POLITECNICO DI TORINO

Master Degree in 'ICT for Smart societies'

POLITECNICO
DI TORINO

Sky View Factor Estimation Based on Deep Learning and Big Data

Candidate: Mat:

Zhang Bolun **S250667**

Supervisor:

Prof. Giacomo Chiesa

Academic year 2021/2022

Abstract:

In urban climate studies, the radiative exchange component is essential, hence urban geometry has been investigated in a view factor framework for decades. The sky view factor (SVF) is a commonly used 3-d indicator and represents the fraction of the overlying hemisphere occupied by the sky. It can be calculated by hemispheric pictures taken by cameras with fish-eye lenses. In this thesis a new code has been developed to define SVF based on 360-degree street view through produced the coordinate processing of the panoramic image which is an effect that can be comparable to the output obtained from traditional fish-eye camera. At the same time, deep learning is used to segment the panoramic image to automatically select SVF and other objects to overlap resulting solar paths for the given location and analyze solar radiation hours. Each step is defined and verified in this thesis. Additionally, initial sample application results are shown.

contents

Introduction:

The sky view factor (SVF) describes a point to point (point to surface) visibility indice and is a commonly used and important measure in urban climate investigations whose aim is the exploration of effects of a complex urban surface on climatological processes in built-up areas. A SVF represents the ratio of the visible sky at a hemisphere center over a specific location. The measure of visibility is generally applied to analyze the energy exchange considering radiation exchange.

With the improvement of speed and accuracy of the SVF calculation, its application is becoming gradually popular. For example, Yamashita et al. emonstrated the link between Sky View Factor and local temperature. By this way, the SVF usually contribues to analyze the urban Heat Island and urban

environment. Then, the SVF is involved into the outdoor comfort indices. The Universal Effective Temperature(ETU) and the Index of Thermal Stress(ITS) are seen as outdoor comfort indices that use SVF in outdoor parameter. Besides help at the studies of urban heat island, the SVF also is used to analyze the river bed morphology to determine the vegetation and water characteristics(Bartnik & Moniewski, 2011). The prediction of urban temperature considers the SVF as well. Moreover, the city structure can be improved obviously in the simulation with SVF(de Morais et al., 2018).

The first calculation of SVF was with the help of a digital camera equipped with a fisheye lens, and then the image was drawn manually. Some pixel counting methods based on grating were developed so as to increase the efficiency and accuracy of the calculation. Generally, Sky-view factors estimation is based on a 3D-GIS extension through using the height information of the surrounding buildings at a specific location to build a 3-D model. SkyHelios has developed into a mature software. According to the data obtained by Digital Surface Models (DSMs) through data sources, it returns to virtual fisheye images. A new vector-based SVF calculation tool was implemented in a free and open source Geographic Information System named OrbisGIS.

Although DSM and 3D-GIS have high accuracy, because the acquisition of information is very complicated, it brings a certain degree of difficulty to the calculation of the sky factor. ‘In the absence of urban morphological

information, fisheye photography has frequently been employed to calculate the SVF for discrete locations' (Grimmond et al., 2001; Chapman and Thornes, 2004; Ali-Toudert and Mayer, 2007). Tsuyoshi Honjo et al. have proved that using spherical camera sampling and using fisheye camera to obtain the sky factor are very similar in accuracy. Even if good results can be obtained with the aid of a fisheye camera, the workload of the fisheye camera will greatly increase in the face of numerous sampling points in the city.

At present, people's demand for data is greatly improved, which can not be met by physical equipment and simulation. In the future, SVF will be potentially used as basic information to provide research in various aspects of the city, such as the calculation of the heat island effect of the entire city, and the calculation of the light energy of a certain building in the city.To develop a technology for widely obtaining amount of SVF, the article focus on use big data, image processing technology and python tools to establish an environment that can complete the data&image collection, calculate and analyze of the sky view factor through computer only. Meanwhile, it will also be compared with the traditional method to prove its feasibility.

In order to achieve the goal, in the section 2.1, the thesis introduces the current calculation formula of SVF based on pixel in fisheye images, and explain the popular image segmentation technology referenced in image processing, which play the role of the basis theory of the whole thesis.

In section 3, the thesis explain the whole experimental method. The first is the acquisition of images, including how to obtain panoramic images from Google maps and what the format of the acquired images looks like. The next step is the preprocessing of Google images so that the output is satisfied to the input format of image segmentation. Then, the segmented image need to be transformed into fish eye image for the next SVF analysis. In the end, the SVF will be showed as different formats and also be added into the solar radiation estimation.

The relative position of the Sun is a major factor in the heat gain of buildings and in the performance of solar energy systems. Accurate location-specific knowledge of sun path and climatic conditions is essential for economic decisions about solar collector area, orientation, landscaping, summer shading, and the cost-effective use of solar trackers. By locating where the sun is visible in the canyon, the light intensity can be obtained. This helps to analyze the comfort of the streets in the city, and can also obtain suggestions for the orientation and location of solar panels.

Therefore, the system not only calculates the value of SVF for a specific location, but also tries to analyze the sun path and solar radiation of a specific location. This will contribute to a detail information of the location in the city.

2. State of the art

2.1. Sky view factor calculating models

The Lambert's cosine law defines that there is no change on surface of 10km radius hemisphere. Furthermore, people’s view can not extend to the radius of ten kilometers in the city, especially in the canyon. In this way, receptive field of view is similar to the plane in the cos law, which represents the scene within our field of vision when we look up at the sky 90 degrees. The SVF is dimensionless, and its valid range is from 0 to 1, where 0 represents absolutely no sky seen from the target point (e.g., a deep valley or a highly dense forest) and 1 stands for complete visibility of the sky for the entire hemisphere (e.g., a vast plain or the peak of a mountain).

