missing enthalpy flow which is compensated by the HVAC to treat the extra air which is either removed or added because of the airflow in DSF. This quantity can be calculated as follows:

$$Q_{vent} = \dot{m} C_p \Delta T \text{ (W)} \quad (12)$$

  where,
  C_p = heat capacity of the air (J/kg-K)
  ΔT = temperature difference of extra air brought inside the zone to compensate for airflow
  $\dot{m}$ = air mass flow rate as in equation 7

Figure 5 shows the total heat flow and types of heat gain/loss from a DSF into the zone. The metric which will quantify the thermal performance of a DSF, simply called "**Thermal metric" (W)** can be expressed as:

$$Thermal\ Metric = Q_{sol,SW} + Q_{sol,LW} + Q_{conv} + Q_{air} + Q_{vent} \text{ (W)} \quad (13)$$

This metric was normalized over the area of the façade which will make unit as W/m^2. It is to be noted that a DSF can have 5 different airflow path which will 5 different Thermal Metric which is as follows:

a. **Indoor Air Curtain**: when the airflow is from indoor to indoors. The DSF cavity is ventilated by indoor air with no connection to the outdoor air.

$$Thermal\ Metric = Q_{sol,SW} + Q_{sol,LW} + Q_{conv} + Q_{air} \text{ (W)} \quad (14)$$

   where,
   Q_{vent} = 0, as there is no air exchange happening between inside and outside.

b. **Outdoor Air Curtain**: when the airflow is from outdoor to outdoor. The DSF cavity is ventilated by outdoor air with no connection to the indoor air.

$$Thermal\ Metric = Q_{sol,SW} + Q_{sol,LW} + Q_{conv} \text{ (W)} \quad (15)$$

   where,
   $Q_{vent} = 0$, as there is no air exchange happening between inside and outside.
   $Q_{air} = 0$, as the heated/cooled air from the cavity does not interact with inside air.

c. **Air Extract**: when the airflow is from indoor to outdoor. Here, the DSF removes indoor air.

$$Thermal\ Metric = Q_{sol,SW} + Q_{sol,LW} + Q_{conv} + Q_{vent}\ \text{(W)} \quad (16)$$

where,
$Q_{air} = 0$, as the heated/cooled air from the cavity does not interact with inside air.
$Q_{vent} = \dot{m}C_p(T_{in} - T_{gap})$, where T_{in} is zone inside temperature and T_{out} is Outside Dry-Bulb temperature.

To maintain the air balance in the zone for the purpose of steady state conditions, same amount of extracted by DSF is added from outside via HVAC. Hence, Q_{vent} is the energy spent to bring the temperature of that amount of air from T_{gap} to T_{in}. To understand better, this extra air, $\dot{m}$, wouldn't be added in the zone in exact same boundary conditions if there wasn't a DSF present in the zone. If the outside temperature is colder than inside, the HVAC would have to heat the air to inside air temperature and vice versa.

d. **Air Supply**: when the airflow is from outdoor to indoor. Here, the DSF supplies air to the indoor environment.

$$Thermal\ Metric = Q_{sol,SW} + Q_{sol,LW} + Q_{conv} + Q_{air} + Q_{vent}\ \text{(W)} \quad (17)$$

where,
$Q_{vent} = \dot{m}C_p(T_{in} - T_{gap,out})$, where T_{in} is zone inside temperature and $T_{gap,out}$ is air gap outlet temperature as expressed in equation 3.

Q_{vent} is the energy spent to bring the temperature of the amount of air from $T_{gap,out}$ to T_{in}, which is supplied inside. If there was no DSF present in the façade of the zone this extra energy would not be spent by HVAC. If the $T_{gap,out}$ is more that zone temperature, HVAC would have the supplied air to lower zone air temperature and vice versa. Like Air extract mode, here also, $\dot{m}$, would be removed from the zone to maintain air balance but that energy lost would not be accounted as the energy lost or gain is happening to outside air and not inside the zone air.

e. **Air Buffer**: when there is no airflow. All the air inlets are closed and DSF acts as a buffer of air gap between two glass.

$$Thermal\ Metric = Q_{sol,SW} + Q_{sol,LW} + Q_{conv}\ \text{(W)} \quad (18)$$

where,
$Q_{vent} = 0$, as there is no air exchange happening.
$Q_{air} = 0$, as the heated/cooled air from the cavity does not interact with inside air.

2.3.2. Visual metric

Whereas, to quantify the visual performance of the DSF the metric "**Visual metric" (lm)** can be expressed as:

$$Visual\ Metric = Q_{sol,SW} \times K\ \text{(lm)} \quad (19)$$

where,
Luminous Efficacy, K = 105 lm/W
This metrics gives the luminous flux that enters the indoor space through the inside interface of the DSF. Similar to thermal metrics, visual metric was also normalized over area of the façade which will make unit of visual metric as lm/m^2

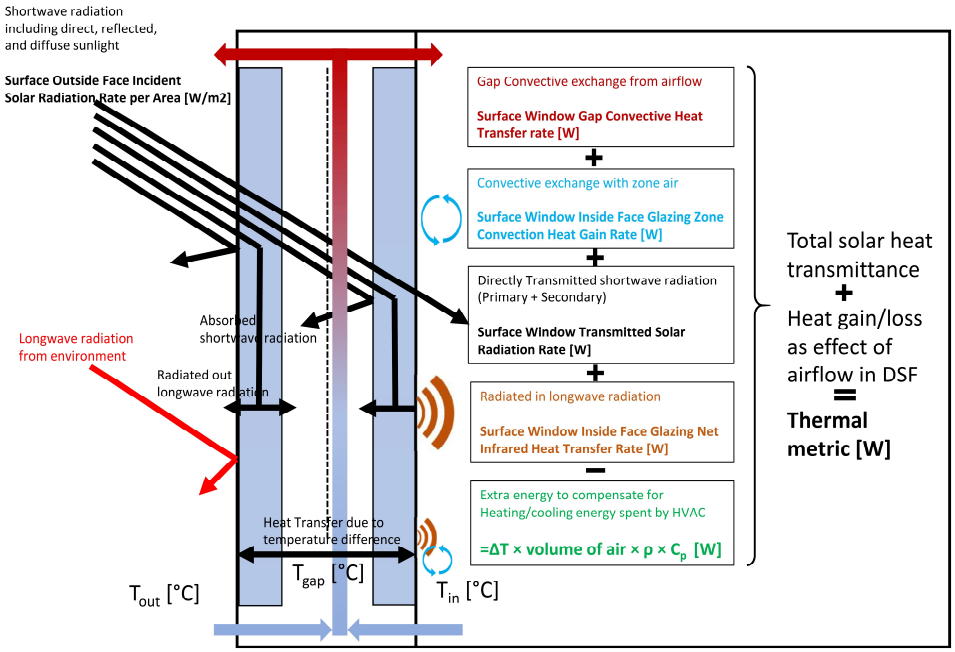


Figure 5: Heat flow schematic representation for a Double Skin Façade (DSF)

2.4. DSF configurations

2.4.1. Variations of Double Skin Façades (DSFs)

In this study, different configurations of Double Skin Façades (DSFs) were simulated to study the effect of varying air gap to varying glass type and reflectivity of shading blinds. Figure 6 shows the three different types of glass arrangement used, i.e. Double glazing – Air gap – Single glazing, Single glazing – Air gap – Double glazing and Double glazing – Air gap – Double glazing. The details of these three-glass arrangements are given in Table 1 while the thermophysical properties of various glass panes used in these arrangements are given in Table 2.

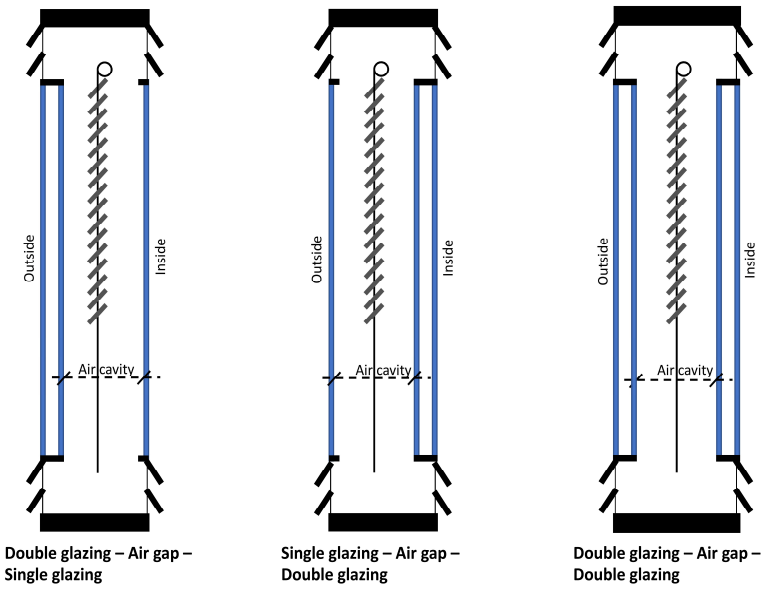


Figure 6: Three different glass arrangements of DSFs used.

Table 1: Summary of layers for three different glass arrangement, with Layer1 being outermost and Layer 7 innermost.

	Double - Airgap - Single	**Single - Airgap - Double**	**Double - Airgap - Double**
Layers 1	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, Planibel Clearlite 10 m
Layers 2	Gap, Argon 16 mm	Gap, Air 200/400/600 mm	Gap, Argon 16 mm
Layers 3	AGC Glass, Planibel Clearlite 6 mm	AGC Glass, Planibel Clearlite 6 mm	AGC Glass, Planibel Clearlite 6 mm
Layers 4	Gap, Air 200/400/600 mm	Gap, Argon 16 mm	Gap, Air 200/400/600 mm
Layers 5	AGC Glass, iplus Top1.1 on Clearlite 10 mm	AGC Glass, iplus Top1.1 on Clearlite 10 mm	AGC Glass, Planibel Clearlite 6 mm
Layers 6	-	-	Gap, Argon 16 mm
Layers 7	-	-	AGC Glass, iplus Top1.1 on Clearli 10 mm

Table 2: Thermophysical properties of types of glass panes used.

	AGC Glass, Planibel Clearlite 6 mm	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, iplus Top1. on Clearlite 10 mm
Thickness	0.006	0.010	0.010
Solar Transmittance at Normal Incidence	0.847	0.803	0.597
Front Side Solar Reflectance at Normal Incidence	0.075	0.071	0.284
Back Side Solar Reflectance at Normal Incidence	0.075	0.071	0.206
Visible Transmittance at Normal Incidence	0.895	0.879	0.885
Front Side Visible Reflectance at Normal Incidence	0.080	0.078	0.046
Back Side Visible Reflectance at Normal Incidence	0.080	0.078	0.053
Infrared Transmittance at Normal Incidence	0.000	0.000	0.000
Front Side Infrared Hemispherical Emissivity	0.840	0.840	0.038
Back Side Infrared Hemispherical Emissivity	0.840	0.840	0.840
Conductivity	1.000	1.000	1.000

The air gap in all three-glass arrangement is varied as 200mm, 400mm and 600mm, whereas the reflectivity of blinds which is inside the air gap is varied to High and Low, i.e. reflectance is 0.8 and 0.1 respectively while keeping emissivity of the slat same as 0.9 in both. This makes the number of types of DSFs to 18 which have been summarized in Table 3.

Table 3: Summary of different configurations of DSF, for example, "DSF_1" is Double - 200 mm Air – Single with High reflective blinds

	Double - Airgap - Single	**Single - Airgap - Double**	**Double - Airgap - Double**
200 mm air gap	DSF_1 Double - 200 mm Air - Single High reflectivity blinds	DSF_3 Single - 200 mm Air - Double High reflectivity blinds	DSF_5 Double - 200 mm Air - Double High reflectivity blinds
200 mm air gap	DSF_2 Double - 200 mm Air - Single Low reflectivity blinds	DSF_4 Single - 200 mm Air - Double Low reflectivity blinds	DSF_6 Double - 200 mm Air - Double Low reflectivity blinds
400 mm air gap	DSF_7 Double - 400 mm Air - Single High reflectivity blinds	DSF_9 Single - 400 mm Air - Double High reflectivity blinds	DSF_11 Double - 400 mm Air - Double High reflectivity blinds
400 mm air gap	DSF_8 Double - 400 mm Air - Single Low reflectivity blinds	DSF_10 Single - 400 mm Air - Double Low reflectivity blinds	DSF_12 Double - 400 mm Air - Double Low reflectivity blinds
600 mm air gap	DSF_13 Double - 600 mm Air - Single High reflectivity blinds	DSF_15 Single - 600 mm Air - Double High reflectivity blinds	DSF_17 Double - 600 mm Air - Double High reflectivity blinds
600 mm air gap	DSF_14 Double - 600 mm Air - Single Low reflectivity blinds	DSF_16 Single - 600 mm Air - Double Low reflectivity blinds	DSF_18 Double - 600 mm Air - Double Low reflectivity blinds

2.4.2. Benchmark glazings

To compare the performance of various DSF variations and test them, four different types of Insulated glazing units (IGUs) were also simulated under similar conditions while similar performance metrics were compared. Figure 7 summarizes the 4 different IGUs studied, while Table 4 list the layer arrangements. Similar glass panes were used for IGUs as in DSFs which have been listed in Table 2.

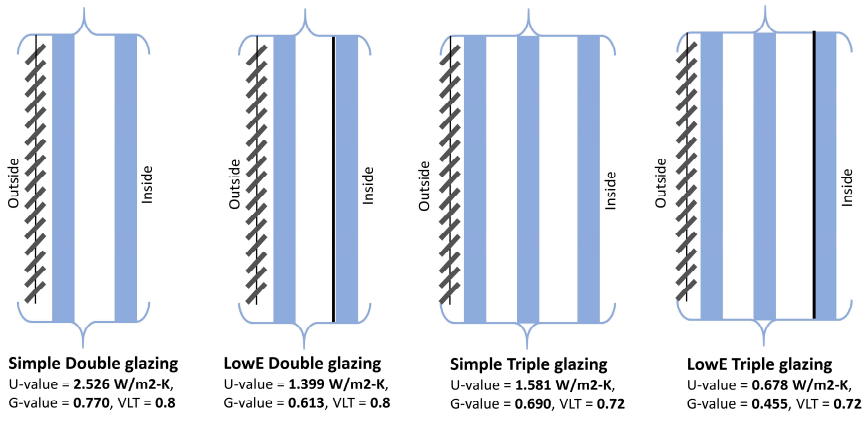


Figure 7: Four different benchmark IGUs studied.

Table 4: Summary of layers for four different IGUs, with Layer1 being outermost and Layer 4 innermost.

	Simple Double glazing	LowE Double glazing	Simple Triple glazing	LowE Triple glazir
Layers 1	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, Planibe Clearlite 10 mm
Layers 2	Gap, Argon 16 mm	Gap, Argon 16 mm	Gap, Argon 16 mm	Gap, Argon 16 mn
Layers 3	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, iplus Top1.1 on Clearlite	AGC Glass, Planibel Clearlite 6 mm	AGC Glass, Planibe Clearlite 6 mm
Layers 4	-	-	Gap, Argon 16 mm	Gap, Argon 16 mn
Layers 5	-	-	AGC Glass, Planibel Clearlite 10 mm	AGC Glass, iplus Top1.1 on Clearlit

2.5. Data post-processing scheme

As seen in sub-sections before, this study consists A total of more than 600 000 simulations where each simulation has 5 different values as inputs (operation modes and boundary conditions) and two outputs (thermal and visual metrics). Given the high number of input and outputs, in order to summarize and synthetize this data into useful information. the data was post-processed with an increased level of synthesis and aggregation. Three levels of post-processing are conceived with three fundamentally different and consequential aims (further detail will be given in the next paragraphs):

1. Level 1: general trends of how input parameters changes w.r.t. thermal and visual gain in simulation cases.
2. Level 2: qualitative analysis of the effect of varying boundary condition on performance
3. Level 3: analysis of how performance range and possibility of decoupling thermal and visual aspects changes

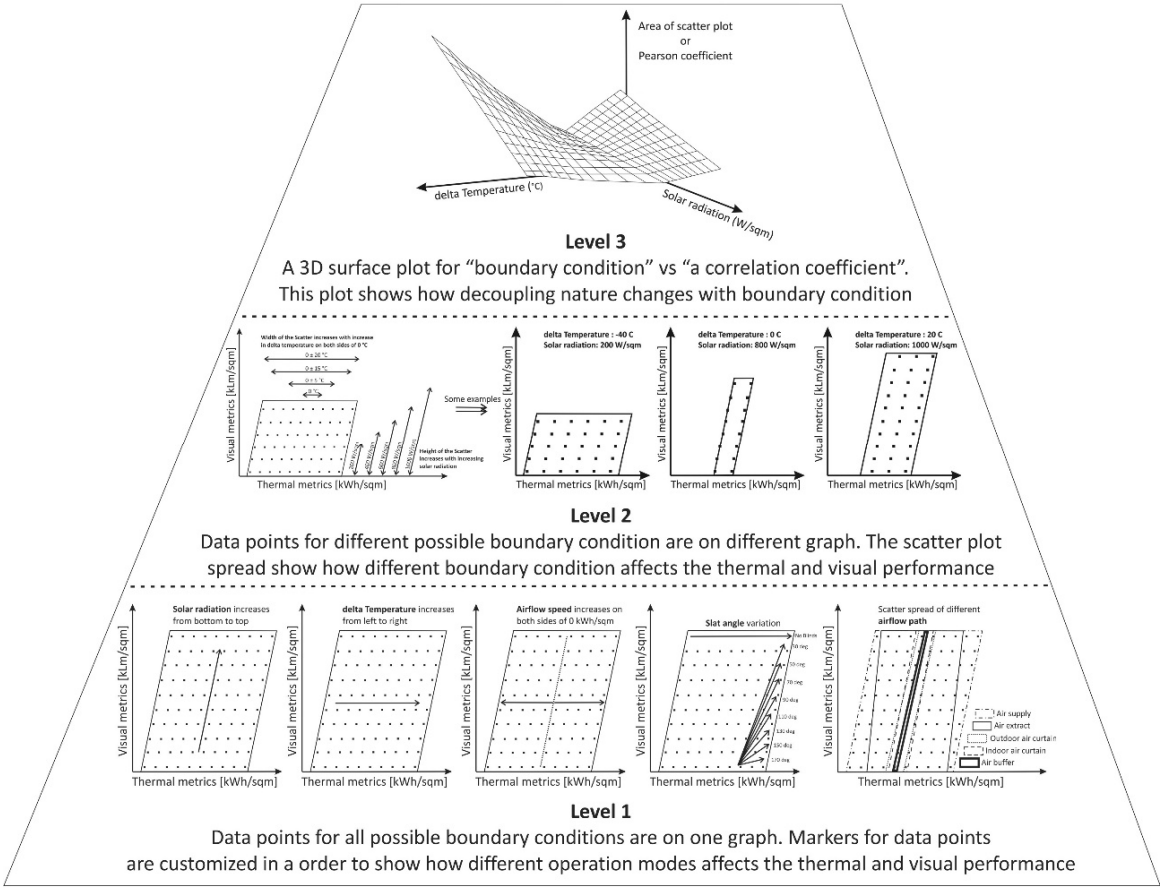


Figure 8: Data presentation and post-processing scheme

A scheme for data presentation was devised where for every other level, the type of information shown in each graph changes while reducing the number of data points on the graphs. The data presentation can be distributed in 3 levels as shown in Figure 8, discussed in detail below.

2.5.1.Level 1

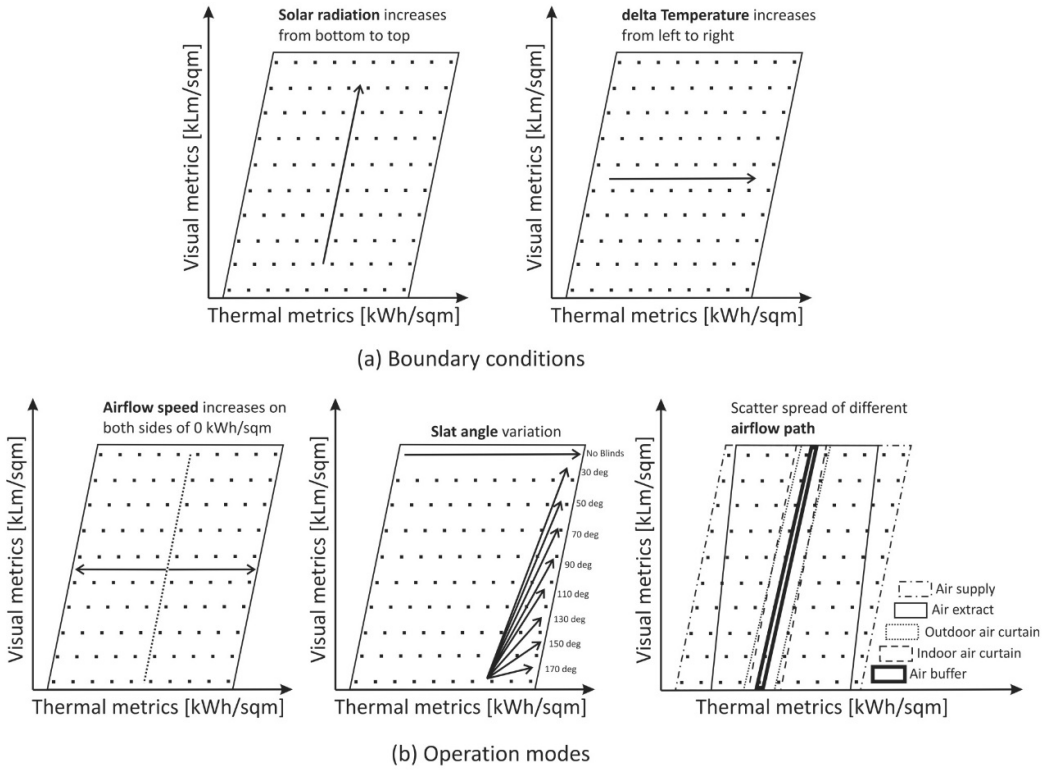


Figure 9: These illustrations show how trend of how (a) boundary conditions, and (b)operation modes parameters effects the thermal and visual performance

In the first level, the aim was to show how a change in every input, i.e. boundary condition and operation mode, changes the thermal and visual performance. An individual scatter plot was made for every DSF, as in Table 3, plotting every simulation case's performance metrics such that the *Thermal Metric* value and Visual Metric value of that case was on X and Y-axis respectively. To show the variation in inputs, the data point's marker size, opacity and color was varied which made it simple to understand. These trends have been shown in Figure 9 and discussed in detail below:

a) **Solar radiation – Size of data point marker**: It was seen in the scatter plot that solar radiation value for which each data point represents increased from bottom to down, i.e. the data points on the bottom-most part of the scatter spread represented simulation cases with lowest solar radiation value and upper-most data points represented simulation cases with high solar radiation. This trend also makes sense that visual gain, represented by visual metric on y-axis, increases when incident solar radiation increases. In all the scatter plots shown in this level, the marker size represents this change. The size of the markers used to represent the data point increased with increasing solar radiation value.

b) **Temperature difference (between indoor and outdoor)– Opacity of data point marker**: For the temperature difference value each data point represented it was seen that it increases from left to right of the scatter spread of points. The leftmost data points on the graph represented simulation cases which had -45 °C as temperature difference while rightmost represented cases with +20 °C. This trend was shown as the opacity of the marker, with most translucent marker representing -45 °C and most opaque marker representing +20 °C.