Several methods are developed to calculate the SVF using different data sources. The first model based on the fisheye image, which is processed graphically. According to Steyn (1980), the SVF is computed by counting the pixel processed from fisheye image who is divided into pixel corresponding to the polar. However, many applications to calculate SVF are complex and may produce large error. One method is to create the outline of 3D buildings in the hemisphere to figure out SVF, which commonly called Vector-based method. Another one is raster-based method, the graphics are processed by computer and calculated with data of digital elevation model (DEM) or digital surface model (DSM).

At present, the development and analysis tools based on fisheye images are very mature (e.g., the SkyHelios model), "the Solar Longwave Environmental irradiation Geometry model" (Lindberg et al., 2008), "the RayMan model" (Matzarakis et al., 2010), and "the ArcView SVF extension" (Gál et al., 2009)). For example, SkyHelios is originally proposed to calculate SVF based on the statistics of pixels in the picture, however, it only divided the non sky and sky pixels in the picture without considering the influence of the angle in the fisheye picture. This simple method makes the results inaccurate (Zhong-Hu Jiao et al., 2019). In addition, the complex method and model is used by involving the obstruct such as trees, by this way, the model comes to real when simulating the urban environment.

The result of SVF figured out from the models (i.e., SkyHelios, ArcView SVF extension, and Solar Longwave Environmental irradiation Geometry) are compared with the SVF calculated from fisheye pictures (Hämmerle et al., 2011). The obtained results indicate that SVF values differ observably between the two groups of models and a weighting factor is used to mitigate this deviation for some of the models. A comparison of SVFs computed by the vector-based method with a 3-D vector building database and a shadow-casting method using the DEM data was performed, and the results show high similarity in terms of urban geometry (Gál et al., 2009). For now, many SVF algorithms have been developed based on DSM and DEM data; hence, validating and cross-checking these theoretical models is crucial when applying the SVF to complex urban and mountainous environments. However, few studies focus on the estimation and comparison of these methods for the same situation using both DSM and DEM data. Some SVF models that are always used in modeling topographic surface radiation are not intercompared using same data and assessment procedures, and their applicability to the DSM image in urban areas also needs thorough verification'(Zhong-Hu Jiao etal. 2019).

In order to figure out the SVF, ground measurements are always useful for the SVF analysis. Traditionally, the SVF is estimated by taking the 180° picture. The fisheye lens provides the horizon-sky widely view from street to upward sky. It is obviously effective to gather fisheye pictures with angular of 3-D surface structures to derive SVF. The method to calculate SVF based on

fisheye image improve the SkyHelios (Blennow, 1995; Bradley et al., 2001; Grimmond et al., 2001). Besides taking photographs, the SVF can also be obtained by different data collection methods, for example stereo images and LiDAR-derived DEM or DSM data. Synthetic fisheye images from Google Earth 3-D mesh data within urban areas are newly used to generate the corresponding SVFs.

The calculation model of the sky view factor includes oscine-weighted, the one based on the sloped coordinate system, the Dozier-Frew and the ring weighting method etc.

Helbig et al. (2009) provide an approach to calculate the SVF. This model assume the equal-diffuse radiation in a specific environment. The grid of pixel is related to plane from the horizon to upward. SVF from the human view can be calculated by the cosine-weighted formula which is showed in formula (2-1-1):

$$SVF = \frac{1}{\pi}\int_{sky}\cos\theta\, d\Omega = \frac{1}{\pi}\int_0^{2\pi} d\phi \int_0^{\pi/2}\sin\theta\cos\theta\, d\theta$$

$$= \frac{1}{2\pi}\int_0^{2\pi}\cos^2\vartheta_h(\phi)\, d\phi \approx \frac{1}{N}\sum_{i=1}^{N}\cos^2\vartheta_{h,i}(\phi_i) \qquad (2\text{-}1\text{-}1)$$

Where:

dΩ represents the solid angle;

θ and ϕ indicate the zenith and azimuth angles in the hemispherical space, respectively;

N is the number of the discretized azimuth angle ϕi;

ϑh(ϕ) is the angle between the inclined surface and the horizon direction in the azimuth angle ϕ in the sloped coordinate system.

The DEM and DSM data can also be regarded as the quadrate pillars side by side, not just a continuous surface connected with triangular patches. This alternative is more suitable for the high-resolution DSM data in urban areas where there are various buildings with great variations of the elevation. Thus, a simplification of this formula is introduced for the horizontal coordinate system, and the ϑh(ϕ) is redefined as the angle between the connection line (from the surrounding point to the target point) and the horizon line at a target point in the azimuth angle ϕ. This method, which is referred to hereafter as the Helbig_h method, is a simpler and easier approach in the calculation of the SVF than the below-mentioned algorithms. If the definition of the SVF is based on the sloped coordinate system, it is then defined as: (Manners et al., 2012)

$$SVF = \frac{1}{2\pi}\int_0^{2\pi} \sin^2\left(H_\phi + \frac{\pi}{2} - \operatorname{atan}\left(\frac{-1}{\tan S\cos(\phi - A)}\right)\right) d\phi$$

$$\approx \frac{1}{N}\sum_{1}^{N}\sin^2\left[H_\phi + \frac{\pi}{2} - \operatorname{atan}\left(\frac{-1}{\tan S_{\phi_i}\cos(\phi_i - A_{\phi_i})}\right)\right] \qquad (2\text{-}1\text{-}2)$$

Where:

H_ϕ is the angle between the connection line (from the surrounding point to the target point) and the normal line at a target point in the azimuth angle ϕ in the horizontal coordinate system;

S and A represent the slope and aspect angles of a pixel, respectively.