To maintain the clarity of the scatter plots, the markers were customized to only size and opacity as shown above while color was represented by 3 different inputs in 3 different graphs respectively. Hence all the DSF were represented in total 3 scatter plots where color of marker was represented by airflow speed, airflow rate and slat angle respectively, which are explained below.

c) **Airflow speed – Color of data point marker**: It was seen that airflow speed each data point represented increased on both side of 0 kWh/sqm, i.e. thermal metric. The airflow speed which data points represented near Thermal metric = 0 kWh/sqm, were for lowest airflow speed while as data points go left or right from 0 kWh/sqm, the airflow speed increased. To represent the 7 different airflow speed, 7 different colored markers were used, while the size and opacity of the marker changed as shown above.

d) **Airflow path – Color of data point marker**: As mentioned in sub-section before, a DSF has 5 minds of airflow paths. The spread of data points representing different airflow paths is interesting to study how these airflow paths/modes can be used during the operation. To represent the 5 different airflow paths/modes, 5 different colored markers were used, while the size and opacity of the marker changed as shown above.

e) **Slat angle of blinds – Color of data point marker:** It was seen that higher the angle of a slat of blinds is lower it is in the data points spread, while the simulation cases with no blinds are on the topmost location of the data point spread. To represent the 7 different slat angle and 1 case for no blinds, 8 different colored markers were used, while the size and opacity of the marker changed as shown above.

The methodology to present data shown in this level can be used for both a DSF and an IGU which can be used for qualitative comparative analysis.

2.5.2.Level 2

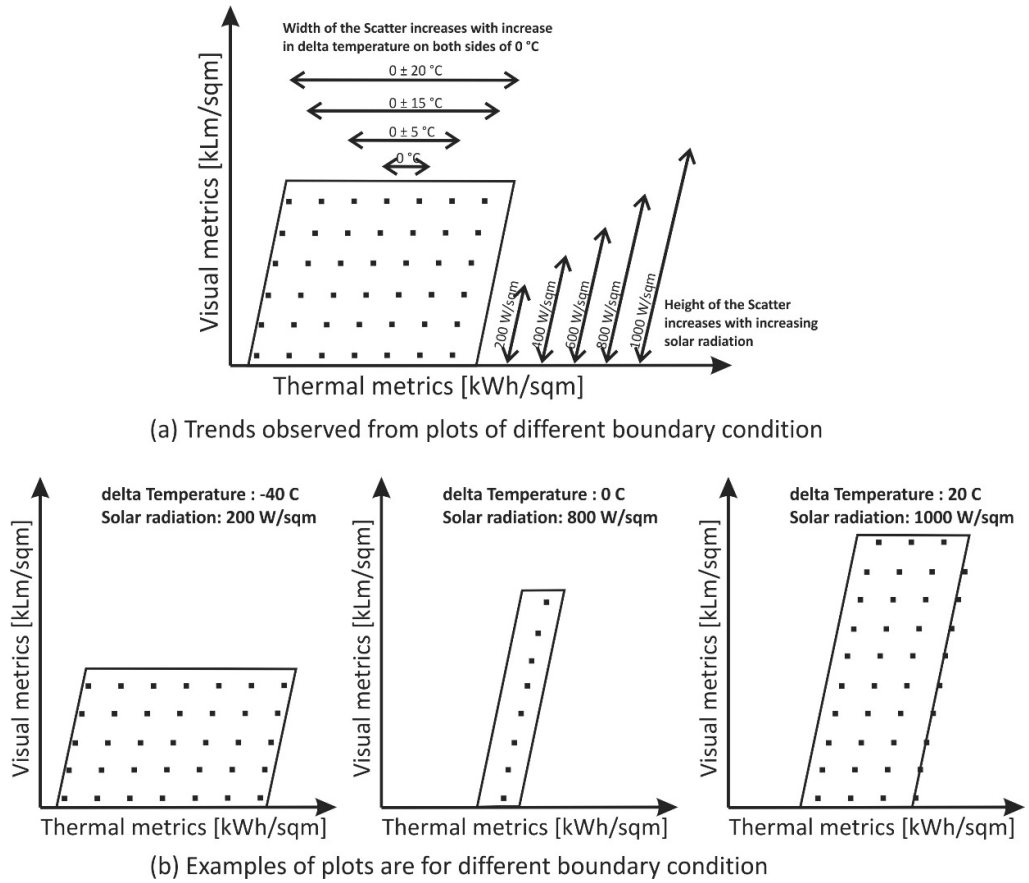


Figure 10: Illustrations showing (a) trends observed from different plots (b) some examples of how these plots look

The aim here is to understand how big the range of thermal and visual performance of a DSF is under each boundary conditions. This was done by plotting together all the possible operational modes that a DSF can have at a particular boundary condition. The simulation cases made had 7 cases of temperature difference (between indoor and outdoor) ranging from -45 °C to +20 °C whereas 6 cases of incident solar radiation, i.e. 0, 200, 400, 600, 800 and 1000 W/sqm which made a total of 84 combinations. While the previous level scatter plots presented the data points of all boundary conditions on one plot, here 84 different plots were made. Using a Python script, programmatically a polygon was made around the scatter plot for every case which represented the boundary of how much varied thermal and visual performance of a DSF is for a particular boundary condition. All the 84 plots had data point markers customized in the same way as done in scatter plots in level 1. From these plots following observations were made, which have also been described in Figure 10:

a) **Width of the polygon**: The width of the polygon, which is the boundary of scatter plot, represented the temperature difference component of the boundary condition for which the specific plot was made. The lowest width was seen for temperature difference was equal to 0 °C. The width of the polygon increases with the increase on both sides of 0, i.e. it was more when the temperature difference was -45 °C and +20 °C.

b) **Height of the polygon**: On the other side, the height of the polygon represented the incident solar radiation component of the boundary condition for which plot was made. The height of polygon increased with increase in solar radiation.

Same as Level 1, the methodology to present data shown in this level can also be used for both a DSF and an IGU which can be used for qualitative comparative analysis. At any particular boundary condition, the performance range of a DSF can be seen visualized as a polygon, whereas for an IGU the performance range can be visualized only as a line.

2.5.3.Level 3

Research questions answered here:
"How to quantify thermal and visual performance of DSF?"

While the aim of the previous level was to qualitatively assess the performance of a DSF at different boundary condition, here the assessment is done quantitively. Another aim of this level is to reduce the 84 plots seen before to just one which would represent the meaning of all the data in 84 plots.

Here, a Python script was used to iteratively calculate the area of the polygon which scatter plot made for every boundary condition and calculate the Pearson correlation coefficient of the data points for every boundary condition. These 2-correlation metrics were calculated in the following manner:

a) **Area of the Polygon**: During each iteration of boundary conditions, the Python script was finding the 4 data points which would represent the 4 vertices of the polygon. These 4 data points are the maximum and minimum points on the scatter spread. The area of this polygon was determined from the following formula:

$$area = \left|\frac{(x_1y_2 - x_2y_1)+(x_2y_3 - x_3y_2)+(x_3y_4 - x_4y_3)+(x_4y_1 - x_1y_4)}{2}\right| \quad (20)$$

b) **Pearson correlation coefficient:** A Pearson correlation is a number between -1 and 1 that indicates the extent to which two variables are linearly related, in this case, Thermal and Visual metrics. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable. A value less than 0 indicates a negative association; that is, as the value of one variable increases, the value of the other variable decreases. This is shown in Figure 11. During each iteration of boundary conditions, the Python script was copying the thermal metric and visual metric in separate arrays. These 2 arrays were used to find the Pearson correlation coefficient for the scatter spread for a particular boundary condition. The formula used in the Python scripts for computing the Pearson, r, is as follows:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{(X_i - \bar{X})^2}\sqrt{(Y_i - \bar{Y})^2}} \quad (21)$$

where,
X_i = every data point in *Thermal Metric* array
Y_i = every data point in *Visual Metric* array
$\bar{X}$= mean of *Thermal Metric* array
$\bar{Y}$ = mean of *Visual Metric* array

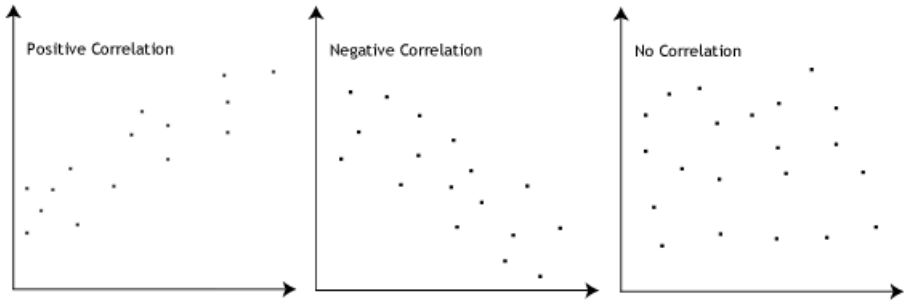


Figure 11: Examples of scatter plots with different correlation

The above 2 correlation metrics were plotted with temperature difference (between indoor and outdoor) and incident solar radiation as three different 3D surface plot. This 3D surface plot shows the trend of how the particular correlation metric varied with varying boundary conditions. An example of how this 3D surface plot would look is shown in Figure 12. The significance of these correlation metrics with respect to boundary condition is discussed in the next chapter where 3D plots are shown for different DSFs.

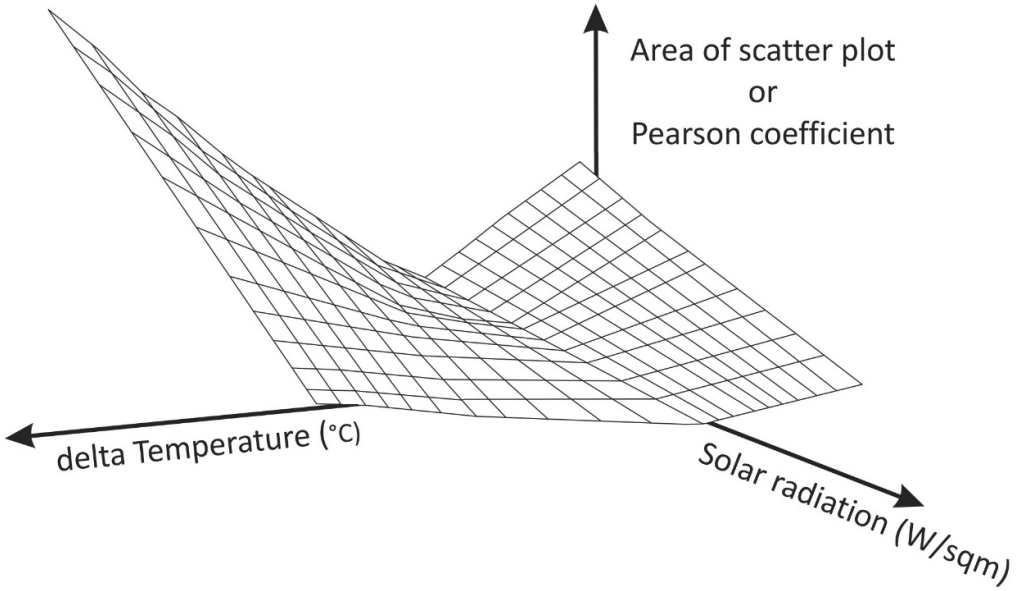


Figure 12: Example of how the 3D surface plot between boundary condition and a correlation metric would look

While it was possible to use methodology of both level 1 and 2 to visualize data for both a DSF and an IGU, it is not possible for this level. As mentioned in previous sub-section, the performance range of an IGU at a particular boundary condition is a line, it would translate to area as zero and value of pearson, *r*, as 1, which would be same for all boundary conditions and hence incomparable.

3. Results and discussion

3.1. Performance range of Insulated Glazing Units (IGUs)

Before looking at the results of how a DSF perform, this subsection presents how different kinds of IGUs perform under different boundary condition and operation modes in case there is a shading device in use. The performance metrics used here are as same as that would be used for investigating the performance of DSFs which will allow us to benchmark the results and get a feel of how the new performance metrics proposed in this study look like against metrics generally used, i.e. T_{vis} and g-value.

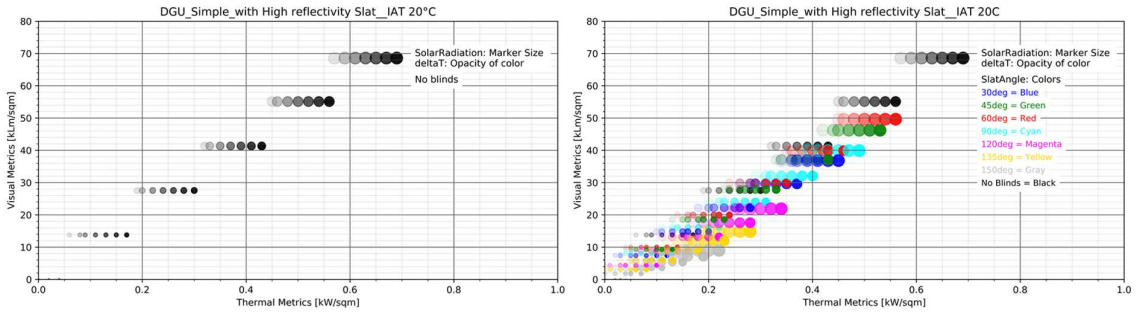
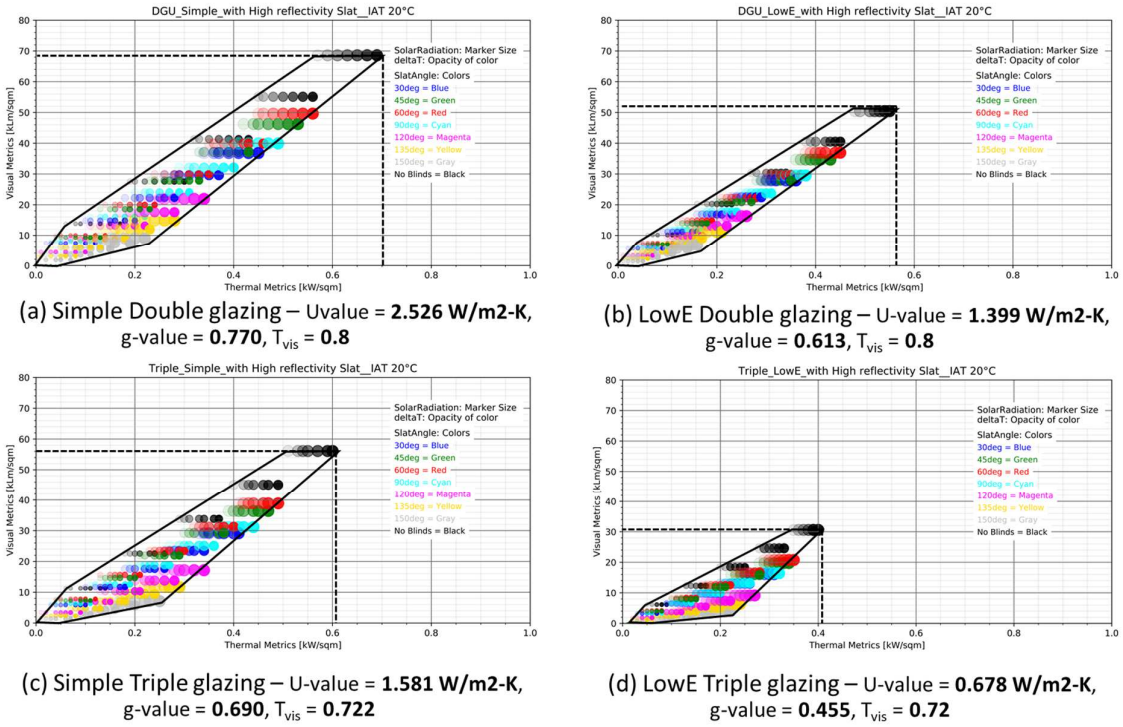


Figure 13: Scatter plot for Visual metric on Y axis against Thermal metric on X axis for a Simple Double-Glazing Unit. Figure (a) show how it perform at various boundary condition without any shading system, while Figure (b) shows the performance with shading sys system installed with different representing different slat angle of the blinds

Figure 13 represents the performance of a Simple Double-Glazing Unit, which is the first IGU as shown in Figure 7. Figure 13 (a) shows how the IGU perform where there are no blinds installed. The increase in solar radiation is represented by the increase in marker size whereas the increase in the temperature difference (between indoor and outdoor) is represented by the increasing opacity of the markers. A similar trend but with reduced range can be seen in Figure 13(b) showing the performance of IGU when blinds are used with different slat angles of slat representing different colors of the marker. For higher angles, i.e. near 180 deg, the solar radiation penetration inside the zones is less because of over lapping surface of the slats in the path of radiation. This reduces the visual gain for cases when higher slat angles are used.

If the performance of different types of IGUs is compared, as shown in Figure 14, it can be clearly seen that a Simple DGU has the highest range of visual gain and thermal gain ranging till 69 kLm/sqm and 0.7 kW/sqm respectively. The performance range decreases when the U-value of the IGUs decrease. This trend makes sense as U-value is the metric defining how much heat transfer can happen through the façade system. Whereas in all the cases it can be clearly observed that there is a positive correlation seen between Visual and Thermal metric, meaning if one increase other value also increases. This trend means that without using a shading system there is no possibility to control the two performance aspects, while using a shading system allows to regulate the visual and thermal performance but not separately.



(a) Simple Double glazing – Uvalue = **2.526 W/m2-K**, g-value = **0.770**, T_{vis} = **0.8**

(b) LowE Double glazing – U-value = **1.399 W/m2-K**, g-value = **0.613**, T_{vis} = **0.8**

(c) Simple Triple glazing – U-value = **1.581 W/m2-K**, g-value = **0.690**, T_{vis} = **0.722**

(d) LowE Triple glazing – U-value = **0.678 W/m2-K**, g-value = **0.455**, T_{vis} = **0.72**

Figure 14: Performance comparison of different types of IGUs

3.2. Performance range of Double Skin Facades (DSFs)

3.2.1. General trends and dependency of operation modes

Research questions answered here:
"How big is the thermal and visual performance range of a DSF?"
"How does different operation modes affect the different performance of a DSF?"

In this sub-section, DSF with configuration Single-200mm airgap-Double with high reflective blinds, represented by DSF_3 in Table 3. Figure 15 is the scatter plot of Visual metric against thermal metric for all the possible simulation cases for DSF_2, except the cases with indoor air temperature (IAT) 20 °C to improve clarity, as there was not much difference in comparison to cases with IAT 25 °C. In this scatter plot all the markers have been programmatically customized to represent increasing solar radiation values they represent with increasing size and increasing temperature difference (between indoor and outdoor) they represent with increasing opacity of marker. This means, all cases which represent simulation cases with solar radiation, 1000 kWh/sqm, and temperature difference, -40 °C, are on the top left whereas simulation cases with solar radiation, 0 kWh/sqm, and temperature difference 20 °C, are on the bottom right. The negative values of Thermal metric mean that the DSF is extracting that certain amount of heat from the zone air, whereas positive value would mean that the DSF is adding that much amount of heat to the zone air. The visual gain ranges from 0 to 38 kLm/sqm whereas thermal gain ranges from -3.8 kW/sqm to 1.8 kW/sqm. Figure 16 illustrates the difference in the performance range of a DSF when compared to an IGU which has the same number of glass panes and shading system.

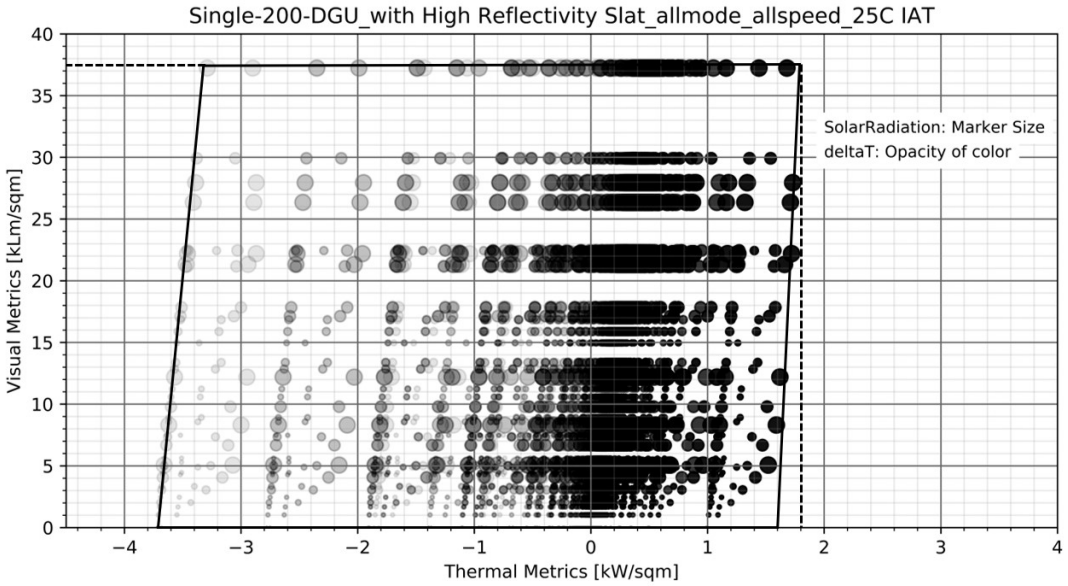


Figure 15: Scatter plot for Visual metric against Thermal metric for a DSF_2, Single-200mm airgap-Double with high reflective blinds

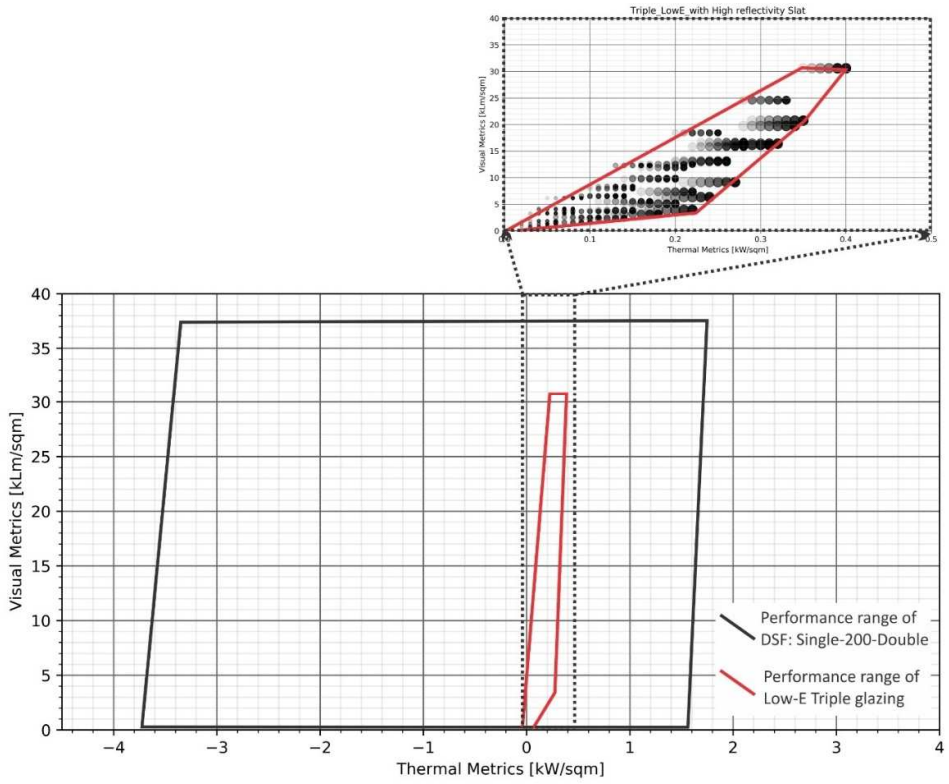


Figure 16: Illustration showing how big and wide is the performance range of a DSF compared to IGU for all boundary conditions. Both DSF and IGU has same 3 glass panes and shading system. DSF here is Single-200-Double and IGU is Low-e Triple glazing

While the opacity and size of the markers in the scatter plot have been customized to represent a change in boundary conditions, the color of the marker is characterized to show a change in different operation modes individually. To maintain clarity in the graphs, three different plots are made with changing color to represent change in operation modes.

The scatter plot of Figure 15 was taken and colors of all the markers were customized to represent the simulation cases with different airflow paths. As can be seen, Air Extract mode was represented by Blue markers, Air Supply mode by Cyan, Outdoor Air Curtain mode by Red, Indoor Air Curtain by green and Air Buffer mode by Black color marker. It can be seen that Air Extract and Air Supply mode have the largest range of thermal gain whereas the remaining three modes had very less. This is evident from the fact that Air Supply and Air Extract mode have an added component of heat gain/loss because of airflow which is not seen in other modes. The scatter plot with different colors for the Airflow path and individual sub-plots for different paths is shown in Figure 17.

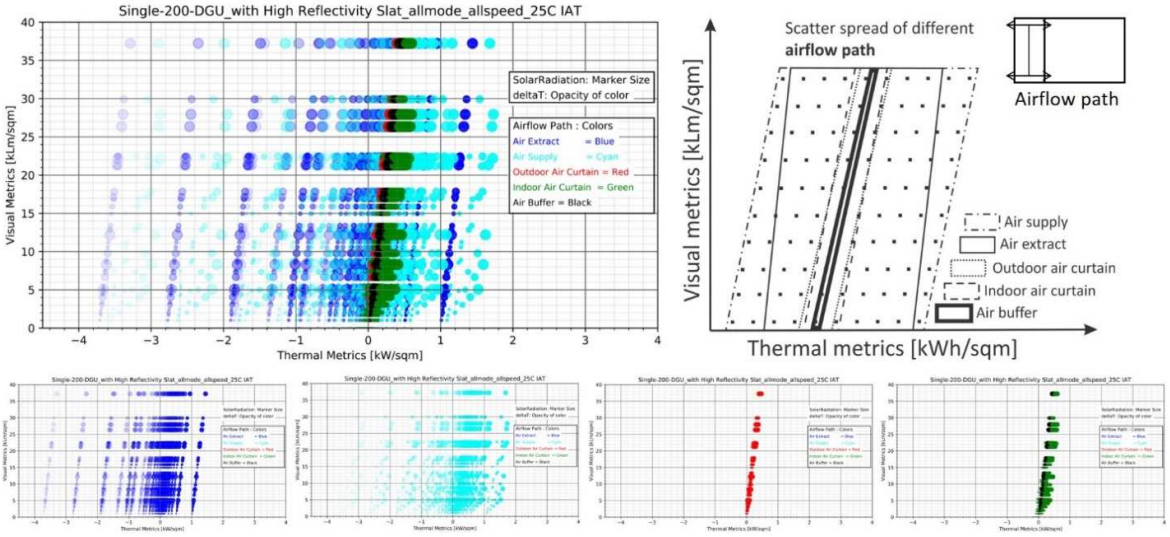


Figure 17: In the scatter plot shown in figure 15, color component was given to the markers for the data points to represent simulation cases with different Airflow Path

The similar customization was again done to Figure 15 separately, where color was added to markers to represent the simulation cases with different Airflow speed. The simulation cases with highest Airflow Speed of 0.0889 m^3/s.m was represented by Blue color whereas lowest Airflow Speed of 0.0028 m^3/s.m was represented by Yellow color markers. For Airflow Speed of 0 m^3/s.m, which represent same cases in Figure 3.4 as Air Buffer are here also represented by black color. With less airflow speed, the air in the cavity is able to spend more time in the cavity itself and hence gather more heat from the incident solar radiation whereas at the same temperature high airspeed cannot. According to the temperature difference (between indoor and outdoor) and amount of heat transfer requirement, the speed and airflow path can be regulated. The scatter plot with different colors for Airflow speed and individual sub-plots for different airflow speed is shown in Figure 18.

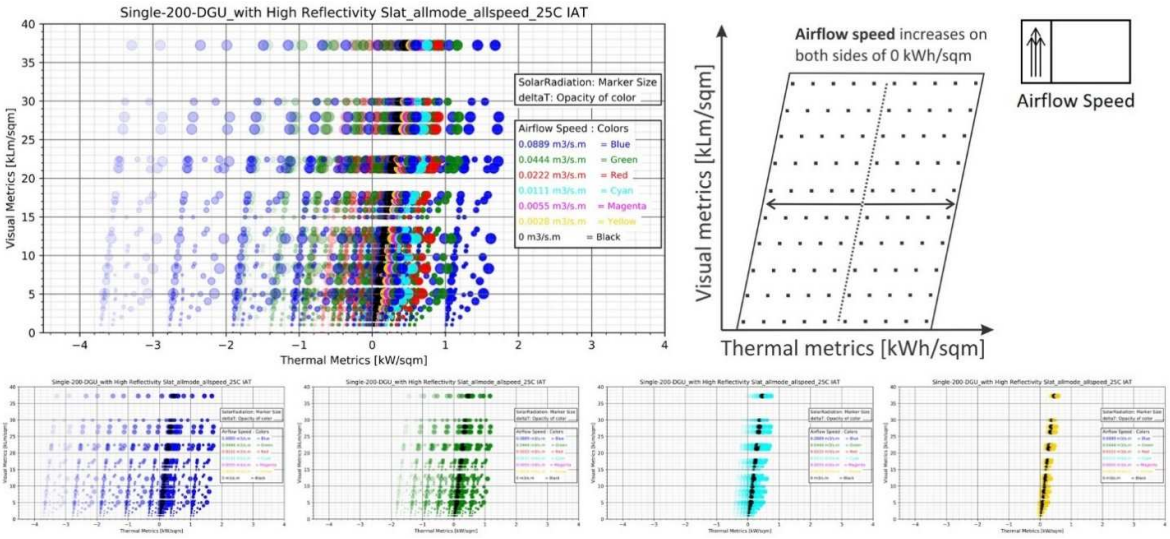


Figure 18: In the scatter plot shown in figure 15, color component was given to the markers for the data points to represent simulation cases with different Airflow Speed

To investigate and visualize how the simulation cases with blinds us or no blinds use are represented in the overall performance, Figure 15 was colored with changing color representing different slat angles, including black markers when no blinds was. The trends observed here and the explanations for them is the same as seen in the performance range of IGUs shown in sub-section 13. The scatter plot with different colors for different slat angle and individual sub-plots for different angles is shown in Figure 19.

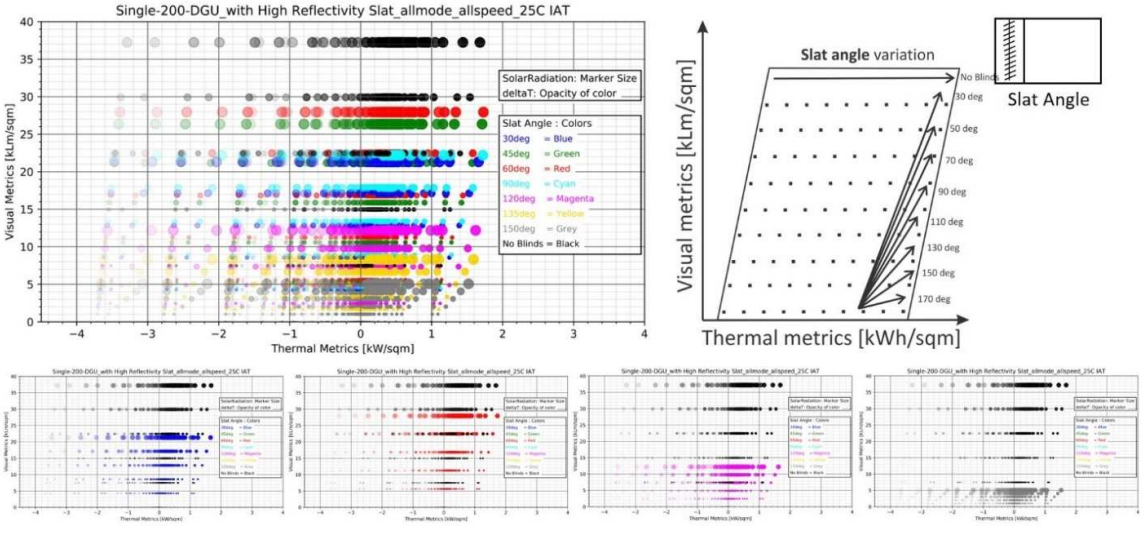


Figure 19: In the scatter plot shown in figure 15, color component was given to the markers for the data points to represent simulation cases with different simulation cases with no blinds and if blinds present then with different slat angles

3.2.2.Effect of reflectivity of blinds in DSFs on performance range

In this sub-section, the overall performance range of DSF_3, i.e. Single-200mm airgap-Double with high reflective blinds, is compared with DSF_4, i.e. Single-200mm airgap-Double with low reflective blinds, to investigate the effects of high reflective-low absorption slats vs low reflective-high absorption slats. As it can be clearly seen in Figure 20, that by using high reflective blinds there is a

more visual gain that low reflective blinds whereas low reflective blinds have more positive thermal gain seen.

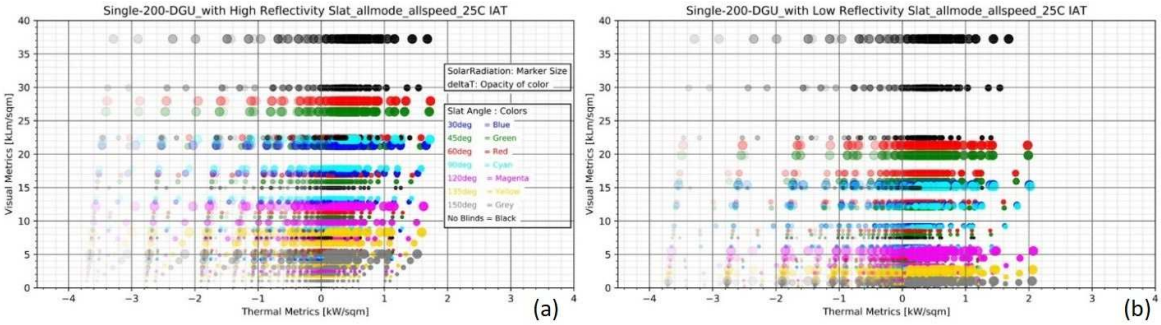


Figure 20: Performance range comparison for DSF with different reflectivity blinds: (a) Single-200mm airgap-Double with high reflective blinds vs (b) Single-200mm airgap-Double with low reflective blinds

This trend seen is due to the fact that high reflective slat absorbs less incident solar radiation and reflect more into the zone and hence more visual gain where lest reflective slat absorbs more solar radiation which is radiated into zone air in form of longwave radiation and hence a bit more positive thermal gain than high reflective blinds. This trend is seen in all different types of DSF configurations.

3.2.3. Effect of different glass configurations in DSFs on performance range

While the effect of different reflectivity of shading blinds in DSF with same glass configuration has been discussed above, the aim of this section is to investigate the performance of DSF with different glass configurations and the reason behind. Here the reflectivity of blinds and air cavity depth is kept same as high reflective and 200 mm respectively. In Figure 21 it can be seen that DSF with configuration Double-200mm-Double represented in (c) has the lowest visual gain range while the thermal range is the same as all other cases. The reason behind this is that there are in total of 4 glasses which incident solar radiation has to cross to reach the zone. With incident solar radiation being partly reflected, absorbed and transmitted at every glass boundary, the amount of solar radiation transmitted in visible range throughout the 4 glass panes is less than other two cases where there are 3 glass panes in total.

Whereas for the same number of glass panes, i.e. 3, for the other two cases DSF with configuration Double-200mm-Single represented in (a) has lower than DSF with configuration Single-200mm-Double represented in (b). The reason behind this is the placement of the Low-E glass and the shading devices in the sequence order of 3 panes and the internal optical reflections between the glass panes, blinds and the cavity. The low-e glass is the 2nd glass from outside in the double glazing, which means in case (a) the 2nd glass from outside is low-e whereas in case (b) it is the 3rd glass, i.e. the last glass, from outside. The low-e or low-emissivity coating minimize the amount of infrared and ultraviolet light that comes through your glass also causing to absorbs a bit more of infrared than other glass panes without low-e coating. A bit higher temperature of glass with low-e coating causes a small optical loss (in the visible range). This same happens with different placement of blinds also which leads to slightly different optical properties of the whole DSF. This leads to case (a) receiving less transmitted solar radiation; with more difference in visual gain being seen in higher incident solar radiation.

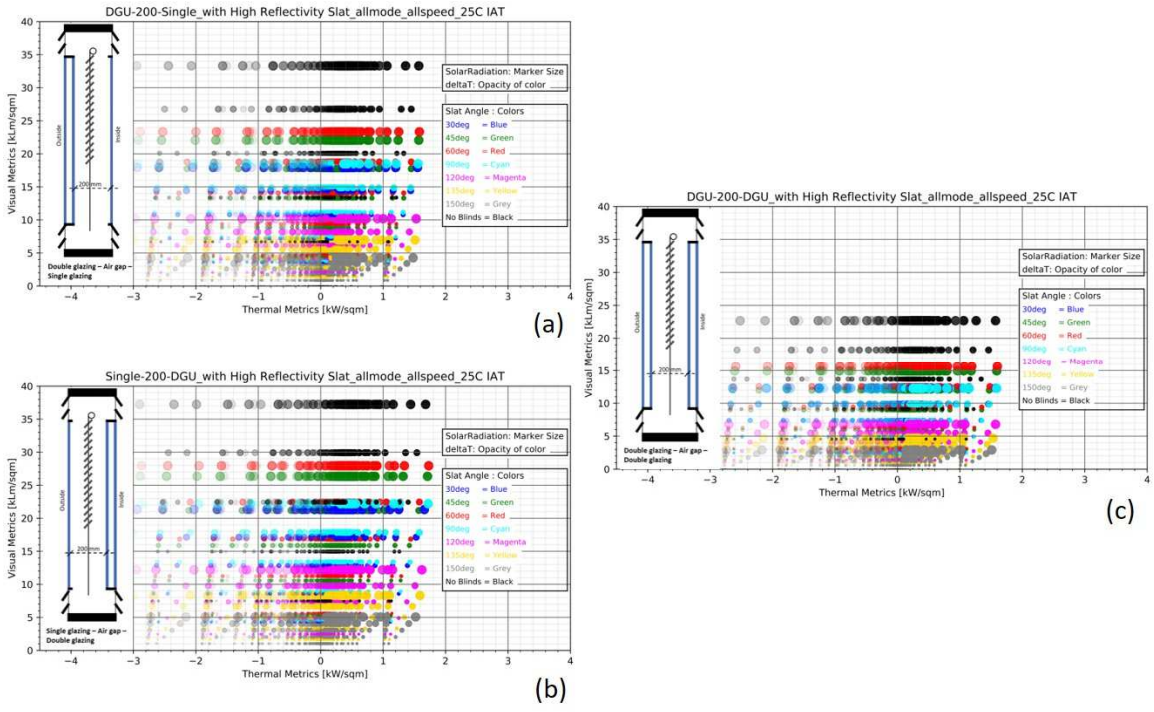


Figure 21: Performance range comparison for DSF with different arrangement: (a) Double-200mm airgap-Single with high reflective blinds; (b) Single-200mm airgap-Double with high reflective blinds; and (c) Double-200mm airgap-Double with high reflective blinds

3.2.4.Effect of different air cavity depth in DSFs on performance range

While there was a definite change seen in the overall performance range of DSFs when the arrangement of glass panes was changed, or different reflective blinds stats were used; there was little to no change is seen when air gap depth was changed in a DSF, as shown in Figure 22.