The third SVF model is also based on the horizontal plane coordinate. With the assumption that thermal radiation from the atmosphere is isotropic, this fast calculation method is given in an analytical form' (Dozier & Frew, 1990):

$$\begin{aligned} SVF &= \frac{1}{\pi}\int_{0}^{2\pi}\int_{0}^{H_\phi}\eta_d(\theta,\phi)\sin\theta\left[\cos\theta\cos S + \sin\theta\sin S\cos(\phi - A)\right]d\theta\, d\phi \\ &\approx \frac{1}{2\pi}\int_{0}^{2\pi}\left[\cos S\sin^2 H_\phi + \sin S\cos(\phi - A)\left(H_\phi - \sin H_\phi\cos H_\phi\right)\right]d\phi \end{aligned} \qquad (2\text{-}1\text{-}3)$$

Where:

$\eta_d(\theta, \phi)$ is the anisotropic factor that is used to describe the anisotropy of reflected or emitted radiation from surrounding topographic contributions, which is equal to 1 due to the isotropic radiation.

The maximum search radius (R) and number of horizontal search directions (N) are two important parameters in calculating the SVF, as they determine the SVF fluctuations. Choosing the optimal number of search directions depends on R and the horizontal resolution of the DSM or DEM data. The low values of R do not require many search directions in the calculation of the SVF. As R increases, more urban features are included in the SVF image, which generally causes the SVF to be smaller, but the computation time noticeably increases. With that result, the considering objects is how to optimize the R by considering the the limitation of calculation. This paper sets N to 32 and R to 500 pixels to calculate SVF; that is, R is 1.5 and 15 km for DSM and DEM data, respectively.

The last method is based on the shadow-casting algorithm (Lindberg & Grimmond, 2010; Ratti et al., 2004; Steyn, 1980). In this method, a virtual hemisphere with light sources is placed over each pixel. The shadow binary image from DSM or DEM images with different solar angles is first calculated by the shadow-casting algorithm. The shadow volume image is constructed through sequentially moving the DEM image at different solar elevation and azimuth angles for each iteration when the height of DEM data is reduced and a portion of the shadow volume is derived. The entire shadow volume is generated by taking the maximum of the volume for each iteration until the height of the moving DEM is lower than the original DEM or is outside the area of interest. To produce the actual shadow binary image, the DEM is

subtracted from the shadow volume image, then the pixel with a negative or zero value is set to 1 that is exposed to sunlight, and the value of other pixels is set to 0, indicating the shadow. Finally, the SVF is calculated by the ring weighting method on the basis of the number of obscured light sources. The weighting formula is as follows:

$$SVF = \sum_{i=1}^{n} \chi_i \frac{1}{\pi} \sin\left(\frac{\pi}{180}\right) \sin\left(\frac{\pi(2\alpha_i - 1)}{2n}\right) \frac{360}{\kappa_i} \quad (2\text{-}1\text{-}4)$$

Where:

χ_i is the i-th shadow binary image;

n is the total number of shadow images and is 653 proposed by Lindberg and Grimmond (2010);

α represents the altitude angle in degree;

κ_i is the number of azimuth angles for the i-th ring.

Called as the Lindberg-Grimmond (L-G) method, this method had been realized as a software called “Urban Multi-scale Environmental Predictor” (Zhong-Hu Jiao et al., 2019).

The methodology in this thesis is to calculate SVF by images obtained from Google Street View who will be converted into fisheye image in the next step. Therefore, SVF calculation formula based on fisheye images is more effective.

Helbig et al. (2009) provide an approach to calculate the SVF. This model assume the equal-diffuse radiation in a specific environment. The grid of pixel is related to plane from the horizon to upward. SVF from the human view can be calculated by the cosine-weighted formula which is showed in formula (2-1-5):

$$\omega = \sin\left(\frac{\pi\theta}{180}\right) \times \frac{90}{\theta} \times \cos\left(\frac{\pi\theta}{180}\right) \qquad (2\text{-}1\text{-}5)$$

Where:

θ is the zenith angle in degrees.

The parameter θ is included because each pixel in the fisheye image has a different influence on the final SVF value. In this equation, the incoming diffuse radiation is scaled by Lambert's cosine law using the zenith angle (Zhong-Hu Jiao et al., 2019).

2.2 Semantic segmentation models

Segmentation has a wide array of applications ranging from scene understanding, inferring support-relationships among objects to autonomous driving. Early methods that relied on low-level vision cues have fast been superseded by popular machine learning algorithms. In particular, deep learning has seen huge success lately in handwritten digit recognition, speech, categorizing whole images and detecting objects in images. Besides, In comparison to traditional machine learning techniques, deep learning can exploit both loosely defined global and local features captured at multiple levels to achieve better prediction accuracy.

Image segmentation involves dividing an image (or video frame) into multiple segments or objects. Segmentation plays a very important role in medical image analysis (for example, tumor boundary extraction and tissue volume measurement), autonomous carriers (for example, navigable surfaces and pedestrian detection), video surveillance, and augmented reality. Many image segmentation algorithms have been developed in the literature, from the earliest methods, such as thresholding, histogram-based methods, region division, k-means clustering, watershed, to more advanced algorithms, such as active contours, graph-based segmentation , Markov random field and sparse method. However, in the past few years, deep learning networks have produced a new generation of image segmentation models, their performance has been

significantly improved, and usually reached the highest accuracy rate on popular benchmarks, which has led many people to believe that the paradigm shift in the field.

Image segmentation can be expressed as a pixel classification problem with semantic labels (semantic segmentation) or a single object segmentation problem (instance segmentation). Semantic segmentation uses a set of object categories (such as people, cars, trees, and sky) to label all image pixels at the pixel level, so it is usually more difficult than predicting a single label for the entire image. Instance segmentation further expands the scope of semantic segmentation by detecting and depicting each object of interest (for example, individual segmentation) in the image. The thesis survey covers the latest literature on image segmentation and discusses more than 100 deep learning-based segmentation methods proposed by 2019. This thesis provides a comprehensive understanding and understanding of different aspects of these methods, including training data, network architecture selection, loss function, training strategy and their main contributions. Then compare and summarize the performance of these methods, and discuss the challenges and future development directions of image segmentation models based on deep learning. According to its main technical contributions, works based on deep learning are divided into the following categories:

1) Fully convolutional networks

2) Convolutional models with graphical models

3) Encoder-decoder based models

4) Multi-scale and pyramid network based models

5) R-CNN based models (for instance segmentation)

6) Dilated convolutional models and DeepLab family

7) Recurrent neural network based models

8) Attention-based models

9) Generative models and adversarial training

10) Convolutional models with active contour models

11) Other models

10 categories are worth mentioning that some of these works are common, such as encoder and decoder parts, skip connection, multi-scale analysis, and the recently used dilated convolution. Therefore, it is difficult to mention the unique contribution of each algorithm, but it is easier to categorize it according to its structural contribution.

2.2.1 Fully Convolutional Networks

This work is considered to be a milestone in image segmentation, proving that it is possible to train deep networks for semantic segmentation in an end-to-end manner on images of variable size. However, although the traditional FCN model is universal and effective, it also has certain limitations. It cannot perform real-time reasoning quickly, cannot effectively consider global context information, and is not easy to convert into 3D images. There are several efforts to overcome some of the limitations of FCN. For example, Liu et al. (2016) proposed a model called ParseNet to solve the problem of FCN ignoring global context information. ParseNet adds global context information to FCN by using the average feature of the layer to increase the feature of each location.

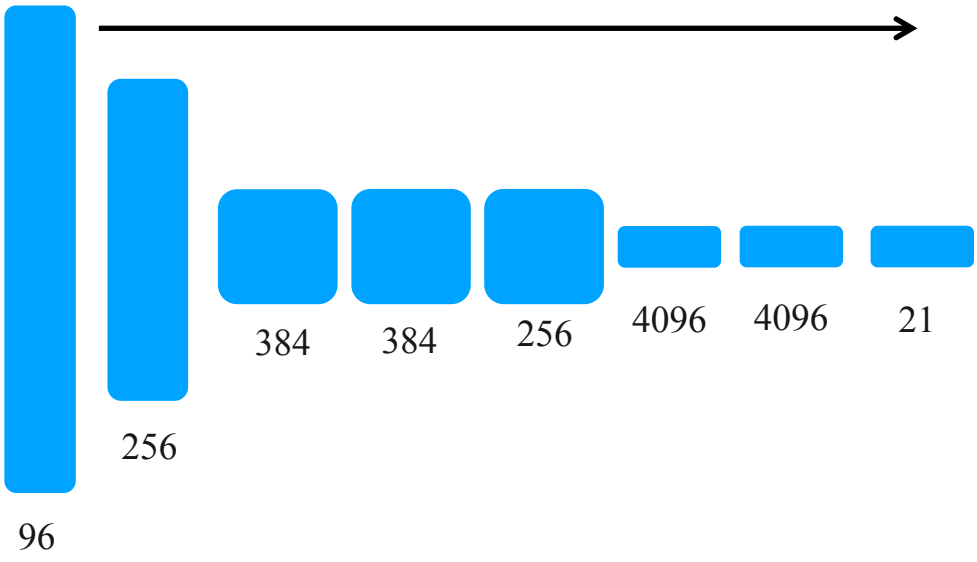


Figure 2-2-1 FCN Net frame (from 'Fully Convolutional Networks for Semantic Segmentation')

FCNs have penetrated into complex segmentation problems, such as brain tumor segmentation, instance rainbow feature segmentation, skin decomposition, and segmentation.

2.2.2 Convolutional Models With Graphical Models

Chen et al. proposed a semantic segmentation algorithm based on the combination of CNN and fully connected CRF. They show that for accurate target segmentation, the response from the final layer of deep CNN is not sufficiently localized (because the invariance of CNN makes it suitable for high-level tasks, such as classification). In order to overcome the problem of poor positioning performance of deep CNN, they combined the response of the final CNN layer with the fully connected CRF. The thesis showed that the model can locate with higher accuracy than previous methods.

Schwing and Urtasun proposed a fully connected deep structure network for image segmentation. They proposed a method to jointly train CNNs and fully connected CRF for semantic image segmentation, and achieved encouraging results on the PASCAL VOC 2012 dataset. Zheng et al. proposed a similar semantic segmentation method combining CRF and CNN. In another related work, Lin et al. proposed an efficient semantic segmentation algorithm based on context depth CRF. Liuatal proposed a semantic classification algorithm that integrates rich information into MRF, including high-order relations and

mixed-label text. Unlike previous work that used iterative algorithms to optimize MRF, they proposed a CNN model, a parsing network, which can achieve deterministic end-to-end calculations in a forwarding process.

2.2.3 Encoder-Decoder Based Models

Another popular family of depth models for image segmentation is based on the convolutional encoder-decoder architecture. Most segmentation work based on Deep Learning uses some kind of encoding-decoding model. The thesis divides these works into two categories, the encoder-decoder model for general segmentation and the encoder-decoder model for medical image segmentation.

Many studies have optimized the network architecture of FCN: one of them is to reduce the loss of detail information, so most of the optimization is for encoder; Another kind of research aims to reduce the mapping of encoder output results (up sampling) to the original input space to reduce the information loss caused by simple bilinear interpolation in FCN. Therefore, an end-to-end trained decoder is designed. The bilinear interpolation in FCN can be replaced by a trainable convolution layer, but in order to make up for the loss brought by down sampling, the results of sampling on a layer in the decoder and the results of encoder on the same layer will be added, which is difficult It also brings additional storage space consumption.