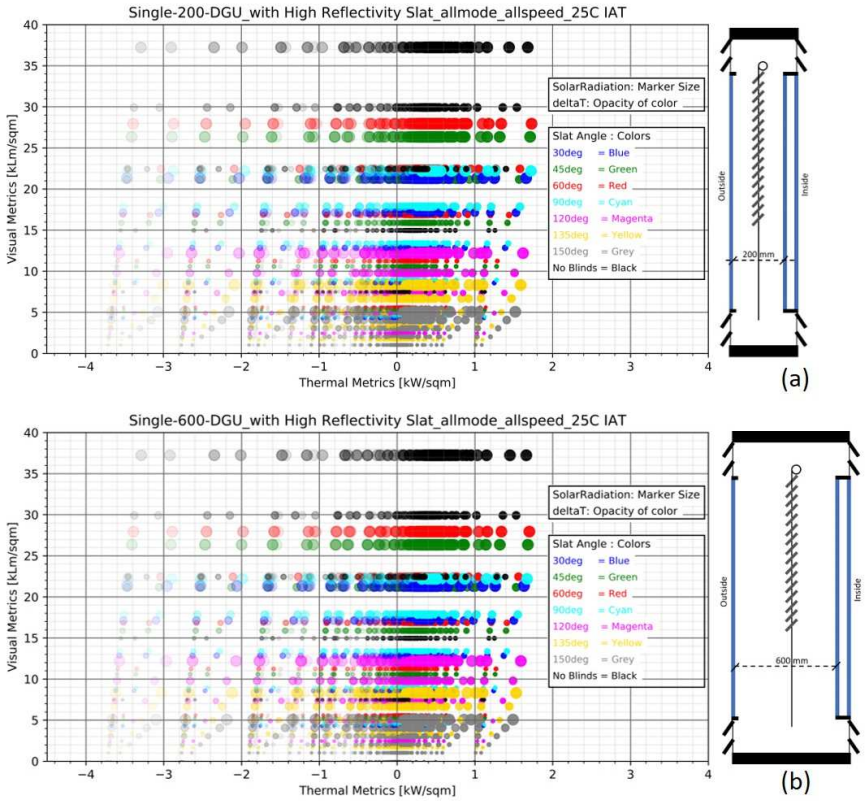


Figure 22: Performance range comparison for DSF with same glass and blinds arrangement but different air gap: (a) Single-200mm airgap-Double with high reflective blinds; (b) Single-600mm airgap-Double with high reflective blinds

As discussed in sub-section 2.2.1, The "Airflow Windows" component in EnergyPlus uses a simplified mode of heat transfer. The convective heat transfer coefficient for heat transfer inside faces of glass in cavity to cavity air is calculated as follows:

$$h_{cv} = 2h_c + 4v \tag{2.2.1.1}$$

where,
h_{cv} = convective heat transfer coefficient from glass to gap air (W/m$_2$K)
h_c = glass-to-glass heat transfer coefficient for non-vented (closed) cavity (W/m$_2$K)
v = mean air velocity in the gap (m/s)

The air velocity is determined by the gap cross-sectional area and air flow rate which is the user input value in the IDF file:

$$v = \frac{F}{A_{gap}} \text{ (m/s)} \tag{2.2.1.2}$$

where,
F = airflow rate (m^3/s)
A_{gap}= gap cross-sectional area (m^2)

According to these equations if A_{gap} is increased, i.e. increasing the air cavity depth, v should increase and hence h_{cv} should also increase leading to an increase in heat transfer. Whereas, when the air gap depth is increased from 200mm to 600mm which is also on a higher level of realistic air gaps in DSFs, there is no change in the value of v and hence not much significance changes in the value of h_{cv}. This is because the value of F, i.e. airflow rate, is very small value, although the values of F taken in this study are in realistic realm of airflow rate in DSFs. An experiment was done to test this

theory. The value of *F*, which is an input in "Airflow Windows" component in EnergyPlus was given a very high value something like 10 m^3/s.m while keeping all the other components same. Under this new value of airflow rate, there was a big change seen in overall thermal gain range of DSF with 200mm air gap vs DSF with 400mm gap and DSF with 600mm gap.

3.2.5. Variations and trends in performance range for different boundary conditions

Research questions answered here:
"How does performance range of a DSF change with boundary conditions?"
"How decoupled are thermal and visual aspects of a DSF?"

Until now, in all the previous sub-section, the study was related to, what is the overall performance range of a DSF at different operation modes and DSF configurations; and studied the reasoning behind them. The performance range of DSF_3, i.e. Single-200mm air-Double with high reflective blinds, as shown in Figure 15 presents the simulation cases with all the possible temperature differences (between indoor and outdoor) and solar radiation combinations, i.e. 14 cases of temperature difference × 6 cases of solar radiation = 84 combinations. In this sub-section all the 84 cases are studied to see how the performance of a DSF varies at every possible boundary condition and how the trends are.

This was done iteratively using a Python script as described in sub-section 2.1.4 and 2.5.2. Then as described in sub-section 2.5.3, for every combination of boundary condition, 2 correlation metrics were calculated, i.e. area of the polygon made over performance range and the Pearson correlation coefficient of thermal metric vs visual metric data. An example scatter plot of one of the 84 combinations is shown in Figure 23.

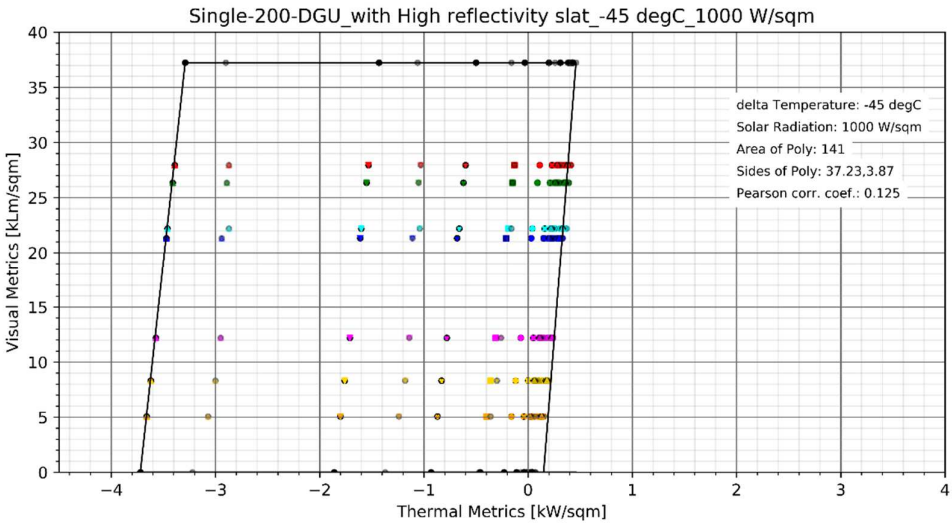


Figure 23: Performance range of DSF_3, i.e. Single-200mm air-Double with high reflective blinds, for temperature difference (between indoor and outdoor) of -45 °C and solar radiation of 1000 W/m2. As can be seen on the right side of the plot area, area and Person correlation coefficient of each case was calculated and recorded for further analysis

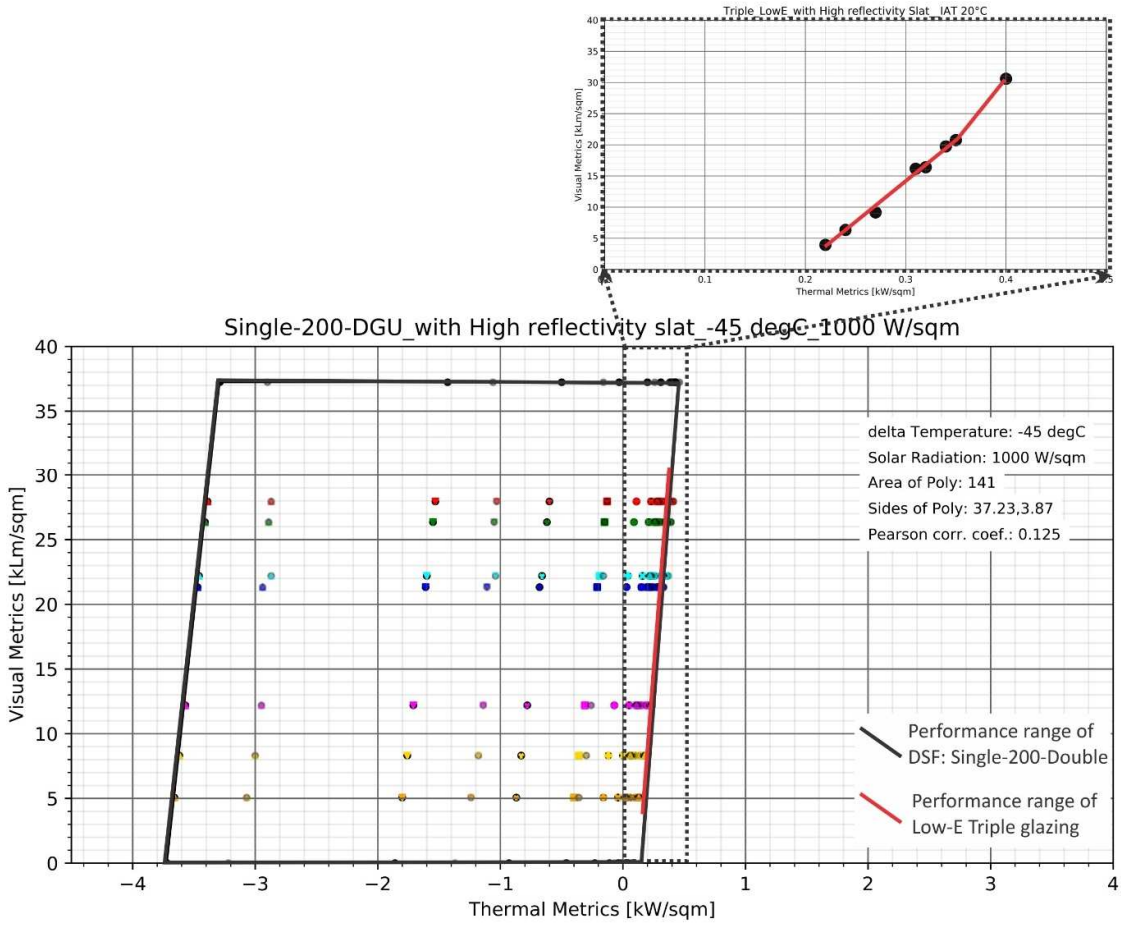


Figure 24: Illustration showing how big and wide is the performance range of a DSF compared to IGU for temperature difference (between indoor and outdoor) of -45 °C and solar radiation of 1000 W/m2. Both DSF and IGU has same 3 glass panes and shading system. DSF here is Single-200-Double and IGU is Low-e Triple glazing

To summarize the shape of the polygon, i.e. the boundary over performance range, of all 84 cases, the shape of polygon has been plotted for fixed value of temperature difference (between indoor and outdoor) with changing solar radiation on z-axis while thermal and visual metrics on x and y axis respectively. Figure 25 shows the all combinations of solar radiation with temperature difference changing from -45 °C to 20 °C.

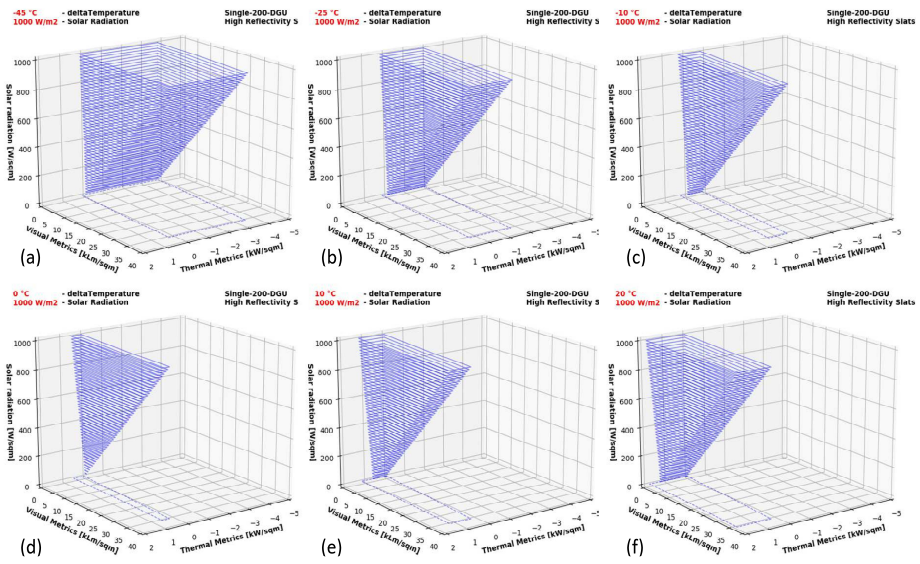


Figure 25: The shape of polygon has been plotted for fixed value of temperature difference (between indoor and outdoor) with changing solar radiation on Z-axis while thermal and visual metrics on x and y axis respectively. Sub-plot (a) represent all cases with temperature difference fixed as -45 °C, (b) -25 °C, (c) -10 °C, (d) 0 °C, (e) 10 °C, (f) 20 °C

From the figure 25 it be can concluded that the height of the polygon is dependent on the outside incident solar radiation on the façade which decided the maximum visual gain whereas width is lowest when temperature difference (between indoor and outdoor) is zero, while increasing on both sides of 0 deciding the thermal gain range. This can be summarized with a graphics shown in Figure 26.

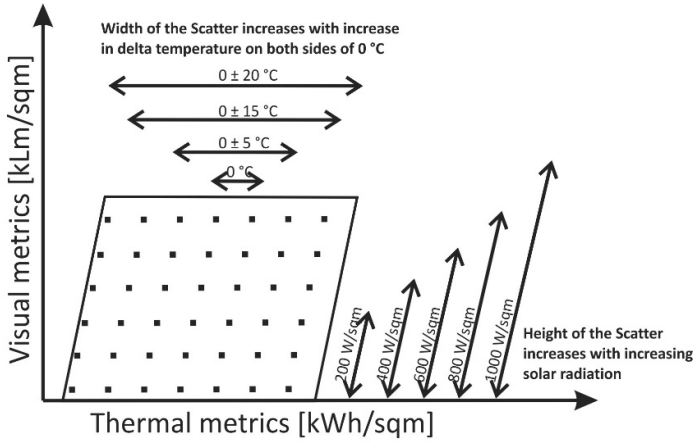


Figure 26: Illustration showing the overall trend seen how performance ranges changes with changing boundary condition

3.2.6.Decoupling performance of a DSF?

Research questions answered here:

"How decoupled are thermal and visual aspects of a DSF?"

While the investigation done above is qualitatively in nature, the area of polygon and the Pearson correlation coefficient recorded of all 84 boundary condition combinations can be further used for a quantitative analysis for how a DSF perform. How both these correlation metrics were calculated has been in detail explained in previous sub-section 2.5.3. When the area of the polygon is plotted on z-axis against thermal and visual metrics on x and y axis respectively, it can be clearly observed that area is the highest in temperature difference (between indoor and outdoor) of -45 °C and increases more with increasing solar radiation. A similar trend is also seen on the temperature difference of 20 °C, shown in Figure 27.

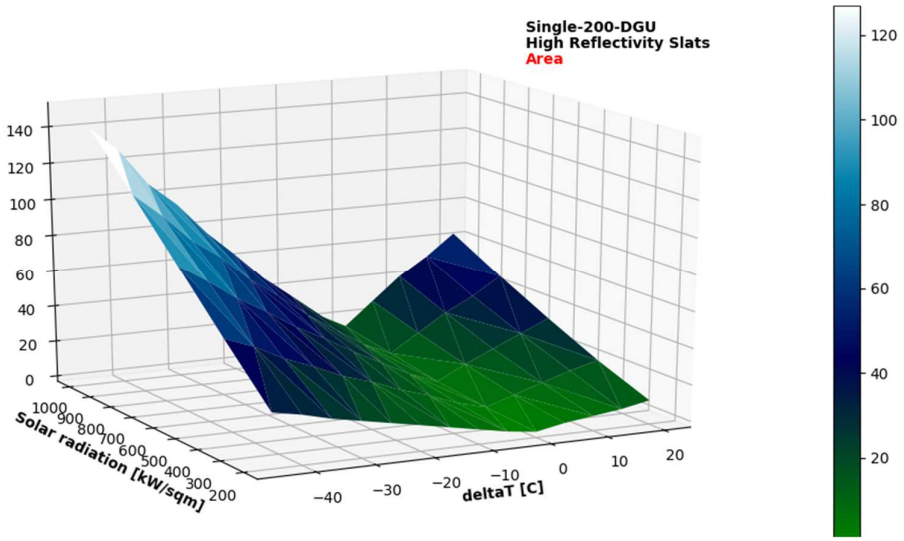


Figure 27: Area of the polygon, i.e. the boundary over performance range, on z-axis plotted against thermal and visual metrics on x and y axis respectively for DSF_3, i.e. Single-200mm air-Double with high reflective blinds.

When Pearson correlation coefficient, *r*, is plotted on z-axis, it can be observed that for boundary condition near 0 °C temperature difference (between indoor and outdoor), the value of *r* is closer to +1, meaning positive correlation while for cases going away from 0 °C temperature difference, the value of *r* trend towards 0, meaning no correlation. It was seen that at value of *r* reduces more towards 0 with decreasing solar radiation.

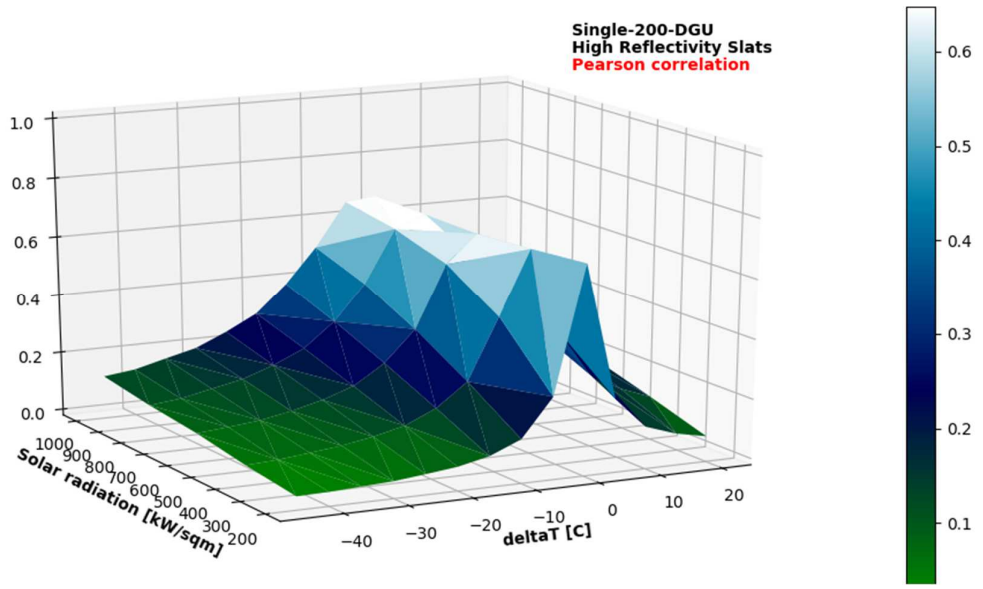


Figure 28: Pearson correlation coefficient, r, on z-axis plotted against thermal and visual metrics on x and y axis respectively for DSF_3, i.e. Single-200mm air-Double with high reflective blinds

While the area of the polygon represents the range of possibilities of thermal and visual gain at a particular boundary condition, the value of *r* represents the degree of freedom the two aspects that can be controlled individually. It can be seen in above two figures that for boundary conditions near 0 °C, neither is there possible range of thermal and visual gain, nor there is large degree of freedom to decouple the two aspects. Whereas, an interesting observation made is that, at the boundary conditions of -45 °C (and also 20 °C) while the range of possibilities increase with increasing solar radiation, the degree of freedom for decoupling decreases. The reason behind is that while at larger solar radiation there are more options of thermal and visual gain, the incident solar radiation become a strong component of the solar gain hence reducing the degree of freedom to control thermal and visual aspect separately.

If the area of polygon and the value of person coefficient, *r*, is compared for all 3 different glass arrangement, it can be seen that while the DSF with Double-200mmair-Double, seems to have less area of the polygon over extreme temperature difference (between indoor and outdoor), i.e. -45 °C and 20 °C whereas because of the less overall transmission of solar radiation, as shown by the lower peak values for the value of pearson coefficient, *r*, there is more degree of freedom to decouple thermal and visual aspects, shown in Figure 29 and 31 for high reflectivity blinds and in Figure 30 and 32 for low reflectivity blinds. Comparing cases with different reflectivity of blinds it can be seen that in comparison while high reflectivity blinds gives a bit more possible range of visual gain; low reflectivity blinds provides much more degree of freedom to decouple the thermal and visual performance.