The goal of SegNet is to design a fast and small storage space deep network model suitable for real-time applications.

The innovation of SegNet is to use the tensor (pooling indices) recording the maximum response feature position of maxpool layer for up sampling, which avoids the consumption caused by learning up sampling in FCN, and then use the trainable convolution layer to make the sparse feature map dense, which will avoid the additional space consumption caused by saving the feature map.

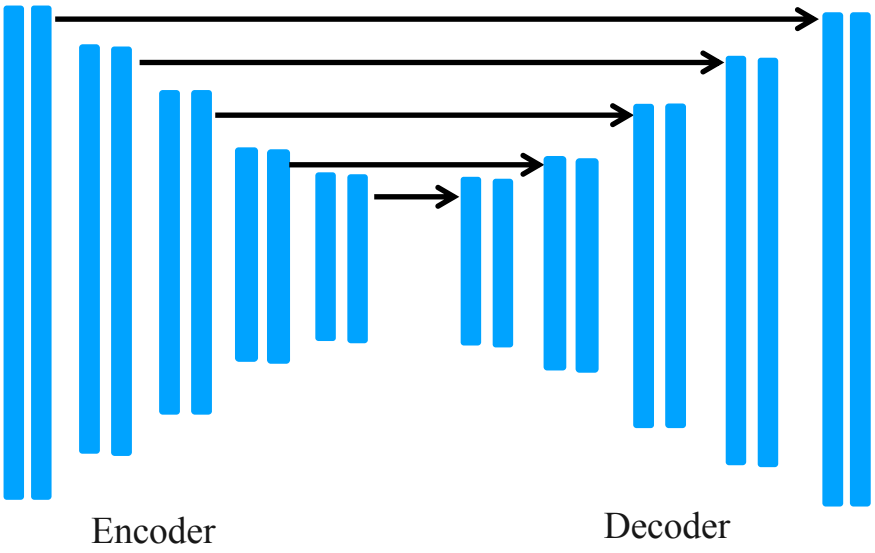


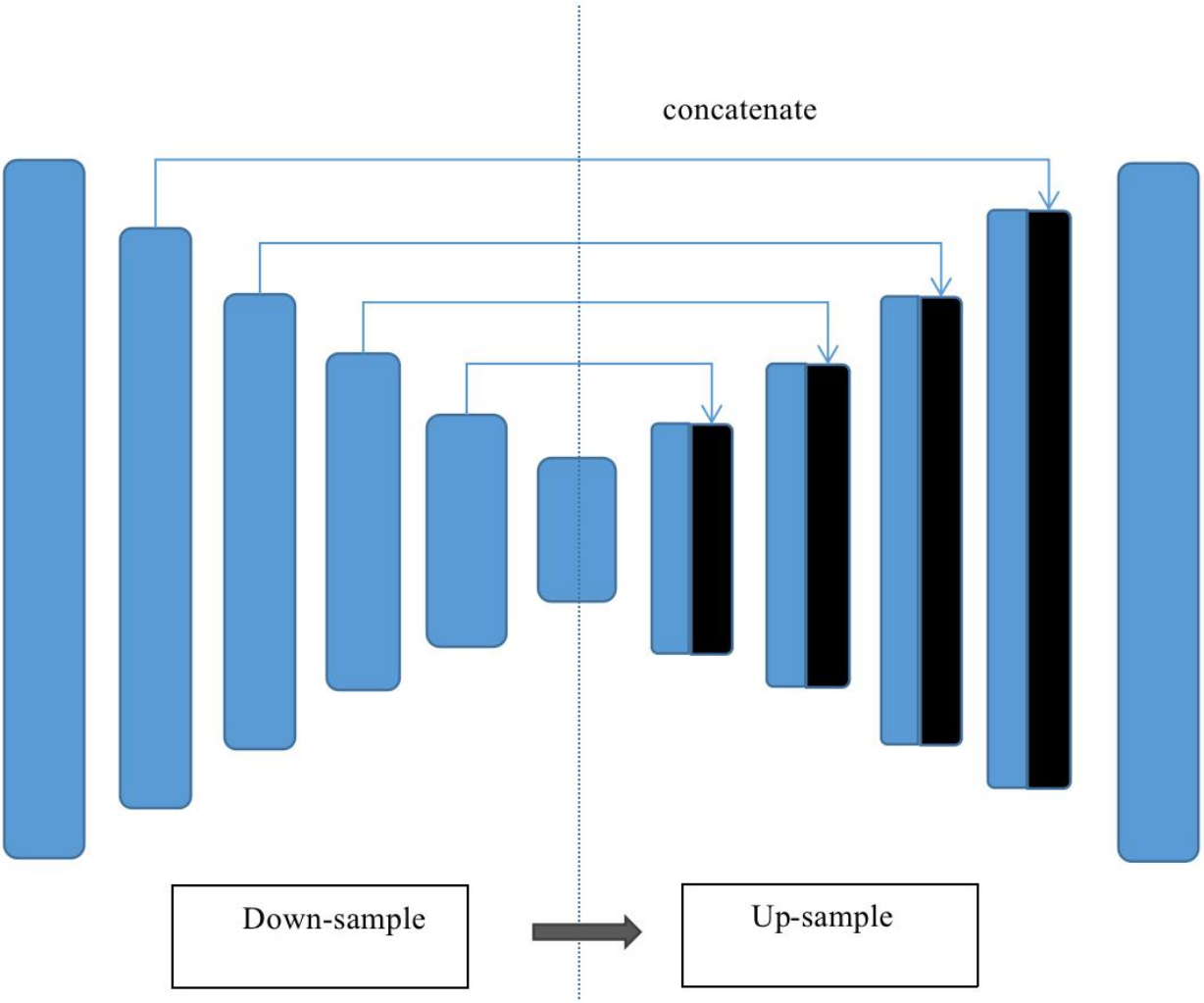
Figure 2-2-2 SegNet (from 'SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling')

Inspired by FCNs and codec models, medical/biomedical image segmentation has initially developed several models. U-Net and V-Net, two well-known architectures, who differ from the FCN's point by point addition, splice features together in the channel dimension to form thicker features.

Ronnebergeretal proposed U-Net for segmentation of biological microscope images. Their network and training strategies rely on the use of data augmentation to learn more effectively from available annotated images. V-Net is another well-known FCN-based model proposed by Milletari et al. for 3D medical image segmentation. For model training, they introduced a new objective function to enable the model to deal with severe imbalances between the number of voxels in the foreground and the background. The network performs end-to-end training on the MRI volume describing the prostate, and learns to predict the segmentation of the entire volume at the same time. Other related work in medical image segmentation includes progressive dense V-net (PDV-net) and so on. Used to quickly and automatically segment lung lobes from chest CT images, and a 3D-CNN encoder for lesion segmentation.

2.2.4 Multi-Scale and Pyramid Network Based Models

Multi-scale analysis (Multi-scale analysis) is an ancient idea in image processing and has been widely used in various neural network structures. One of the most prominent models is the Feature Pyramid Network (FPN) proposed by Lin et al. Although it is mainly used for target detection, it is also used for segmentation, using the inherent multi-scale pyramid hierarchy of deep CNNs to construct feature pyramids with marginal additional costs.



In order to fuse low-resolution and high-resolution features, FPN is composed of a bottom-up path, a top-down path, and horizontal connections. Then, the connected feature maps are processed by 3×3 convolution to produce the output of each stage. Finally, each stage of the top-down path generates a prediction to detect the object. For image segmentation, the author uses two multilayer perceptrons (MLPs) to generate masks.

Zhao et al. developed the Pyramid Scene Parsing Network (PSPN), which is a multi-scale network that can better learn the global contextual representation of the scene. The residual network (ResNet) is used as the feature extractor, and different patterns are extracted from the input image through the extended network. Then these feature maps are input into the pyramid pool module to distinguish patterns of different scales. They are assembled on four different scales, and each scale corresponds to a pyramid layer, and is processed by a 1×1 convolutional layer to reduce their dimensionality. The output of the pyramid layer is up-sampled and connected with the initial feature map to capture local and global context information. Finally, a convolutional layer is used to generate a pixel-by-pixel prediction.

Ghiasi and Fowlkes developed a Laplacian pyramid-based multi-resolution reconstruction architecture, which uses skip connections and multiplicative gating of high-resolution feature maps to continuously reconstruct the subdivision boundaries of low-resolution maps. Studies have shown that the spatial resolution of convolutional feature maps is low, but high-dimensional

feature representations contain a large amount of sub-pixel positioning information. There are other models that use multi-scale analysis for segmentation, such as DM-Net (Dynamic Multi-scale Filter Network), Context Contrast Network and Gated Multi-scale Aggregation (CCN), APC-Net, MSCI, and salient object segmentation.

2.2.5 R-CNN Based Models

He Kaiming proposed a Mask R-CNN for object instance segmentation, which surpassed all previous benchmarks on many COCO challenges. This model effectively detects objects in the image while generating high-quality segmentation masks for each instance.

Hu et al. proposed a new partially supervised training paradigm and a new weight transfer function. This paradigm makes the constrained state classification model a large set of categories. All categories have box annotations, but only a small number of categories have Mask comment. Chen et al. developed an instance segmentation model MaskLab, which is based on a faster R-CNN with semantic and directional features. Another interesting model is Tensormask, proposed by Chen et al., based on dense sliding window instance segmentation. They used dense instance segmentation as a prediction task on 4D tensors and proposed a general framework to make new operators on 4D tensors possible. They proved that the tensor view has a greater gain

than the baseline, and the result is comparable to the masked R-CNN. TensorMask has achieved promising results in dense object segmentation (Many other instance segmentation models are developed based on R-CNN, such as those developed for masking suggestions, including R-FCN, DeepMask, SharpMask, PolarMask, and boundary Perceptual instance segmentation. It is worth noting that another promising research direction is to try to solve the problem of instance segmentation by learning grouping clues for bottom-up segmentation, such as deep watershed transformation and semantic instance segmentation through deep volume learning.

To know more about the selective search algorithm, follow this link. These 2000 candidate region proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM to classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values to increase the precision of the bounding box. For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could've been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.

2.2.6 Dilated Convolutional Models and DeepLab Family

Dilation/dilation convolution introduces another parameter for the convolutional layer, namely the expansion rate. It can expand the receptive field without increasing the calculation cost. Dilated convolution has been widely used in real time periods. Some of the most important ones include the DeepLab family, multi-scale Context Aggregation, dense upsampling convolution and hybrid dilated convolution (DUC-HDC), Densespp and ENet.

DeepLabv1 and DeepLabv2 are one of the most popular image segmentation methods developed by Cheneta. The latter has three key features: First, it uses extended convolution to solve the problem of reduced resolution in the network (caused by max pooling and striding). The second is atrus Spatial Pyramid Pool (ASPP), which uses filters to detect incoming convolutional feature layers at multiple sampling rates to capture objects and image context at multiple scales to be reliable at multiple scales Divide the object locally. The third method is to combine deep CNNs and probabilistic graphical models to improve the location of target boundaries. The best DeepLab (using ResNet-101 as the backbone) achieved a mIoU score of 79.7% in the 2012 Pascal VOC Challenge and 70.4% in the Cityscape Challenge.