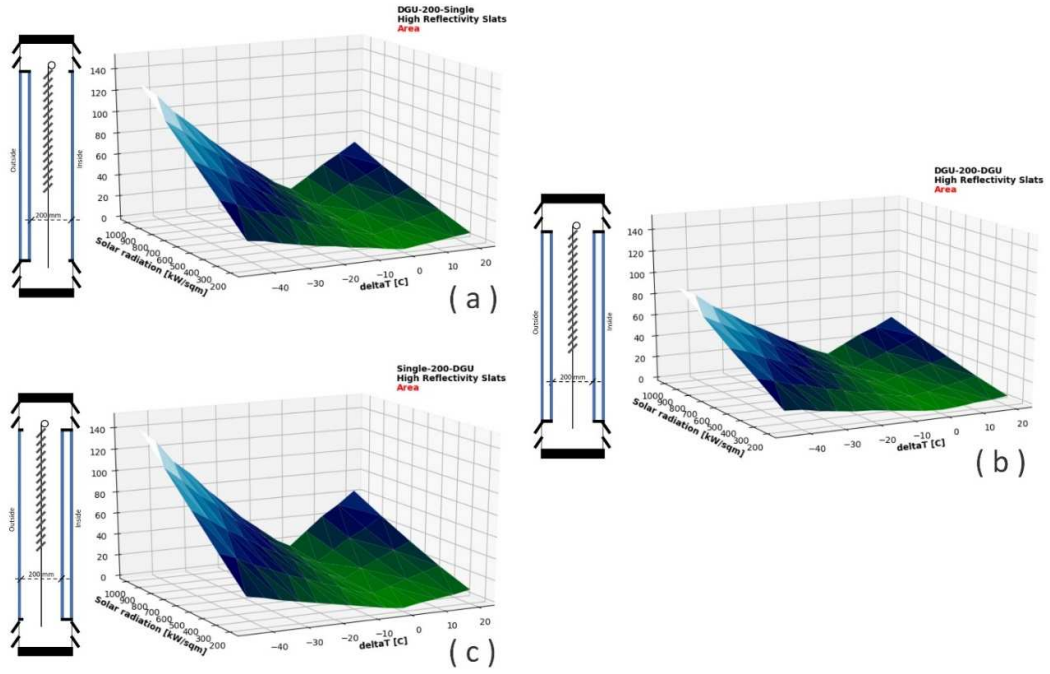


Figure 29: Area of the polygon, i.e. the boundary over performance range, on z-axis plotted against thermal and visual metrics on x and y axis respectively for three different glass arrangement for high reflectivity blinds

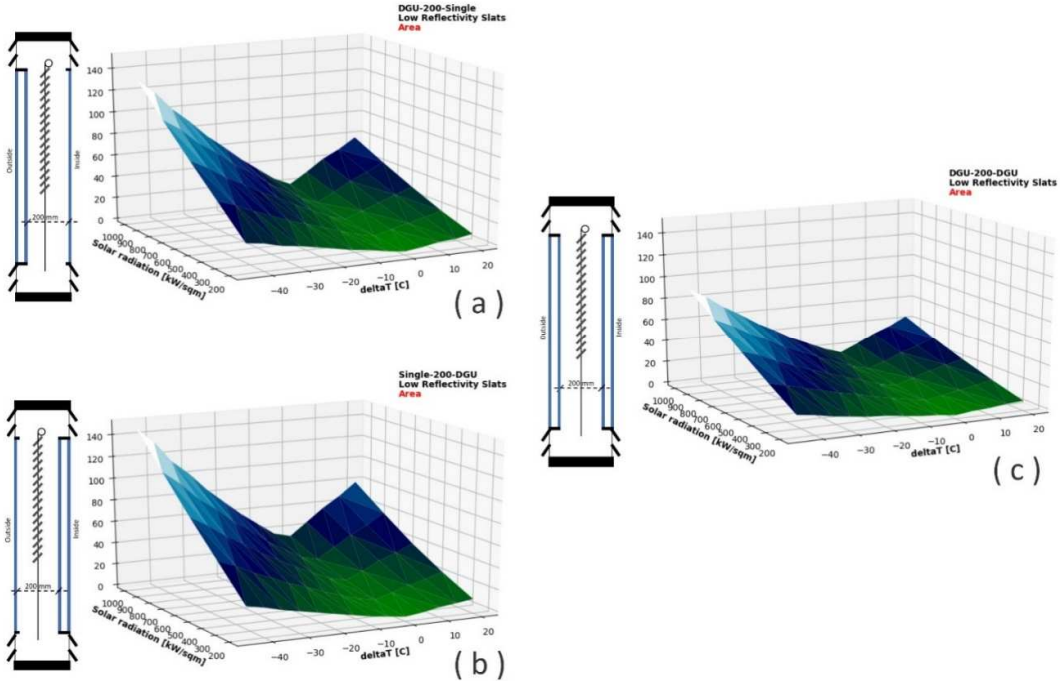


Figure 30: Area of the polygon, i.e. the boundary over performance range, on z-axis plotted against thermal and visual metrics on x and y axis respectively for three different glass arrangement for low reflectivity blinds

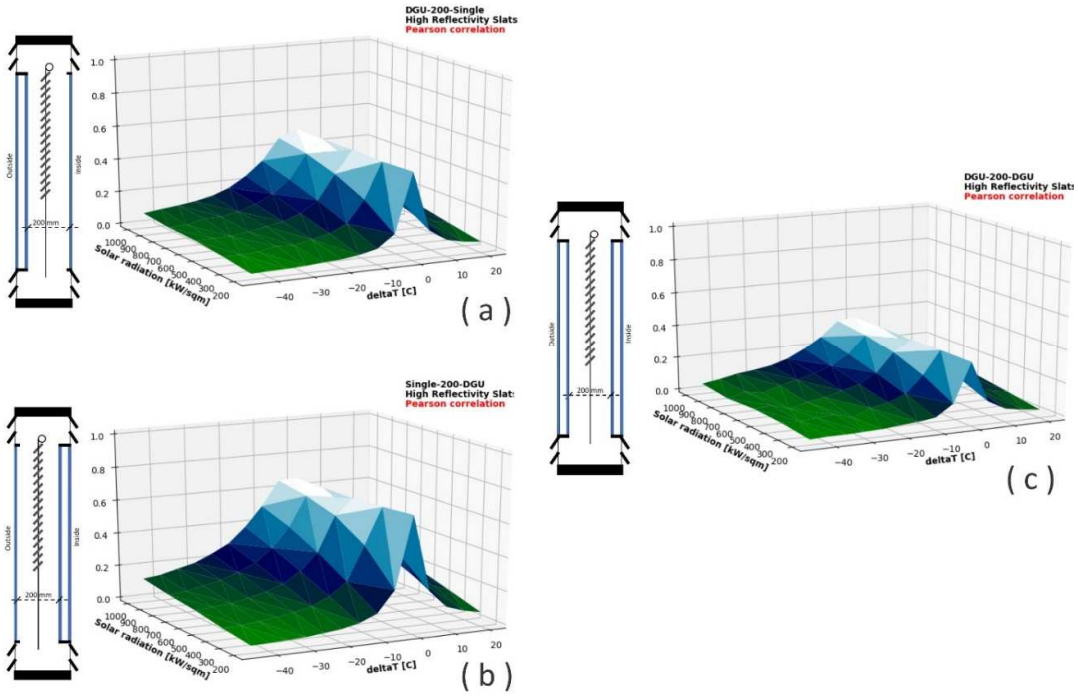


Figure 31: Pearson correlation coefficient, r, on z-axis plotted against thermal and visual metrics on x and y axis respectively for three different glass arrangement for high reflectivity blinds

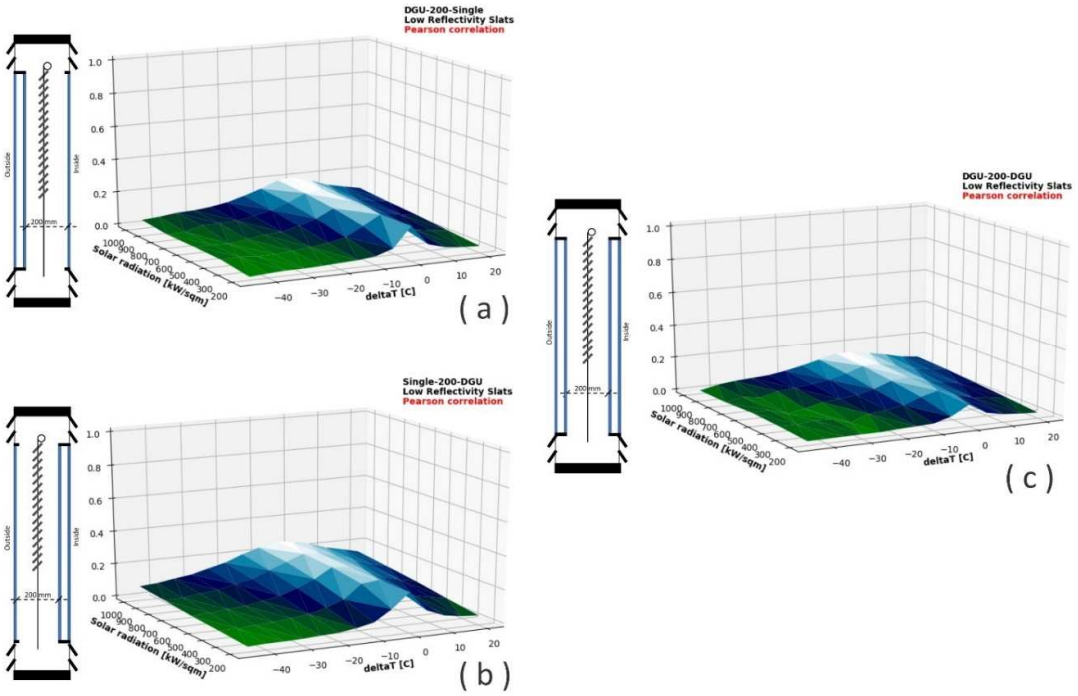


Figure 32: Pearson correlation coefficient, r, on z-axis plotted against thermal and visual metrics on x and y axis respectively for three different glass arrangement for low reflectivity blinds

4. Applications

This section explains different kinds of applications for how the work done in this thesis can be used to design and operate a DSF.

Research questions answered here:
"How this work can be used to design and operate a DSF for a certain climate?"

4.1. Studying potential of different configurations of DSFs in different climate

If temperature difference (between indoor and outdoor) and incident solar radiation combination on all orientations, i.e. north, south, east and west, for a particular climate is plotted on the 2D graph version of Figure 27, it can be seen how much independency of thermal and visual metric can be enabled in that particular climate. The analysis was done for 4 different kind of climates using EPW files for: Rome which is classified as Csa in Köppen climate classification; Oslo which is classified as Dfb; Delhi: which is classified as Cwa; and Nairobi which is classified as Cwb in Köppen climate classification. The resulting plots have been shown in Figure 33, where each plot has scatter distribution of temperature difference vs incident solar radiation on all 4 orientations, i.e. North, East, West and South.

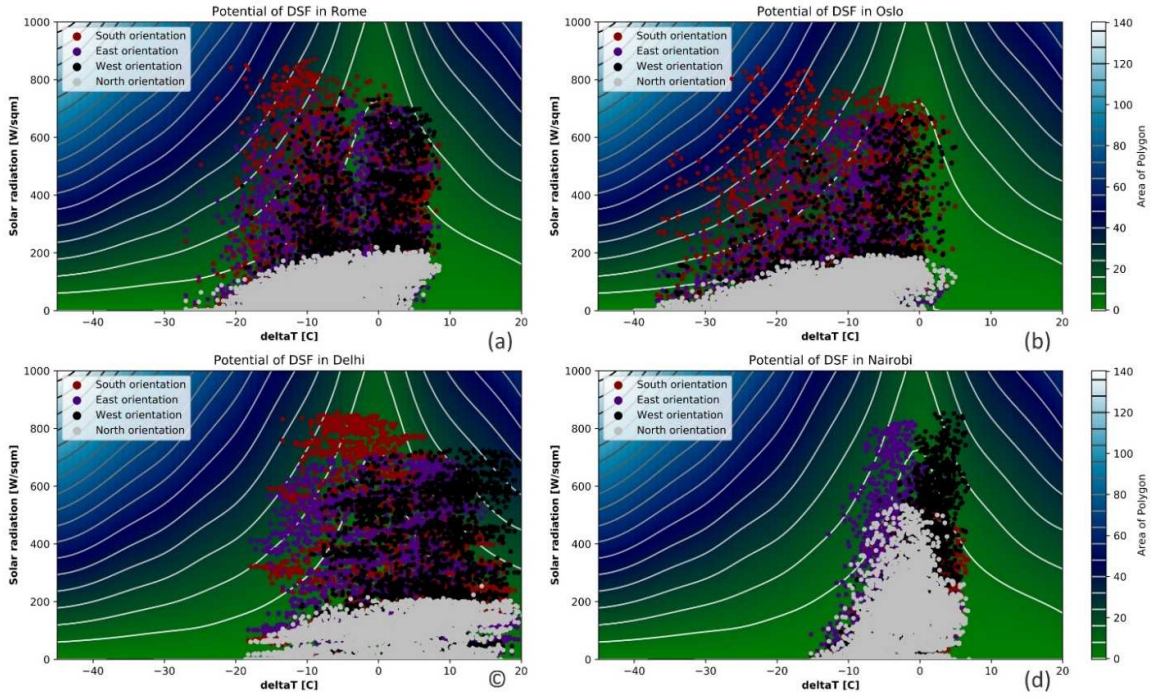


Figure 33: Potential of DSF for all orientations in different climates: (a) Rome: Köppen climate classification: Csa; (b) Oslo: Dfb; (c) Delhi: Cwa; and (d) Nairobi: Cwb

It can be seen that Nairobi, Figure 33(a) has the distribution of its of temperature difference (between indoor and outdoor) vs incident solar radiation mostly on the low "Area of Polygon" which represented by green, meaning if the DSF is used in this climate on any of the orientations, there would be very less range of possibilities of thermal and visual gain. Whereas, from Figure 33(b) it can be seen that if DSF is used in south orientation in Oslo there is both high range of performance and high degree of freedom for decoupling thermal and visual aspects. For Delhi as shown in Figure 33(c) it can be seen that there is good range of performance and degree of freedom when DSF is used in west orientations.

The results shown in Figure 33 describe the potential of DSFs in different climate is a qualitative and visual way, whereas a detailed quantitative study can be done where the potential of certain DSF configuration is quantified using some metrics, for example, the amount of area covered by boundary conditions of a certain climate on performance range graph while considering the value of z-value, i.e. color in performance graph. This quantification potential can be used to compare the performance and potential of single DSF in different climates or different kinds of DSFs in a single climate, and in one or in anther orientation.

IGU like double glazing unit or triple glazing unit; and electrochromic or thermochromic façade systems are commercialized by characterizing their thermal and visual performance in terms of some pre-defined metrics, i.e. g-value, U-value and T_{vis}. In the same way, a DSFs can also be commercialized using the metrics which can be developed using the method shown above. Their performance range over a spectrum of boundary conditions and operation modes can be visualized to convey their benefits as an efficient building envelope system aiming to reduce building energy consumption and indoor air quality for a certain climate and for all or a certain orientation.

4.2. Designing advanced control strategies for DSFs

The data gathered in this study in form of performance range of a DSF for different boundary conditions can be used along with advanced control algorithms. This section of the chapter proposes a methodology which can be used to perform such work.

4.2.1.Methodology of workflow

As it was previously discussed in sub-section 3.2.5 of chapter 3, the overall thermal and visual performance of DSF changes with the change in temperature difference (between indoor and outdoor) and incident solar radiation, as illustrated in Figure 3.12. The coordinates of the vertices of the polygon which describes the range of performance of a DSF at a particular boundary condition, were recorded for every case. As the boundary conditions are discreetly separated these vertices were interpolated using bilinear interpolation algorithms which made it possible to get the values of vertices of the polygon for any given temperature difference and solar radiation value. The interpolated shape of polygon is illustrated in Fig. 3.11, where a smooth change can be seen in the shape on z-axis which represent solar radiation changing from 0 W/sqm to 1000 W/sqm. Bilinear interpolation is an extension of linear interpolation for interpolating the output which is dependent on two variables (for this case: values of vertices is dependent on temperature difference and solar radiation) on a rectilinear 2D grid. The key idea is to perform linear interpolation first in one direction, and then again in the other direction. The initial set of vertices of polygon for 14 values of temperature difference and 6 values of solar radiation along with a Python script of bilinear interpolation can be used to describe and simulate the performance range of a DSF at any given boundary condition.

To design control strategies for a DSF in EnergyPlus there are some limitations. The airflow rate value is given as a schedule value or fixed value in “Airflow Windows” which allows it to be changed during every timestep of the simulation. Whereas, the airflow path can only be fixed to one mode before the start of simulations and also there are no actuators available for Energy Management System (EMS) of EnergyPlus to change the airflow path during simulation. This makes it practically impossible to test the controls of different airflow path in simulation of DSF.

In this case, the simulation model the DSF can be replaced with a simple glazing system whose thermal and visual gain can be simulated using the data gathered before. At every timestep’s boundary condition, the bilinear interpolation over the initial set of vertices of polygon data can used to give range of thermal and visual performance. Advanced control strategies can be the used to tell the best

value of thermal and visual gain in the selected range, which will translate as amount of gains from simple glazing unit acting as a DSF.

The "Simple glazing unit" component in EnergyPlus takes U-value, g-value and T_{vis} as inputs. To simulate a thermal gain of a DSF, the U-value and g-value input of simple glazing unit would be fixed to 0 to stop any thermal gain from outside and internal gain component was added to put thermal gain in the zone heat balance. The power input of this internal gain component would be controlled at every timestep using EMS actuators or a schedule value. It can take in both positive and negative values representing gain and loss respectively. To simulate the visual gain from a DSF, the T_{vis} input of simple glazing unit would be calibrated using solar radiation and visual gain data gathered before from simulations.

Figure 34 describes the steps of this methodology.

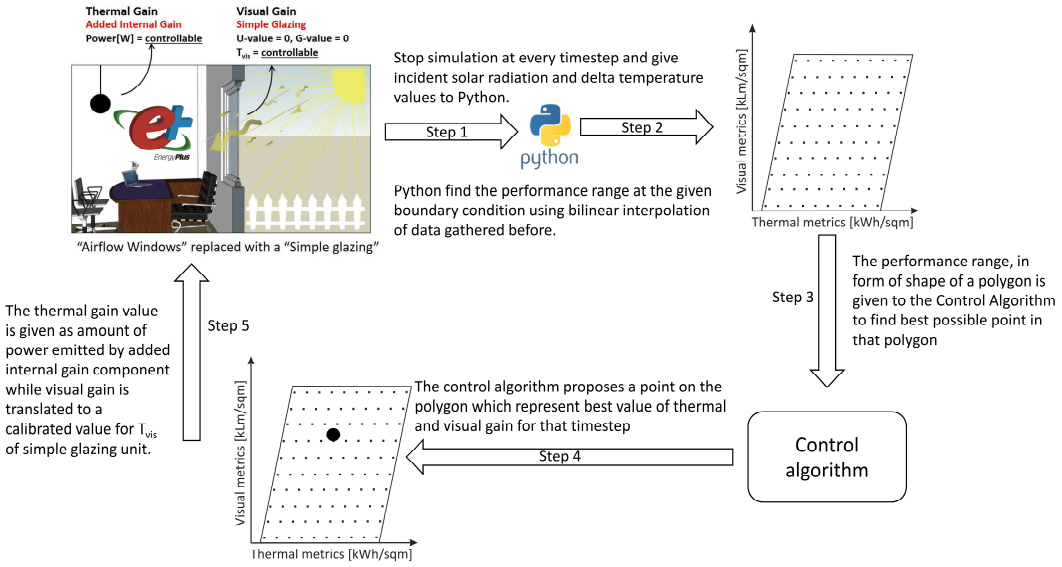


Figure 34: Steps of workflow used for simulating a DSF with an advanced control algorithm

4.2.2. Control algorithms

As shown in Figure 34, advanced control algorithms will be used to find the best possible data point, i.e. thermal gain and visual gain value, on the polygon shape which is given by Python script. Following sub-section describes the control algorithms in detail.

The best possible value of thermal gain and visual gain at a particular timestep would be found keeping following two control objectives:

a) minimum heating requirement at the particular timestep
b) UDI, i.e. Useful Daylight Illuminance, in the zone should be between 500 and 2000.

This can be done in two ways as described below

a) Option 1
Find best possible Thermal gain, Visual gain combination at a particular timestep considering performance of DSF with respect to the two control objectives, **for only that timestep.** The current timestep would be called planning horizon**.** It would be done in following steps: (note that these are sub-steps of Step 3 as shown in Figure 4.1)

Step 3.1: Take the same EnergyPlus simulation file which is used in Step 1 and stop simulation before the timestep which the control algorithm is running for.

Step 3.2: Run a multi objective optimization for every possibility of Thermal gain, Visual gain combination in the polygon which was provided by Python script in step 2, considering the two objectives only for that particular timestep, i.e. for planning horizon.

Step 3.3: As there are two control objectives, there would be several "right" options from multi-objective run. Select the best according to some pre-defined criteria.

Step 3.4: Give output as a value of thermal gain and visual gain data point on the polygon.

b) Option 2

Find best possible Thermal gain, Visual gain combination at a particular timestep considering performance of DSF for **that timestep and also for a definite future (X timesteps).** Here X timesteps would be called cost horizon. It would be done in following steps:

Step 3.1: Take the same EnergyPlus simulation file which is used in Step 1 and stop simulation before the timestep which the control algorithm is running for.

Step 3.2: Run a multi objective optimization for every possibility of Thermal gain, Visual gain combination in the polygon, considering the two objectives for cost horizon.

Step 3.3: As there are two control objectives, there would be several "right" options from multi-objective run. Select the best according to some pre-defined criteria.

Step 3.4: take results for only first timestep, i.e. for the planning horizon, and discard the rest.

Step 3.5: Give output as a value of thermal gain and visual gain data point on the polygon.

4.3. On-board real-time controller for DSF

Same methodology described in section 4.2 can be used to control the operation modes of a DSF in real time using an on-board computer controller. The only change would be to use a reduced order mathematical energy model of the room in which the DSF is installed. This would be done so that the optimization problem can be solved in very small time which can be feasible for real time considering the minimum operation mode change time.

In addition, there would be needing to translate the best combination of the thermal and visual gain selected by controller to operation mode and control commands. One way to carry out this task could be the adoption of an Artificial Neural Network (ANN) based mathematical model which would require all the data gathered from this study to train itself. This trained ANN model would be able to guess the set of operation modes, i.e. Airflow Path, Airflow Rate and Blind state, from the selected thermal and visual gain value at given boundary conditions. To improve the predication efficiency of the ANN model, more simulations might be required as done in this study which would cover remaining boundary conditions, airflow rates and blind angles.

This way of using ANN models for predicting controls for DSF has been described in steps in Fig 35

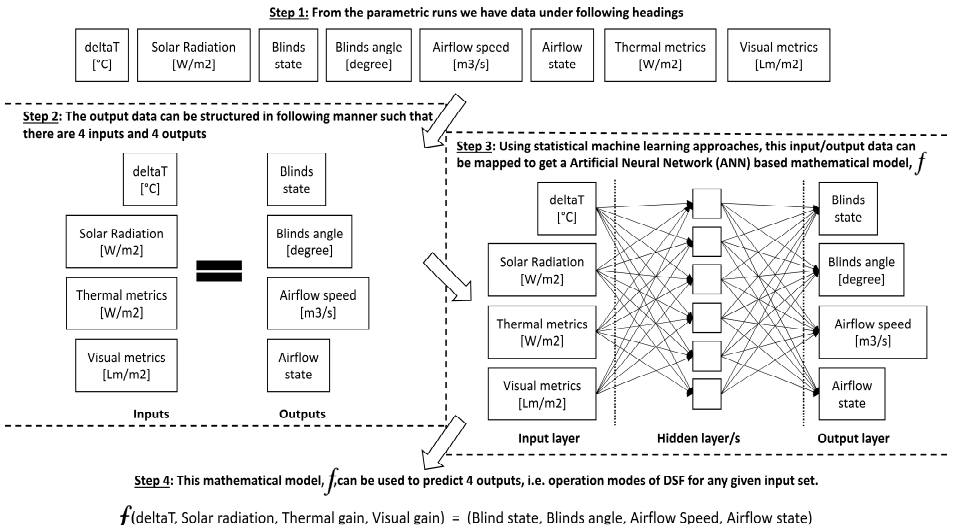


Figure 35: Steps of workflow for predicting controls of DSF using ANN models

5. Limitations and discussion

In this thesis work, the focus has been to develop a methodology to study the thermal and visual performance range of a DSF and quantitatively study the flexibility of decoupling thermal and visual aspects. Whereas the major the objectives of the work were to streamline the whole process of doing many thousands of steady state simulations, managing the data gathered on both macro and micro level, coding efficient data processing scripts and majorly designing a very orderly and coherent data presentation scheme to communicate the "Big Data" gathered in this study.