Subsequently, Chen et al. proposed DeepLabv3, which combines cascaded and parallel extended convolution modules. Parallel convolution modules are grouped in ASPP. 1×1 convolution and batch normalization are added to ASPP. In 2018, Chen et al. released Deeplabv3+, which uses an encoder-decoder

architecture, including Atrus separable convolution, spatial convolution and dot convolution for each input channel. They use the DeepLabv3 framework as the encoder. The best DeepLabv3+ pre-trained on the COCO and JFT datasets achieved an mIoU score of 89.0% in the 2012 Pascal VOC Challenge.

2.2.7 Recurrent Neural Network Based Models

Although CNN is a natural solution to computer vision problems, it is not the only possibility. RNNs are very useful in building short-term/long-term dependency models between pixels to (potentially) improve the estimation of segmentation maps. Using RNNs, pixels can be connected together and processed sequentially to model global information for semantic segmentation.

The main tasks include:

1. Scene labeling with lstm recurrent neural networks

2. Semantic object parsing with graph lstm

3. Da-rnn: Semantic mapping with data associated recurrent neural networks

4. Segmentation from natural language expressions

2.2.8 Attention-Based Models

Different from other works, in these works, the volume and integral genre is trained to learn the typical semantic features of the labeled objects. Huang et al. proposed a semantic segmentation method based on the reverse attention mechanism. Their Reverse Attention Network (RAN) architecture also trains the model to capture the opposite concept (ie, features that are not related to the target class).

RAN is a three-branch network that performs direct and reverse attention learning processes at the same time. Li et al. developed a pyramid attention network for semantic segmentation. This model makes full use of the influence of global context information on semantic segmentation. They combined the attention mechanism and the spatial pyramid to extract precise dense features for pixel labeling, instead of complex extended convolutions and well-designed decoding networks. Recently, Fu et al. proposed a dual attention network for scene segmentation, which can capture rich context dependencies based on a self-attention mechanism.

Many other studies have explored the attention mechanism of semantic segmentation. For example, OCNet proposed an object context pool inspired by self-attention mechanism, expectation maximization attention (EMANet), Criss cross attention network (CCNet), end-to-end with repeated attention End instance segmentation, point spatial attention network and discriminative feature network (DFN) for scene analysis.

2.2.9 Generative Models and Adversarial Training

Since GANs were proposed, they have been widely used in the field of computer vision and used for image segmentation. Luc et al. proposed a confrontational semantic segmentation training method. They trained a convolutional semantic segmentation network and an adversarial network that distinguishes the ground truth segmentation map from the ground truth segmentation map generated by the segmentation network. They showed that this differential training method improves the accuracy on the PASCAL VOC 2012 data set.

Su Li et al. proposed semi-weakly supervised semantic classification using Gans. It includes a proxy network, provides additional training examples for multi-class classifiers, acts as a discriminator in the GAN framework, assigns sample labels y from K possible classes or marks them as fake samples (extra classes). In another work, Hung et al. developed a semi-supervised semantic segmentation framework using adversarial networks. They designed an FCN discriminator to distinguish the predicted probability map from the ground truth segmentation distribution taking into account the spatial resolution. The loss function considered by this model includes three items: the cross-entropy loss of segmentation ground truth, the adversarial loss of the discriminating network, and the semi-supervised loss based on the confidence map, that is, the output of the discriminator.

Xue et al. proposed a multi-scale L1 loss adversarial medical image segmentation network. They use segmentation or generation rate segmentation label mapping, and propose a network with a multi-scale L1 loss function to force critics and segmenters to learn to capture global and local features of long-distance and short-distance spatial relationships between pixels .

2.2.10 CNN Models With Active Contour Models

The exploration of the synergy between FCNs and active contour models (ACMs) has recently attracted research interest. One method is to establish a new loss function based on the ACM principle. For example, Chen et al. proposed a supervised loss layer, which contains the area and size information of the prediction mask during the FCN training process, which solves the problem of cardiac MRI central compartment segmentation. Similarly, Gur et al. proposed an unsupervised loss function based on active contours without edge morphology for microvascular image segmentation. A different approach initially tried to use ACM only as a post-processing program for FCN output, and some efforts tried to achieve moderate co-learning by pre-training FCN. The work of Le et al. is an example of an ACM post-processor for semantic segmentation of natural images.

Hatamizadeh et al. proposed an integrated deep activity injury (DALS) model to train the dorsal root bone to predict the parameter function of the new

local parameterized level energy collection function. In other related work, Marcos et al. proposed Deep Structure Active Contours (DSAC), which combines ACMs and pre-trained FCNs in a structured prediction framework for instance segmentation in aerial images (though manual initialization). For the same application, Cheng et al. proposed a deep active ray network (DarNet) similar to DSAC, but adopted a different explicit ACM formula based on polar coordinates to prevent the contours from intersecting themselves. Hatamizadeh et al. recently introduced a truly end-to-end back propagation trainable, fully integrated FCN-ACM combination called Deep Convolution Active Contour (DCAC).

2.3 Objectives of the Thesis

The thesis aims to develop a platform for calculating the Sky View Factor. Through indicating a couple of coordinates, or a text file including a serial of coordinates, users are able to figure out the SVF. Meanwhile, users can analyze the sun path and solar irradiation through also define the coordinates.

The system can provide the panoramic image, re-fisheye image and fisheye image with sun path, so that people can consider the irradiation with the sun path and the SVF result.