As described initially in the introduction, the whole process starts with modelling a model of a DSF in a building energy simulation program. The "Airflow Windows" in EnergyPlus simulation program was chosen for this purpose. EnergyPlus provides two ways of modelling a DSF. The simpler way, "Airflow Windows" which is used in this study, is a dedicated component available to simulate ventilated glazed cavities. This component uses a simplified analytical model when it comes to airflow calculation algorithms but is coupled with detailed heat transfer models which are used all throughout EnergyPlus. The limitations of simple airflow modelling were seen in sub-section 3.2.4 where there was no change seen in thermal gain range when air gap was increased to 3 times. The more complex way to DSF in EnergyPlus is done using the thermal and airflow network model based on AIRNET (Walton 1989). Here the ventilated air cavity is divided into several stacked zones, where each zone is an airflow network node. These nodes are linked by using different airflow network objects, which iteratively calculates the pressure, airflow linkage, temperature and humidity at every node. This way of modelling requires way much more time and expertise to correctly model a DSF, which in the end may give better or at least same results as with "Airflow Windows" component.

Although, the "Airflow Windows" component had many technical and operational limitations, using a "the most right" component of modelling was not the aim of this thesis. The major aim was to design a methodology to analyze the performance of a DSF in an unusual and comprehensive way that points out towards a more innovative use of DSF systems and not to model a DSF in the most accurate way. The methodology proposed in this thesis can be replicated using the same steps, scripts, performance metrics and data presentation schemes to see better or improved results. Whereas if another way of modelling a DSF is used, the performance metrics may be modified to better work coherently with outputs results of simulation, while keeping the meaning behind them same.

As seen in the types of DSF configurations selected in this thesis, a DSF can have equal to or more than 3 glass planes which may have total glass thickness of more than 25mm. This amount of thermal mass would have thermal inertial effects upto to more than 1 hour during peak incident solar radiations. This will lead to time shifted effects of operation modes on overall thermal gain which

have not been studied in this thesis. The simulations done for this thesis were steady state in nature, which means the immediate effect of changes was seen on thermal and visual gain. On the other hand, EnergyPlus also does not have capacitive node in thermal models of heat transfer for glazing systems. To study the inertial effect of operation modes the same study could be done but with every operation mode considering also some number of every iteration of history operation modes. This would lead to a very large computation problem which might be near to impossible to perform. At the same time if the gathered data from this study is used in real time controller for a DSF, to incorporate the thermal inertial effect, appropriate modifications would have to made to cost functions of the optimization and controller equations

6. Conclusions

This thesis develops and demonstrates a methodology to study the range of thermal and visual performance of a DSF in different boundary conditions and varied operation modes; and how flexible a certain DSF is to control the thermal and visual aspects separately. To do so steady state energy simulations were performed in a mathematical model of a DSF on a range of realistic boundary conditions, i.e. temperature difference (between indoor and outdoor) and incident solar radiation. The control parameters of the DSF were varied to a set of realistic values to get different operation modes. All the data processing, data management and plotting of results was done using Python.

Performance metrics for thermal gain and visual gain are presented in this study. The main difference between the presented metrics, and the standard way to evaluate the performance of façades as U-value, g-value, and so on, is that the presented metrics cannot be calculated directly from physical characteristics of the materials adopted in a typical façade multilayer system / construction, and do not have a general physical meaning. Whereas, they represent the final effect of the façade in terms of thermal and visual gains.

Regarding research questions related to analysis of performance of a DSF, have been discussed in Chapter 2. Here, the overall range of thermal and visual gain was shown as a scatter plot and presented how the performance changes with changing boundary conditions and operation modes. Air Supply and Air Extract presents a large range of thermal gain than other airflow paths, whereas in terms of airflow rate, higher rate provides the maximum range which reduces with decreasing airflow rate. When a DSF is used without slats highest amount of visual gain was noticed whereas different slat angle, if blinds are used, provide much more options of visual gain. Results were then studied for different combinations of boundary conditions to study how performance and flexibility changes. The area of the total range of performance and person correlation coefficient, r, were calculated for every boundary condition. The value of area represented a comparative figure of how big or small is the range of thermal and visual performance of a DSF, whereas the value of r represented how much correlation was present between thermal and visual gain, which could also be interpreted as a comparative figure for degree of freedom for decoupling the two aspects. The results outlined a trend that both range of performance and degree of freedom of decupling was lowest when temperature difference (between indoor and outdoor) was zero and lowest solar radiation. Both these values increased with increase of temperature difference on both sides of 0 with highest being at -45 °C and 20 °C. On solar radiation axis, while the range of performance increased with increasing incident solar radiation, the degree of freedom of decoupling decreased. The obvious reason for this trend is that although increasing solar radiation provides good range of performance, it becomes a strong influencer at higher values which starts increasing correlation between thermal and visual gains. This trend was seen in all different kinds of DSFs whereas the reason behind it was confirmed in DSF with double glazing on both inside and outside. Because of overall less solar radiation transmission due to less U-value of the combined glass panes, almost same and high degree of freedom for decoupling was seen at all solar radiation and highest temperature difference combinations.

On the research question, *"How this work can be used to design and operate a DSF for a certain climate?"*, a methodology was presented on how to study the potential of a DSF in any given climate and how different DSFs can be compared during initial design stage for application in buildings. Another detailed methodology is presented on how to use the results of this thesis to efficiently

operate a DSF. This methodology will be used to find the best possible set of controls and operation modes for a DSF considering its thermal and visual performance at a certain timestep and its effects on future timesteps. Thesis is concluded by a discussion of the limitations part of this study, the reason behind why certain aspects of DSF modelling were not considered and how this would be different in a real time controller.

References

[1] Pomponi, F., Piroozfar, P. A., Southall, R., Ashton, P., & Farr, E. R. (2016). Energy performance of Double-Skin Façades in temperate climates: A systematic review and meta-analysis. Renewable and Sustainable Energy Reviews, 54, 1525-1536.

[2] Oesterle, E. (2001). Double skin facades: integrated planning; building physics, construction, aerophysics, air-conditioning, economic viability. Prestel.

[3] Gratia, E., & De Herde, A. (2004). Natural ventilation in a double-skin facade. Energy and buildings, 36(2), 137-146.

[4] Poirazis, H. (2004). Double skin façades for office buildings. Holland: Lund Institute of Technology.

[5] Saelens, D., & Hens, H. (2001). Experimental evaluation of airflow in naturally ventilated active envelopes. Journal of Thermal Envelope and Building Science, 25(2), 101-127.

[6] Haase, M., da Silva, F. M., & Amato, A. (2009). Simulation of ventilated facades in hot and humid climates. Energy and Buildings, 41(4), 361-373.

[7] Baldinelli, G. (2009). Double skin façades for warm climate regions: Analysis of a solution with an integrated movable shading system. Building and Environment, 44(6), 1107-1118.

[8] Wigginton, M., & McCarthy, B. The Environmental Second Skin'. Research carried out for the UK Department of the Environment Transport and the Regions.

[9] Pappas A. Energy performance of a DSF–analysis for the Museum of Contemporary Art, Denver. SOLAR 2006. Denver, USA; 2006.

[10] Oka, S., Andjelkovic, A. S., Cvjetkovic, T. B., Djakovic, D. D., & Stojanovic, I. H. (2015). Development of simple calculation model for energy performance of double skin facades (vol 16, pg S251, 2012). THERMAL SCIENCE, 19(2), 749-749.

[11] Xu, L., & Ojima, T. (2007). Field experiments on natural energy utilization in a residential house with a double skin façade system. Building and Environment, 42(5), 2014-2023.

[12] Yılmaz, Z., & Çetintaş, F. (2005). Double skin façade's effects on heat losses of office buildings in Istanbul. Energy and Buildings, 37(7), 691-697.

[13] Stec, W. J., & Van Paassen, A. H. C. (2005). Symbiosis of the double skin facade with the HVAC system. Energy and Buildings, 37(5), 461-469.

[14] Kragh, M. (2001). Monitoring of Advanced Facades and Environmental Systems. whole-life performance of facades University of Bath, CWCT.

[15] Kragh M. Facade engineering and building physics. Examples of current best practice and recent innovations. Integrated Facade Symposium. San Francisco;21 April, 2010.

[16] Ghadamian, H., Ghadimi, M., Shakouri, M., Moghadasi, M., & Moghadasi, M. (2012). Analytical solution for energy modeling of double skin façades building. Energy and Buildings, 50, 158-165.

[17] Kim, G., Schaefer, L., & Kim, J. T. (2013). Development of a double-skin facade for sustainable renovation of old residential buildings. Indoor and built environment, 22(1), 180-190.

[18] DiMaio, F., & Van Passen, A.H.C. (2001). Modelling the air infiltrations in the second skin façade. In: Proceedings of IAQVEC 2001 – The 4th International Conference on Indoor Air Quality, Ventilation and Energy Conservation in Buildings, 2-5 October 2001, Changsha (China), pp.873-880.

[19] Corgnati S.P., Perino M., & Serra V. (2007). Experimental assessment of the performance of an active transparent façade during actual operating conditions. Solar Energy 81:8, pp.993-1013.

[20] Crawley, D. B., Lawrie, L. K., Pedersen, C. O., & Winkelmann, F. C. (2000). Energy plus: energy simulation program. ASHRAE journal, 42(4), 49-56.

[21] EnergyPlus. EnergyPlus engineering reference: the reference to EnergyPlus calculations. Tech. rep. Berkeley, CA: Ernest Orlando Lawrence Berkeley National Laboratory. 72, 85, 100. 2013.

Appendix 1: Python scripts developed for this study

Making simulation cases

```
import csv
with open('cases.csv', 'w', newline='') as csvfile:
    spamwriter = csv.writer(csvfile, delimiter=',', quoting=csv.QUOTE_MINIMAL)

    spamwriter.writerow(['Case No.','Outside Dry bulb Temp', 'Outside Dew point Temp','Rh','Inside Air Temp','Direct Solar Radiation','Blinds state','Blinds angle','Airflow speed','Airflow state','Airflow Source','Airflow Destination','Blinds refl'])

    outsidedrytemp = [-20,-10,0,10,20,30,40]
    outsidedewtemp = [-26,-17,-7,2,12,22,31]  #dewpoint temp calculated for Rh 0.6
    insidetemp = [20,25]
    solarrad = [0,2000,4000,6000,8000,9955]
    blinds = ['on','off']
    blindangle = [0,30,45,60,90,120,135,150]
    airflowspeed = [10/3600,20/3600,40/3600,80/3600,160/3600,320/3600]
    airflowmode = ['supply mode','extract mode','interior air curtain','exterior air curtain']

    case = 1
    for i in range(len(outsidedrytemp)):
        for j in range(len(insidetemp)):
            for k in range(len(solarrad)):
                for m in range(len(blindangle)):
                    spamwriter.writerow(['Case_'+str(case),outsidedrytemp[i],outsidedewtemp[i],'60',insidetemp[j],solarrad[k],blinds[0],blindangle[m],'0','interior air curtain','IndoorAir','IndoorAir','Highrefl blinds'])
                    case=case+1
                    for n in range(len(airflowspeed)):
                        for p in range(len(airflowmode)):
                            if airflowmode[p]=='supply mode':
                                s='OutdoorAir'
                                d='IndoorAir'
                            if airflowmode[p] == 'extract mode':
                                s='IndoorAir'
                                d='OutdoorAir'
                            if airflowmode[p] == 'interior air curtain':
                                s='IndoorAir'
                                d='IndoorAir'
                            if airflowmode[p] == 'exterior air curtain':
                                s='OutdoorAir'
                                d='OutdoorAir'
                            spamwriter.writerow(['Case_'+str(case),outsidedrytemp[i],outsidedewtemp[i],'60',insidetemp[j],solarrad[k],blinds[0],blindangle[m],airflowspeed[n],airflowmode[p],s,d,'Highrefl blinds'])
                            case=case+1

    for i in range(len(outsidedrytemp)):
        for j in range(len(insidetemp)):
            for k in range(len(solarrad)):
                for m in range(len(blindangle)):
                    spamwriter.writerow(['Case_'+str(case),outsidedrytemp[i],outsidedewtemp[i],'60',insidetemp[j],solarrad[k],blinds[0],blindangle[m],'0','interior air curtain','IndoorAir','IndoorAir','Lowrefl blinds'])
                    case=case+1
                    for n in range(len(airflowspeed)):
                        for p in range(len(airflowmode)):
                            if airflowmode[p]=='supply mode':
                                s='OutdoorAir'
```

```
                                d='IndoorAir'
                            if airflowmode[p] == 'extract mode':
                                s='IndoorAir'
                                d='OutdoorAir'
                            if airflowmode[p] == 'interior air curtain':
                                s='IndoorAir'
                                d='IndoorAir'
                            if airflowmode[p] == 'exterior air curtain':
                                s='OutdoorAir'
                                d='OutdoorAir'
                            spamwriter.writerow(['Case_'+str(case),outsidedrytemp[i],outsidedewtemp[i],'60',insidetemp[j],solarrad[k],blinds[0],blindangle[m],airflowspeed[n],airflowmode[p],s,d,'Lowrefl blinds'])
                            case=case+1

      #blinds off, i.e. there is no blinds on windows
    for i in range(len(outsidedrytemp)):
        for j in range(len(insidetemp)):
            for k in range(len(solarrad)):
                spamwriter.writerow(['Case_'+str(case),outsidedrytemp[i],outsidedewtemp[i],'60',insidetemp[j],solarrad[k],blinds[1],'-','0','interior air curtain','IndoorAir','IndoorAir','No Blinds'])
                case=case+1
                for n in range(len(airflowspeed)):
                    for p in range(len(airflowmode)):
                        if airflowmode[p]=='supply mode':
                            s='OutdoorAir'
                            d='IndoorAir'
                        if airflowmode[p] == 'extract mode':
                            s='IndoorAir'
                            d='OutdoorAir'
                        if airflowmode[p] == 'interior air curtain':
                            s='IndoorAir'
                            d='IndoorAir'
                        if airflowmode[p] == 'exterior air curtain':
                            s='OutdoorAir'
                            d='OutdoorAir'
                        spamwriter.writerow(['Case_'+str(case),outsidedrytemp[i],outsidedewtemp[i],'60',insidetemp[j],solarrad[k],blinds[1],'-','',airflowspeed[n],airflowmode[p],s,d,'No Blinds'])
                        case=case+1

print("The total number of cases are: ", case-1)
```

Preparing multiple IDFs and EPWs

```
import csv
with open('cases.csv', newline='') as f:
    reader = csv.reader(f)
    data = []
    for row in reader:
        data.append(row)
    size=len(data)
    print(size)

    for i in range(1,size):
    #for i in range(14690,14691):

        filename=data[i][0]
        dbt=data[i][1]
        dpt=data[i][2]
        rh=data[i][3]
        iat=data[i][4]
        dsr=data[i][5]
        bst=data[i][6]
```

```
        ban=data[i][7]
        asp=data[i][8]
        aso=data[i][10]
        ade=data[i][11]
        refl=data[i][12]

        # making IDFs
        j='9zones FINAL_temp.idf'
        with open(j, 'r') as file :
            dataidf = file.read()
        # Replace the target string
        dataidf = dataidf.replace('$IAT@@', str(iat))
        dataidf = dataidf.replace('$OAT@@', str(dbt))

        if bst == 'on':
            dataidf = dataidf.replace('$BST1@@', 'Glaz2')
            dataidf = dataidf.replace('$BST2@@', 'Glaz2')
            dataidf = dataidf.replace('$BST3@@', 'Glaz2')
            dataidf = dataidf.replace('$BST4@@', 'Glaz2')
            dataidf = dataidf.replace('$BST5@@', 'Glaz2')
            dataidf = dataidf.replace('$BST6@@', 'Glaz2')
            dataidf = dataidf.replace('$BST7@@', 'Glaz2')
            dataidf = dataidf.replace('$BST8@@', 'Glaz2')
            dataidf = dataidf.replace('$BST9@@', 'Glaz2')
            dataidf = dataidf.replace('$BAN@@', str(ban))
        if bst == 'off':
            dataidf = dataidf.replace('$BST1@@', '')
            dataidf = dataidf.replace('$BST2@@', '')
            dataidf = dataidf.replace('$BST3@@', '')
            dataidf = dataidf.replace('$BST4@@', '')
            dataidf = dataidf.replace('$BST5@@', '')
            dataidf = dataidf.replace('$BST6@@', '')
            dataidf = dataidf.replace('$BST7@@', '')
            dataidf = dataidf.replace('$BST8@@', '')
            dataidf = dataidf.replace('$BST9@@', '')
            dataidf = dataidf.replace('$BAN@@', '45')
        dataidf = dataidf.replace('$ASP@@', str(asp))
        dataidf = dataidf.replace('$ASO@@', str(aso))
        dataidf = dataidf.replace('$ADE@@', str(ade))
        dataidf = dataidf.replace('$BLINDS@@', str(refl))
        with open('cases/'+str(filename)+'.idf', 'w') as file:
            file.writelines( dataidf )

        # making EPWs
        k='NFRC OUTSIDE_temp.epw'
        with open(k, 'r') as file :
            dataepw = file.read()
        # Replace the target string
        dataepw = dataepw.replace('$DBT@@', str(dbt))
        dataepw = dataepw.replace('$DPT@@', str(dpt))
        dataepw = dataepw.replace('$RH@@', str(rh))
        dataepw = dataepw.replace('$DSR@@', str(dsr))
        with open('cases/'+str(filename)+'.epw', 'w') as file:
            file.writelines( dataepw )
        if  i%100==0:
            print(filename)

```

Making batch file for EP simulations

```
import csv
import glob
import subprocess
import os

```

```
def split(a, n):
    k, m = divmod(len(a), n)
    return (a[i * k + min(i, m):(i + 1) * k + min(i + 1, m)] for i in range(n))

#numfiles = 31028
numfiles = len(glob.glob1('cases/','*.idf'))

#distribute in 5 files-----5 as x
x=5

print(list(split(range(numfiles), x)))
len1=len(list(split(range(numfiles), x))[0])
print(list(split(range(numfiles), x))[0][len1-1])
print(list(split(range(numfiles), x))[0][0])

#make individual BAT files
for i in range(x):
    with open('cases/Run_'+str(i+1)+'simulation.bat', 'w') as filef:
        with open('temp.bat', 'r') as file :
            dataepg = file.read()
        start = list(split(range(numfiles), x))[i][0]
        last =  list(split(range(numfiles), x))[i][len(list(split(range(numfiles),
x))[i])-1]
        dataepg = dataepg.replace('@start@', str(start+1))
        dataepg = dataepg.replace('@end@', str(last+1))
        filef.writelines( dataepg )

#check if runEPfile.bat exist and if delete
if os.path.exists("cases/runEPfile.bat"):
  os.remove("cases/runEPfile.bat")


#make runEPfile.bat to run all cmd windows together
f= open("cases/runEPfile.bat","a+")
f.write('cd "cases/"\n')

for i in range(x):
     f.write('start '+'Run_'+str(i+1)+'simulation.bat \n')
f.close()

run runEPfile.bat
subprocess.call('cases/runEPfile.bat',creationflags=subprocess.CREATE_NEW_CONSOLE)

subprocess.call([r'cases\runEPfile.bat'])
```

Batch file for EP simulations

```
FOR /L %%A IN (@start@,1,@end@) DO (
TITLE Case_%%A running
energyplus -w Case_%%A.epw -p Case_%%A -r Case_%%A.idf

IF EXIST "Case_%%Aout.epmidf" DEL "Case_%%Aout.epmidf"
IF EXIST "Case_%%Aout.epmdet" DEL "Case_%%Aout.epmdet"
IF EXIST "Case_%%Aout.eso" DEL "Case_%%Aout.eso"
IF EXIST "Case_%%Aout.rdd" DEL "Case_%%Aout.rdd"
IF EXIST "Case_%%Aout.mdd" DEL "Case_%%Aout.mdd"
IF EXIST "Case_%%Aout.eio" DEL "Case_%%Aout.eio"
IF EXIST "Case_%%Aout.end" DEL "Case_%%Aout.end"
:IF EXIST "Case_%%Aout.err" DEL "Case_%%Aout.err"
IF EXIST "Case_%%Aout.dxf" DEL "Case_%%Aout.dxf"
IF EXIST "Case_%%Atbl.tab" DEL "Case_%%Atbl.tab"
IF EXIST "Case_%%Aout.txt" DEL "Case_%%Aout.txt"
:IF EXIST "Case_%%AoutMeter.csv" DEL "Case_%%AoutMeter.csv"
IF EXIST "Case_%%AoutMeter.tab" DEL "Case_%%AoutMeter.tab"
IF EXIST "Case_%%AoutMeter.txt" DEL "Case_%%AoutMeter.txt"
```

```
IF EXIST "Case_%%Aout.det" DEL "Case_%%Aout.det"
IF EXIST "Case_%%Aout.sln" DEL "Case_%%Aout.sln"
IF EXIST "Case_%%Aout.Zsz" DEL "Case_%%Aout.Zsz"
:IF EXIST "Case_%%AoutZsz.csv" DEL "Case_%%AoutZsz.csv"
IF EXIST "Case_%%AoutZsz.tab" DEL "Case_%%AoutZsz.tab"
IF EXIST "Case_%%AoutZsz.txt" DEL "Case_%%AoutZsz.txt"
IF EXIST "Case_%%Aout.ssz" DEL "Case_%%Aout.ssz"
:IF EXIST "Case_%%AoutSsz.csv" DEL "Case_%%AoutSsz.csv"
IF EXIST "Case_%%AoutSsz.tab" DEL "Case_%%AoutSsz.tab"
IF EXIST "Case_%%AoutSsz.txt" DEL "Case_%%AoutSsz.txt"
IF EXIST "Case_%%Aout.mtr" DEL "Case_%%Aout.mtr"
IF EXIST "Case_%%Aout.mtd" DEL "Case_%%Aout.mtd"
IF EXIST "Case_%%Aout.bnd" DEL "Case_%%Aout.bnd"
IF EXIST "Case_%%Aout.dbg" DEL "Case_%%Aout.dbg"
IF EXIST "Case_%%Aout.sci" DEL "Case_%%Aout.sci"
IF EXIST "Case_%%Aout.svg" DEL "Case_%%Aout.svg"
IF EXIST "Case_%%Aout.shd" DEL "Case_%%Aout.shd"
IF EXIST "Case_%%Aout.wrl" DEL "Case_%%Aout.wrl"
:IF EXIST "Case_%%AoutScreen.csv" DEL "Case_%%AoutScreen.csv"
:IF EXIST "Case_%%AoutMap.csv" DEL "Case_%%AoutMap.csv"
IF EXIST "Case_%%AoutMap.tab" DEL "Case_%%AoutMap.tab"
IF EXIST "Case_%%AoutMap.txt" DEL "Case_%%AoutMap.txt"
IF EXIST "Case_%%Aout.audit" DEL "Case_%%Aout.audit"
:IF EXIST "Case_%%AoutTable.csv" DEL "Case_%%AoutTable.csv"
IF EXIST "Case_%%AoutTable.tab" DEL "Case_%%AoutTable.tab"
IF EXIST "Case_%%AoutTable.txt" DEL "Case_%%AoutTable.txt"
:IF EXIST "Case_%%AoutTable.html" DEL "Case_%%AoutTable.html"
:IF EXIST "Case_%%AoutTable.htm" DEL "Case_%%AoutTable.htm"
IF EXIST "Case_%%AoutTable.xml" DEL "Case_%%AoutTable.xml"
IF EXIST "Case_%%AoutDElight.in" DEL "Case_%%AoutDElight.in"
IF EXIST "Case_%%AoutDElight.out" DEL "Case_%%AoutDElight.out"
IF EXIST "Case_%%AoutDElight.dfdmp" DEL "Case_%%AoutDElight.dfdmp"
IF EXIST "Case_%%AoutDElight.eldmp" DEL "Case_%%AoutDElight.eldmp"
IF EXIST "Case_%%AoutSpark.log" DEL "Case_%%AoutSpark.log"
IF EXIST "Case_%%Aout.expidf" DEL "Case_%%Aout.expidf"
IF EXIST "Case_%%Aout.rvaudit" DEL "Case_%%Aout.rvaudit"
IF EXIST "Case_%%Aout.sql" DEL "Case_%%Aout.sql"
IF EXIST "Case_%%Aout.edd" DEL "Case_%%Aout.edd"
:IF EXIST "Case_%%AoutDFS.csv" DEL "Case_%%AoutDFS.csv"
IF EXIST "Case_%%Aout*.mat" DEL "Case_%%Aout*.mat"

)
exit
```

Post processing the collected data

```
import csv
import glob
from scipy import interpolate
import os.path
import sys

#casenum = sys.argv[1]
casenum = 1


def air_density_ip(dbt1):
    t =[-20,-10,0,10,20,30,40]
    dens=[1.395,1.3413,1.2922,1.2466,1.2041,1.1644,1.127]
    if float(dbt1) >= -20 and float(dbt1) <= 40:
        f = interpolate.interp1d(t, dens, fill_value = "interpolate")
    else:
        f = interpolate.interp1d(t, dens, fill_value = "extrapolate")
    return f(float(dbt1))

```

```
def extract_extra(dbt2,iat1,asp1):
    extra = (float(iat1)-float(dbt2))*float(asp1)*1*air_density_ip(dbt2)*1000/1.5
    return float(extra)

def supply_extra(dbt3,iat2,asp2,gapconv):
    extra = (float(iat2)-
gapconv_gaptemp(gapconv,dbt3,asp2))*float(asp2)*1*air_density_ip(gapconv_gaptemp(ga
pconv,dbt3,asp2))*1000/1.5
    return float(extra)

def gapconv_gaptemp(gapconv1,dbt4,asp3):
    gap_temp = float(dbt4)+(float(gapconv1)/(float(asp3)*1*air_density_ip(dbt4)*100
0/1.5))
    return float(gap_temp)

print(gapconv_gaptemp(206,20,0.044))
print(extract_extra(-20,20,0.044))
print(supply_extra(-20,20,0.044,206))
print(air_density_ip(44.24))

dbttemp =[]
iattemp =[]
bsttemp =[]
asptemp =[]
asttemp =[]

with open('cases.csv', newline='') as f:
    reader = csv.reader(f)
    data1 = []
    for row in reader:
        data1.append(row)
    size=len(data1)
    for i in range(1,size):
        #filename=data1[i][0]
        dbttemp.append(data1[i][1])
        iattemp.append(data1[i][4])
        #dsr=data1[i][5]
        bsttemp.append(data1[i][6])
        #ban=data1[i][7]
        asptemp.append(data1[i][8])
        asttemp.append(data1[i][9])
        #aso=data1[i][10]
        #ade=data1[i][11]

#print(dbttemp)

a1_shgc=[]
a1_shgc.append('a1_Thermal metrics [W/m2]')
a1_vlt= []
a1_vlt.append('a1_Visual metrics [Lm/m2]')
a1_outSRa=[]
a1_outSRa.append('a1_Incident Solar Radiation Rate per Area [W/m2]')
a1_transSRa=[]
a1_transSRa.append('a1_Transmitted Solar Radiation Rate [W/m2]')
a1_gapCONVa=[]
a1_gapCONVa.append('a1_Gap Convective Heat Transfer Rate [W/m2]')
a1_insCONVa=[]
a1_insCONVa.append('a1_Zone Convection Heat Gain Rate [W/m2]')
a1_insINFRAa=[]
a1_insINFRAa.append('a1_Net Infrared Heat Transfer Rate [W/m2]')
a1_gaptemp=[]
a1_gaptemp.append('a1_Gap Temperature [C]')
a1_deltatemp=[]
a1_deltatemp.append('a1_delta Temperature [C]')


```

```
for x in range(1,35701):
    if  x%5000==0:
        print(x)
    if str(os.path.exists('cases/glaz'+str(casenum)+'/Case_'+str(x)+'out.csv'))=='F
alse':
        zero=0
        a1_outSRa.append(zero)
        a1_transSRa.append(zero)
        a1_gapCONVa.append(zero)
        a1_insCONVa.append(zero)
        a1_insINFRAa.append(zero)
        a1_shgc.append(zero)
        a1_vlt.append(zero)
        a1_deltatemp.append(zero)
        a1_gaptemp.append(zero)

    else:
        with open('cases/glaz'+str(casenum)+'/Case_'+str(x)+'out.csv', newline='')
as f:
                    reader = csv.reader(f)
                    data = []
                    for row1 in reader:
                        data.append(row1)
                    k = 12                  #line in which data is saved, i.e. date

                    a1_outSR = data[k][6]
                    a1_outSRa.append(a1_outSR)
                    a1_transSR = float(data[k][7])/4.75    #WINDOW AREA IS 4.75
                    a1_transSRa.append(a1_transSR)
                    a1_gapCONV = float(data[k][8])/4.75
                    a1_gapCONVa.append(a1_gapCONV)
                    a1_insCONV = float(data[k][9])/4.75
                    a1_insCONVa.append(a1_insCONV)
                    a1_insINFRA = float(data[k][10])/4.75
                    a1_insINFRAa.append(a1_insINFRA)

                    if asttemp[x-1] == 'interior air curtain':
                        a1_shgc.append((float(a1_transSR)+float(a1_insCONV)+float(a1
_insINFRA)+float(a1_gapCONV)))
                        if asptemp[x-1] == '0':
                            a1_gaptemp.append('0')
                        else:
                            a1_gaptemp.append(gapconv_gaptemp(a1_gapCONV,iattemp[x-
1],asptemp[x-1]))
                    if asttemp[x-1] == 'exterior air curtain':
                        a1_shgc.append((float(a1_transSR)+float(a1_insCONV)+float(a1
_insINFRA)))
                        if asptemp[x-1] == '0':
                            a1_gaptemp.append('0')
                        else:
                            a1_gaptemp.append(gapconv_gaptemp(a1_gapCONV,dbttemp[x-
1],asptemp[x-1]))
                    if asttemp[x-1] == 'supply mode':
                        a1_shgc.append((float(a1_transSR)+float(a1_insCONV)+float(a1
_insINFRA)+float(a1_gapCONV))-supply_extra(dbttemp[x-1],iattemp[x-1],asptemp[x-
1],a1_gapCONV))
                        if asptemp[x-1] == '0':
                            a1_gaptemp.append('0')
                        else:
                            a1_gaptemp.append(gapconv_gaptemp(a1_gapCONV,dbttemp[x-
1],asptemp[x-1]))
                    if asttemp[x-1] == 'extract mode':
                        a1_shgc.append((float(a1_transSR)+float(a1_insCONV)+float(a1
_insINFRA))-extract_extra(dbttemp[x-1],iattemp[x-1],asptemp[x-1]))
                        if asptemp[x-1] == '0':
                            a1_gaptemp.append('0')
```

```
                        else:
                            a1_gaptemp.append(gapconv_gaptemp(a1_gapCONV,iattemp[x-1],asptemp[x-1]))

                    a1_vlt.append(float(a1_transSR)*105)
                    a1_deltatemp.append(float(dbttemp[x-1])-float(iattemp[x-1]))


        with open('cases.csv','r') as csvinput:
            with open('cases/cases_glaz'+str(casenum)+'_collected_final.csv', 'w') as csvoutput:
                writer = csv.writer(csvoutput, lineterminator='\n')
                reader = csv.reader(csvinput)

                all = []
                x=0
                for row in reader:
                    row.append(a1_outSRa[x])
                    row.append(a1_transSRa[x])
                    row.append(a1_gapCONVa[x])
                    row.append(a1_insCONVa[x])
                    row.append(a1_insINFRAa[x])
                    row.append(a1_shgc[x])
                    row.append(a1_vlt[x])
                    row.append(a1_gaptemp[x])
                    row.append(a1_deltatemp[x])

                    all.append(row)
                    x=x+1
                writer.writerows(all)
```

Plotting 2D graphs

```
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
import numpy as np
import matplotlib as mpl
import csv
import math
from scipy import stats
import sys

#casenum = int(sys.argv[1])
casenum = 2
casenames=['DGU-200-Single','Single-200-DGU','DGU-200-DGU','DGU-400-Single','Single-400-DGU','DGU-400-DGU','DGU-600-Single','Single-600-DGU','DGU-600-DGU']

runmode = ['Path','Flowrate','Slatangle']
#modetorun = runmode[int(sys.argv[2])-1]
modetorun = runmode[2]

print(casenames[casenum-1])

print(modetorun)

nam =[]
deltaT=[]
iat=[]
sr=[]
bst=[]
ban=[]
asp=[]
```

```
ast=[]
ref=[]
thm1=[]
vis1=[]

def uploadfile(num):

    with open('../DSFs/cases_glaz'+str(num)+'_collected_final.csv', newline='') as
f:
        reader = csv.reader(f)
        data = []
        for row in reader:
            data.append(row)
        size=len(data)

        for i in range(1,size):
            nam.append(data[i][0])
            iat.append(int(data[i][4]))
            deltaT.append(int(data[i][21]))
            sr.append(data[i][5])

            bst.append(data[i][6])
            ban.append(data[i][7])
            asp.append(data[i][8])
            ast.append(data[i][9])
            ref.append(data[i][12])

            thm1.append(round((float(data[i][18])/1000),2))
            vis1.append(round((float(data[i][19])/1000),2))

        print(len(nam),nam[len(nam)-1],nam[0],thm1[0])

def min_function(somelist):
    min_value = None
    for value in somelist:
        if not min_value:
            min_value = value
        elif value < min_value:
            min_value = value
    return min_value

def min_x_on_maxyaxis(listx,listy,maxy):
    min_value = None
    for x in range(0,len(listx)):
        if listy[x] == maxy:
            if not min_value:
                min_value = listx[x]
            elif listx[x] < min_value:
                min_value = listx[x]
    return min_value

def max_x_on_minyaxis(listx,listy,miny):
    min_value = None
    for x in range(0,len(listx)):
        if listy[x] == miny:
            if not min_value:
                min_value = listx[x]
            elif listx[x] > min_value:
                min_value = listx[x]
    return min_value

def max_function(somelist):
    max_value = None
    for value in somelist:
        if not max_value:
            max_value = value
```

```
        elif value > max_value:
            max_value = value
    return max_value

def PolygonArea(corners):
            n = len(corners) # of corners
            area = 0.0
            for i in range(n):
                j = (i + 1) % n
                area += corners[i][0] * corners[j][1]
                area -= corners[j][0] * corners[i][1]
            area = abs(area) / 2.0
            return area

       def color_datapoint(x):
            airspeed = ['0.088888889','0.044444444','0.022222222','0.011111111','0.005555556','0.002777778']
            markert = ["^","v","1","s","P","x"]
            for aa in range(0,len(airspeed)):
                if asp[x] == airspeed[aa]:
                    airmode=['supply mode','exterior air curtain','interior air curtain','extract mode']
                    alphass = [0.4,0.6,0.8,1]

                    for y2 in range(0,len(airmode)):
                        if ast[x] == airmode[y2]:
                            plt.scatter(thm1[x], vis1[x], 10, color='black', marker='o' , alpha=alphass[y2])


                    for y1 in range(0,len(airmode)):
                        if ast[x] == airmode[y1]: #delta changes aplpha(transp) of marker
                            slatang = ['30','45','60','90','120','135','150']#['45','90']
                            colors = ['blue','green','red','cyan','magenta','gold','orange']#['blue','green']
                            for z1 in range(0,len(slatang)):
                                if ban[x] == slatang[z1]:
                                    plt.scatter(thm1[x], vis1[x], 10, color=colors[z1], marker=markert[aa], alpha=alphass[y1])


       def allfunction2(name,visual,thermal,reflec):  #all combos of speed/mode/iat

            airspeed = ['0.088888889','0.044444444','0.022222222','0.011111111','0.005555556','0.002777778']
            airspeednm = ['0.0889','0.0444','0.0222','0.0111','0.0055','0.0028'] #aa

            airmode=['extract mode','supply mode','exterior air curtain','interior air curtain']#x11
            airmodenm =['EX','SU','EC','IC']

            slatang = ['30','45','60','90','120','135','150']#z1

            deltaTemp = [-45,-40,-35,-30,-25,-20,-15,-10,-5,0,5,10,15,20]                                    #y1,2
            alphass = [0.1,0.17,0.24,0.31,0.38,0.45,0.52,0.58,0.65,0.72,0.79,0.86,0.93,1]

            solarrad = [0,2000,4000,6000,8000,10000]
                        #x1,2
            size = [2,20,40,80,160,320]

```

```
147.
148.           if modetorun == 'Path':
149.               colors = ['blue','cyan','red','green']
   #AirflowMode            #x11
150.
151.           if modetorun == 'Slatangle':
152.               colors = ['blue','green','red','cyan','magenta','gold','grey']
   #SlatAngle          #z1
153.
154.           if modetorun == 'Flowrate':
155.               colors = ['blue','green','red','cyan','magenta','gold']
   #AirflowSpeed           #aa
156.
157.           fig = plt.figure(figsize=(10,5))
158.
159.           for x11 in range(0,len(airmode)):
160.           #for x11 in run:
161.               for x in range(0,len(nam)):
162.                   for aa in range(0,len(airspeed)):
163.                   #for aa in run:
164.                       if (int(iat[x]) == 25 or int(iat[x]) == 25) and  (asp[x] ==
   airspeed[aa] or asp[x] == '0') and ast[x] ==airmode[x11] :
165.                           for y1 in range(0,len(deltaTemp)):
166.                           #for y1 in rundt:
167.                               if int(deltaT[x]) == deltaTemp[y1]:
168.                                   for x1 in range(0,len(solarrad)):
169.                                   #for x1 in runsr:
170.                                       if int(sr[x]) == solarrad[x1]:
171.                                           for z1 in range(0,len(slatang)):
172.                                           #for z1 in run:
173.                                               if ban[x] == slatang[z1] and ref[x]
   == reflec:
174.                                                   if modetorun == 'Path':
175.                                                       if asp[x] == '0':
176.                                                           #plt.scatter(thermal[x],
    visual[x], size[x1], color='black', marker=".", alpha=alphass[y1])     #AirflowMod
   e        #x11
177.                                                           plt.scatter(thermal[x],
   visual[x], 20, color='black', marker=".", alpha=1)      #AirflowMode        #x11
178.
179.                                                       else :
180.                                                           #plt.scatter(thermal[x],
    visual[x], size[x1], color=colors[x11], marker=".", alpha=alphass[y1])     #Airflo
   wMode        #x11
181.                                                           plt.scatter(thermal[x],
   visual[x], 20, color='black', marker=".", alpha=1)      #AirflowMode        #x11
182.
183.
184.                                                   if modetorun == 'Slatangle':
185.                                                       plt.scatter(thermal[x], visu
   al[x], size[x1], color=colors[z1], marker=".", alpha=alphass[y1])       #SlatAngle
           #z1
186.
187.                                                   if modetorun == 'Flowrate':
188.                                                       if  asp[x] == '0':
189.                                                           plt.scatter(thermal[x],
   visual[x], size[x1], color='black', marker=".", alpha=alphass[y1])      #AirflowSpe
   ed       #aa
190.                                                       else :
191.                                                           plt.scatter(thermal[x],
   visual[x], size[x1], color=colors[aa], marker=".", alpha=alphass[y1])       #Airflo
   wSpeed       #aa
192.
193.                           for y2 in range(0,len(deltaTemp)):
194.                           #for y2 in rundt:
195.                               if int(deltaT[x]) == deltaTemp[y2]:
```

```
                                    for x2 in range(0,len(solarrad)):
                                    #for x2 in runsr:
                                        if int(sr[x]) == solarrad[x2] and ban[x] == '-':
                                            if modetorun == 'Path':

                                                if asp[x] == '0':
                                                    plt.scatter(thermal[x], visual[x], size[x2], color='black', marker='.', alpha=alphass[y2])      #AirflowMode          #x11
                                                    #plt.scatter(thermal[x], visual[x], 20, color='black', marker=".", alpha=1)     #AirflowMode        #x11
                                                else :
                                                    plt.scatter(thermal[x], visual[x], size[x2], color=colors[x11], marker='.', alpha=alphass[y2])      #AirflowMode          #x11
                                                    #plt.scatter(thermal[x], visual[x], 20, color='black', marker=".", alpha=1)     #AirflowMode        #x11

                                            if modetorun == 'Slatangle':
                                                plt.scatter(thermal[x], visual[x], size[x2], color='black', marker='.', alpha=alphass[y2])      #SlatAngle          #z1

                                            if modetorun == 'Flowrate':
                                                if asp[x] == '0':
                                                    plt.scatter(thermal[x], visual[x], size[x2], color='black', marker='.', alpha=alphass[y2])      #AirflowSpeed          #aa
                                                else :
                                                    plt.scatter(thermal[x], visual[x], size[x2], color=colors[aa], marker='.', alpha=alphass[y2])       #AirflowSpeed          #aa


            plt.grid(b=True, which='major', color='#666666', linestyle='-')
            plt.minorticks_on()
            plt.grid(b=True, which='minor', color='#999999', linestyle='-', alpha=0.2)

            plt.title(name+'_with '+reflec+' Reflectivity Slat_'+'allmode'+'_allspeed_25C IAT')

            plt.xlabel('Thermal Metrics [kW/sqm]')
            plt.ylabel('Visual Metrics [kLm/sqm]')

            plt.axis([-4.5, 4, 0, 40])


            if modetorun == 'Path':

                plt.text(2, 32, 'SolarRadiation: 1000 W/sqm', fontsize=9, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
                plt.text(2, 30, 'deltaT: -45 C', fontsize=9, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
                #plt.gca().add_patch(plt.Rectangle((1.95, 29.5),2,4, fill=True, edgecolor='black', facecolor='None', linewidth=1, alpha=1,zorder=20))

                plt.text(2, 26, 'Airflow Path : Colors', fontsize=9, fontweight='medium', bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
                plt.text(2, 24, 'Air Extract         = Blue', color='blue', fontsize=8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
                plt.text(2, 22, 'Air Supply          = Cyan', color='cyan', fontsize=8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
```

```
239.                 plt.text(2, 20, 'Outdoor Air Curtain = Red', color='red', fontsize=8
    , bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
240.                 plt.text(2, 18, 'Indoor Air Curtain  = Green', color='green', fontsi
    ze=8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
241.                 plt.text(2, 16, 'Air Buffer = Black', color='Black', fontsize=8, bbo
    x=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
242.                 plt.gca().add_patch(plt.Rectangle((1.95, 15),2,12.5, fill=True, edge
    color='black', facecolor='None', linewidth=1, alpha=1,zorder=20))
243.
244.             if modetorun == 'Slatangle':
245.
246.                 plt.text(2, 32, 'SolarRadiation: Marker Size', fontsize=9, bbox=dict
    (facecolor='white',edgecolor='white', alpha=1,zorder=10))
247.                 plt.text(2, 30, 'deltaT: Opacity of color', fontsize=9, bbox=dict(fa
    cecolor='white',edgecolor='white', alpha=1,zorder=10))
248.                 #plt.text(2, 28,  'No Blinds : Black dots', color='Black', fontsize=
    9, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
249.                 plt.gca().add_patch(plt.Rectangle((1.95, 29.5),2,4, fill=True, edgec
    olor='black', facecolor='None', linewidth=1, alpha=1,zorder=20))
250.
251.                 plt.text(2, 26, 'Slat Angle : Colors', fontsize=9, fontweight='mediu
    m', bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
252.                 plt.text(2, 24, '30deg     = Blue', color='blue', fontsize=8, bbox=d
    ict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
253.                 plt.text(2, 22, '45deg     = Green', color='green', fontsize=8, bbox
    =dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
254.                 plt.text(2, 20, '60deg     = Red', color='red', fontsize=8, bbox=dic
    t(facecolor='white',edgecolor='white', alpha=1,zorder=10))
255.                 plt.text(2, 18, '90deg     = Cyan', color='cyan', fontsize=8, bbox=d
    ict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
256.                 plt.text(2, 16, '120deg    = Magenta', color='magenta', fontsize=8,
    bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
257.                 plt.text(2, 14,  '135deg    = Yellow', color='gold', fontsize=8, bbo
    x=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
258.                 plt.text(2, 12,  '150deg    = Grey', color='grey', fontsize=8, bbox=
    dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
259.                 plt.text(2, 10,  'No Blinds = Black', color='Black', fontsize=8, bbo
    x=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
260.                 plt.gca().add_patch(plt.Rectangle((1.95, 9),2,18.5, fill=True, edgec
    olor='black', facecolor='None', linewidth=1, alpha=1,zorder=20))
261.
262.             if modetorun == 'Flowrate':
263.                 plt.text(2, 32, 'SolarRadiation: Marker Size', fontsize=9, bbox=dict
    (facecolor='white',edgecolor='white', alpha=1,zorder=10))
264.                 plt.text(2, 30, 'deltaT: Opacity of color', fontsize=9, bbox=dict(fa
    cecolor='white',edgecolor='white', alpha=1,zorder=10))
265.                 #plt.text(2, 28,  'No Blinds : Black dots', color='Black', fontsize=
    9, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
266.                 plt.gca().add_patch(plt.Rectangle((1.95, 29.5),2,4, fill=True, edgec
    olor='black', facecolor='None', linewidth=1, alpha=1,zorder=20))
267.
268.                 plt.text(2, 26, 'Airflow Speed : Colors', fontsize=9, fontweight='me
    dium', bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
269.                 plt.text(2, 24, '0.0889 m3/s.m     = Blue', color='blue', fontsize=8
    , bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
270.                 plt.text(2, 22, '0.0444 m3/s.m     = Green', color='green', fontsize
    =8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
271.                 plt.text(2, 20, '0.0222 m3/s.m     = Red', color='red', fontsize=8,
    bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
272.                 plt.text(2, 18, '0.0111 m3/s.m     = Cyan', color='cyan', fontsize=8
    , bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
273.                 plt.text(2, 16, '0.0055 m3/s.m    = Magenta', color='magenta', fonts
    ize=8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
274.                 plt.text(2, 14, '0.0028 m3/s.m    = Yellow', color='gold', fontsize=
    8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
```

```
                plt.text(2, 12, '0 m3/s.m         = Black', color='Black', fontsize=8, bbox=dict(facecolor='white',edgecolor='white', alpha=1,zorder=10))
                plt.gca().add_patch(plt.Rectangle((1.95, 11),2,16.5, fill=True, edgecolor='black', facecolor='None', linewidth=1, alpha=1,zorder=20))

            if modetorun == 'Path':
                plt.savefig('full/AirflowPath_'+name+'_with '+reflec+' Reflectivity Slat_'+'allmode'+'_'+'allspeed'+'_25C IAT.png',dpi=800) #AirflowMode        #x11
                #plt.savefig('full/Path_'+str(int(sys.argv[1]))+'_.png',dpi=800)#AirflowSpeed       #aa

            if modetorun == 'Slatangle':
                plt.savefig('full/SlatAngle_'+name+'_with '+reflec+' Reflectivity Slat_'+'allmode'+'_'+'allspeed'+'_25C IAT.png',dpi=800)   #SlatAngle          #z1
                #plt.savefig('full/SlatAngle_'+str(int(sys.argv[1]))+'_.png',dpi=800)

            if modetorun == 'Flowrate':
                plt.savefig('full/AirflowSpeed_'+name+'_with '+reflec+' Reflectivity Slat_'+'allmode'+'_'+'allspeed'+'_25C IAT.png',dpi=800)#AirflowSpeed       #aa
                #plt.savefig('full/AirflowSpeed_'+str(int(sys.argv[1]))+'_.png',dpi=800)

            plt.close(fig)

            print(name+'_with '+reflec+' Reflectivity Slat_'+'allmode'+'_allspeed'+'_25C IAT.png')

        uploadfile(casenum)
        #allfunction2(casenames[casenum-1],vis1,thm1,'High')
        allfunction2(casenames[casenum-1],vis1,thm1,'Low')
```

Plotting 3D graphs

```
import csv
import glob
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from scipy import interpolate
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from mpl_toolkits.axes_grid1 import make_axes_locatable
import sys
from PIL import Image
import numpy as np
import PIL

from PIL import Image
from PIL import ImageOps
from PIL import ImageDraw

import imageio

dt=[]
sr=[]
x1pt=[]
x2pt=[]
x3pt=[]
x4pt=[]
y1pt=[]
y2pt=[]
y3pt=[]
y4pt=[]
'''''
```

```
2       3               4               5           6       7           8   9   10
    11  12  13  14  15  16          17              18              19
     20
Area    Side1_visual    Side2_thermal   Perimeter   Pearson Spearman    x1  x2  x3
    x4  y1  y2  y3  y4  Area_RANK   Pearson_RANK    Spearman_RANK   delta visual_Rank
     delta thermal_Rank

delta of thermal metrics=side2

delta of visual metrics=side1

thermal =x pts  Thermal Metrics [kW/sqm]
visual =y pts  Visual Metrics [kLm/sqm]


        x2,y2-------x3,y3
         /           /
      x1,y1--------x4,y4


Spearman Correlation Coefficient
Pearson Correlation Coefficient
'''

def combineimages(number):
    images = list(map(Image.open, ['movie1/'+str(number)+'_'+str(nameright[0])+'_movie.png','movie1/'+str(number)+'_'+str(nameright[1])+'_movie.png','movie1/'+str(number)+'_'+str(nameright[2])+'_movie.png']))

    widths, heights = zip(*(i.size for i in images))

    total_width = sum(widths)
    max_height = max(heights)

    new_im = Image.new('RGB', (total_width, max_height))

    x_offset = 0
    for im in images:
      new_im.paste(im, (x_offset,0))
      x_offset += im.size[0]

    new_im.save('test_High.png',quality=100)

    images = list(map(Image.open, ['movie1/'+str(number)+'_'+str(nameright[3])+'_movie.png','movie1/'+str(number)+'_'+str(nameright[4])+'_movie.png','movie1/'+str(number)+'_'+str(nameright[5])+'_movie.png']))

    widths, heights = zip(*(i.size for i in images))

    total_width = sum(widths)
    max_height = max(heights)

    new_im = Image.new('RGB', (total_width, max_height))

    x_offset = 0
    for im in images:
      new_im.paste(im, (x_offset,0))
      x_offset += im.size[0]

    new_im.save('test_Low.png',quality=100)


    list_im = ['test_High.png', 'test_Low.png']
    imgs    = [ PIL.Image.open(i) for i in list_im ]
    # pick the image which is the smallest, and resize the others to match it (can be arbitrary image shape here)
```

```
    min_shape = sorted( [(np.sum(i.size), i.size ) for i in imgs])[0][1]

    # for a vertical stacking it is simple: use vstack
    imgs_comb = np.vstack( (np.asarray( i.resize(min_shape) ) for i in imgs ) )
    imgs_comb = PIL.Image.fromarray( imgs_comb)
    imgs_comb.save( 'movie1/Final'+str(number)+'.png' )
    im = Image.open('movie1/Final'+str(number)+'.png')
    d = ImageDraw.Draw(im)
    left = (0, 540)
    right = (1980, 540)
    line_color = (0, 0, 0)
            d.line([left, right], fill=line_color, width=4)
            im.save('movie1/Final'+str(number)+'.png')

        solarrad = [0,200,400,600,800,1000]
        deltaT = [-45,-40,-35,-30,-25,-20,-15,-10,-5,0,5,10,15,20]

        name = ['glaz1_High','glaz2_High','glaz3_High','glaz1_Low','glaz2_Low','glaz
    3_Low']
        nameDSF=['DGU-200-Single','Single-200-DGU','DGU-200-DGU','DGU-200-
    Single','Single-200-DGU','DGU-200-DGU']
        namerefl=['High','High','High','Low','Low','Low']
        nameright=['DGU-200-Single_High','Single-200-DGU_High','DGU-200-
    DGU_High','DGU-200-Single_Low','Single-200-DGU_Low','DGU-200-DGU_Low']

        for i in range(0,len(name)):
            with open('csv outputs/dT_SR data_'+str(name[i])+'.csv', newline='') as
    f:
                temp1=[]
                temp2=[]
                temp3=[]
                temp4=[]
                temp5=[]
                temp6=[]
                temp7=[]
                temp8=[]
                temp9=[]
                temp10=[]
                reader = csv.reader(f)
                data = []
                for row in reader:
                    data.append(row)
                size=len(data)
                for i in range(1,size):
                    temp1.append(float(data[i][0]))
                    temp2.append(float(data[i][1])) #'Solar radiation [kW/sqm]'
                    temp3.append(float(data[i][7]))
                    temp4.append(float(data[i][8]))
                    temp5.append(float(data[i][9]))
                    temp6.append(float(data[i][10]))
                    temp7.append(float(data[i][11]))
                    temp8.append(float(data[i][12]))
                    temp9.append(float(data[i][13]))
                    temp10.append(float(data[i][14]))
                dt.append(temp1) #'deltaT [C]'
                sr.append(temp2) #'Solar radiation [kW/sqm]'
                x1pt.append(temp3)
                x2pt.append(temp4)
                x3pt.append(temp5)
                x4pt.append(temp6)
                y1pt.append(temp7)
                y2pt.append(temp8)
                y3pt.append(temp9)
                y4pt.append(temp10)

        print(len(dt))
```

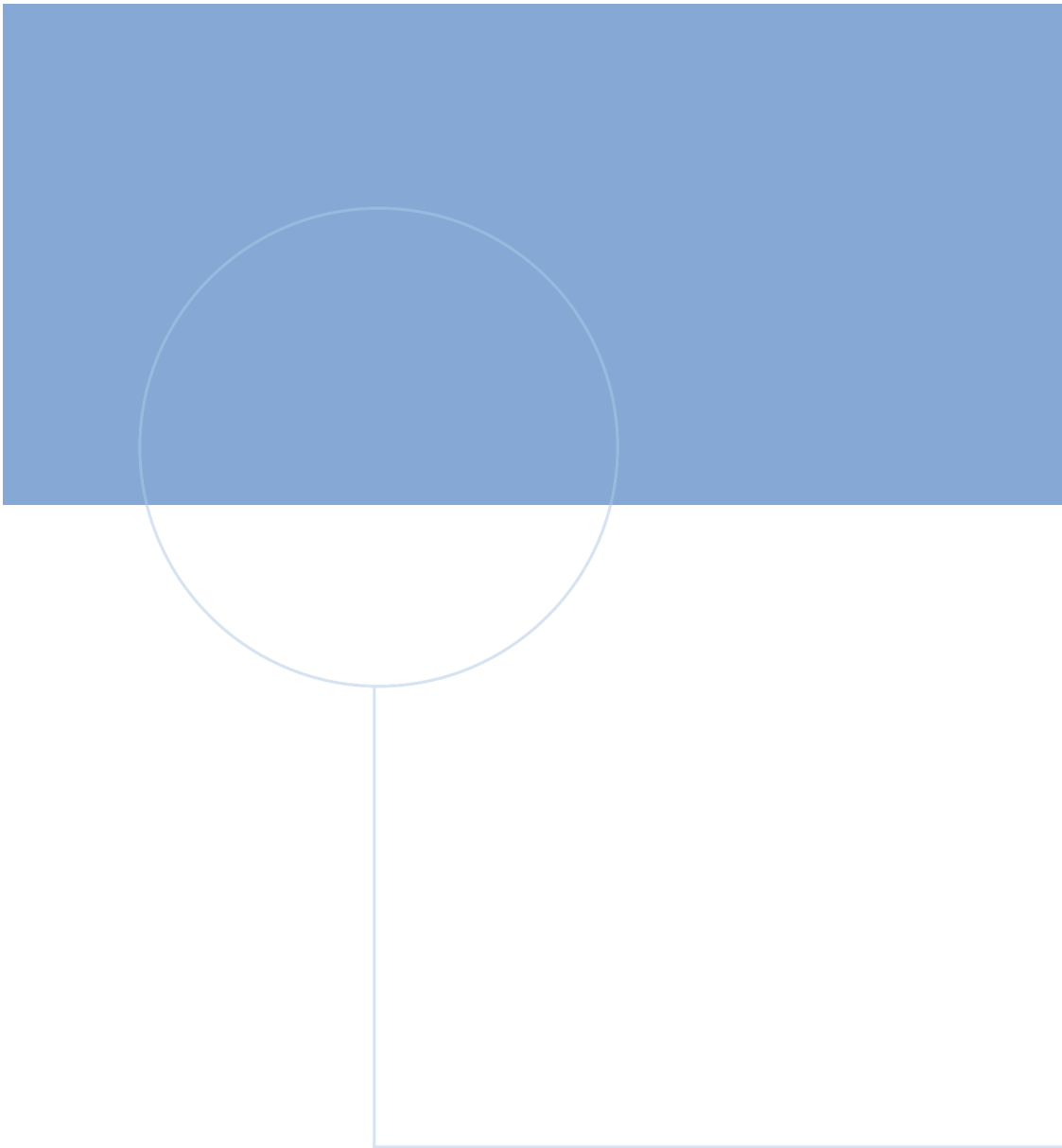
```
151.
152.       fig = plt.figure(figsize=(10,5))
153.       ax = Axes3D(fig)
154.
155.       def make3Dsurface(x,y,z,title1,color,xlabel,ylabel,zlabel,knum):
156.           surf = ax.scatter(x,y,z, c=color,s=2,marker='o',alpha =0.5,linewidth=1,
    antialiased=True)
157.           #surf = ax.plot_trisurf(x,y,z, cmap=plt.get_cmap('Blues'), linewidth=5,
    antialiased=True)
158.           surf = ax.plot(x,y,z, c=color,alpha =0.5,linewidth=1, antialiased=True)

159.           #ax.scatter(xs, ys, zs, c=c, marker=m)
160.           ax.set_ylabel(ylabel,fontsize=10,fontweight='bold')
161.           ax.set_xlabel(xlabel,fontsize=10,fontweight='bold')
162.           ax.set_xlim(0, 1000) #thermal
163.           ax.set_ylim(-50, 20) #visual
164.           #ax.set_zlim(-4.5,2)
165.           ax.set_zlim(0,30)
166.
167.           ax.zaxis.set_rotate_label(True)
168.           ax.set_zlabel(zlabel, rotation = 90,fontsize=10,fontweight='bold')
169.           alltext=ax.texts
170.           for t in alltext:
171.               t.set_visible(False)
172.           ax.text2D(0.05, 0.95, title1+' Reflectivity Slats', color='red', transfo
    rm=ax.transAxes, fontsize=10,fontweight='bold')
173.
174.
175.
176.           #plt.savefig(title+'3D.png')
177.           #plt.show()
178.           #plt.close(fig)
179.
180.
181.       def make3dplots11(test):
182.           for k in test:
183.               for j in range(0,len(deltaT)):
184.                   zaxis_thm=[]
185.                   zaxis_vis=[]
186.                   xaxis=[]
187.                   yaxis=[]
188.                   for i in range(0,len(dt[k])):
189.                       if dt[k][i]==deltaT[j]:
190.                           #zaxis_thm.append(x1pt[k][i])
191.                           zaxis_thm.append(x2pt[k][i])
192.                           zaxis_thm.append(x3pt[k][i])
193.                           #zaxis_thm.append(x4pt[k][i])
194.                           zaxis_thm.append(x2pt[k][i])
195.
196.                           zaxis_vis.append(y1pt[k][i])
197.                           zaxis_vis.append(y2pt[k][i])
198.                           #zaxis_vis.append(y3pt[k][i])
199.                           #zaxis_vis.append(y4pt[k][i])
200.                           zaxis_vis.append(y1pt[k][i])
201.
202.                           xaxis.extend([sr[k][i],sr[k][i],sr[k][i]])
203.                           yaxis.extend([dt[k][i],dt[k][i],dt[k][i]])
204.                   #make3Dsurface(xaxis,yaxis,zaxis_thm,str(nameright[k]),'blue','S
    olar radiation [W/sqm]','Delta Temperature [°C]','Thermal Metrics [kW/sqm]',k)
205.                   make3Dsurface(xaxis,yaxis,zaxis_vis,str(nameright[k]),'blue','So
    lar radiation [W/sqm]','Delta Temperature [°C]','Visual Metrics [kLm/sqm]',k)
206.                   ax.view_init(elev=22, azim=80)
207.                   '''''
208.                   plt.savefig('movie1/'+str(flag+1)+'_'+str(nameright[k])+'_movie.
    png')
209.
```

```
                    img = Image.open('movie1/'+str(flag+1)+'_'+str(nameright[k])+'_movie.png')
                    img_with_border = ImageOps.expand(img,border=1,fill='black')
                    img_with_border.save('movie1/'+str(flag+1)+'_'+str(nameright[k])+'_movie.png')
                    '''

                #plt.show()
                plt.savefig('movie1/Vis_'+str(nameright[k])+'_movie.png')
                plt.cla()

        def make3dplots(test):
            for k in test:
                flag=0
                for i in range(0,len(dt[k])):
                    zaxis_thm=[x1pt[k][i],x2pt[k][i],x3pt[k][i],x4pt[k][i]]
                    zaxis_vis=[y1pt[k][i],y2pt[k][i],y3pt[k][i],y4pt[k][i]]
                    xaxis=[sr[k][i],sr[k][i],sr[k][i],sr[k][i]]
                    yaxis=[dt[k][i],dt[k][i],dt[k][i],dt[k][i]]
                    make3Dsurface(xaxis,yaxis,zaxis_thm,str(nameright[k]),'blue','Solar radiation [W/sqm]','Delta Temperature [°C]','Thermal Metrics [kW/sqm]',k)
                    #make3Dsurface(xaxis,yaxis,zaxis_vis,str(int(dt[k][i])),'blue','Solar radiation [W/sqm]','Delta Temperature [°C]','Visual Metrics [kLm/sqm]',k)
                    ax.view_init(elev=18, azim=54)
                    '''''
                    plt.savefig('movie1/'+str(flag+1)+'_'+str(nameright[k])+'_movie.png')

                    img = Image.open('movie1/'+str(flag+1)+'_'+str(nameright[k])+'_movie.png')
                    img_with_border = ImageOps.expand(img,border=1,fill='black')
                    img_with_border.save('movie1/'+str(flag+1)+'_'+str(nameright[k])+'_movie.png')
                    '''
                    flag=flag+1

                plt.show()
                plt.cla()


        #high=[0,1,2]
        #test=[3,4,5]
        #low=[3,4,5]
        make3dplots11([2])
        #make3dplots11([0,1,2])
        #make3dplots11([3,4,5])


        def gifmaker_indi(test,count):
            for k in test:
                #for x in range(count):
                    #combineimages(x+1)
                images = []
                for i in range(count):
                    images.append(imageio.imread('movie1/'+str(i+1)+'_'+str(nameright[k])+'_movie.png'))
                imageio.mimsave('F_'+str(nameright[k])+'_movie1.gif', images,loop=0)


        def gifmaker_all(count):
            for x in range(count):
                combineimages(x+1)
            images = []
            for i in range(count):
                images.append(imageio.imread('movie1/Final'+str(i+1)+'.png'))
            imageio.mimsave('all_movie1.gif', images,loop=0)
```

```
267.
268.
269.
270.        #gifmaker_indi([0,1,2],115)
271.        #gifmaker_indi([3,4,5],115)
272.        #gifmaker_all(count)
```

For making video from data

```
1.  import cv2
2.  import os
3.  import imageio
4.
5.  video_name = 'mp444.avi'
6.  #fourcc = cv2.VideoWriter_fourcc(*'DIVX')
7.  images = []
8.  flag=115
9.  casenum =2
10. for i in range(flag):
11.     images.append('movie/f'+str(casenum)+'/'+str(i+1)+'_movie.png')
12. frame = cv2.imread(images[0])
13. height, width, layers = frame.shape
14.
15. video = cv2.VideoWriter(video_name, 0, 10,(width,height))
16.
17. for image in images:
18.     video.write(cv2.imread(image))
19.
20. cv2.destroyAllWindows()
21. video.release()
```



NTNU
Norwegian University of
Science and Technology