3. Methodology and tool development

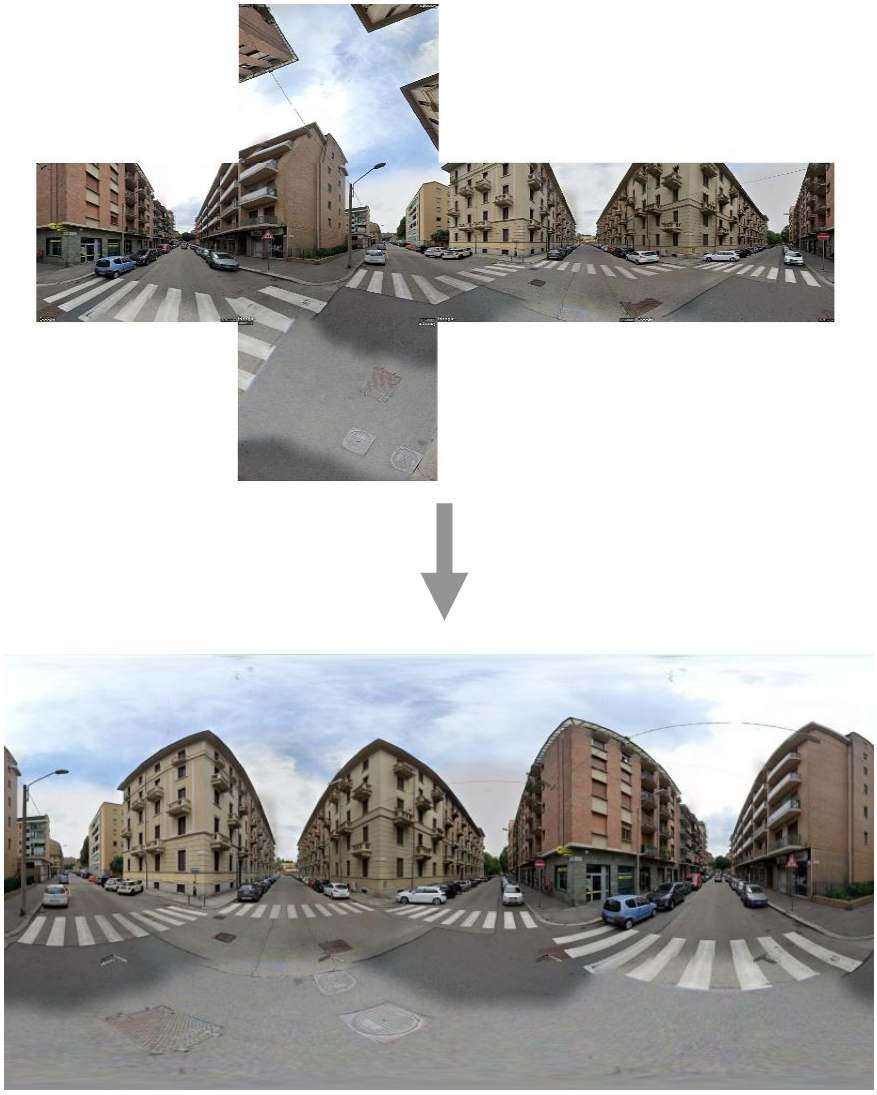
The workflow of this thesis will be divided into the following four steps as showed in figure 3-1. In the stage-1, it is feasible to use latitude and longitude to obtain panoramic slices of Google Street View. In addition, the parameter set previously will request images with different characteristic, such as setting fov to request images with different width of view. The configuration of request is listed as following Table 3-1 for description. And then in stage-2, the work is stitching six pictures into panoramic pictures. In stage-3, the work is to use semantic segmentation to perform image information and Classification, by

this way, find the class of the sky and non-sky, and finally convert it into a fisheye image for the calculation of the sky factor.

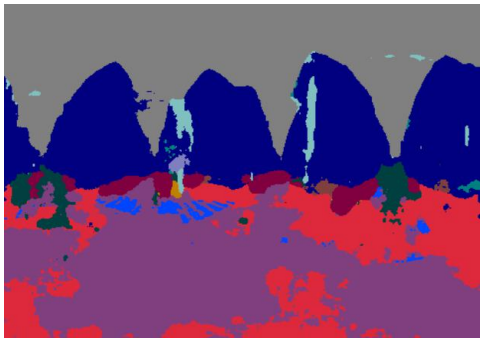
An image of methodology pipeline : stage -1 -> stage -2 -> stage -3

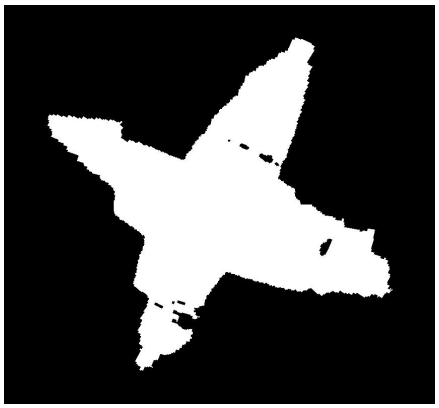
Stage -1. Request images from google by API

Stage -2. Stitch cube images into panoramic image



Stage -3 Play segmentation on panoramic image and refish it





3.1 Image downloading and processing

With the support of Google's big data, Google Street View provides a more convenient way of data acquisition for image collection. Google Maps provides users with free channels and methods for obtaining panoramic pictures. Get 6 cube photography of 360x480 panorama pictures through Google Street View API. This is effective for most locations in the city. Although the image returned by Google is not a fisheye image, it can be converted into the required fisheye image through the conversion of spherical coordinates and plane coordinates.

According to the Google street view API, it can be possible to easily request a panoramic picture by Google Street View through the specific coordinates and parameters concerning picture. Before sending the request, the parameters should be set as a list like table:

Google API developer keys	Every client can apply for the keys of Google Street View API
latitude && longitude	To get the coordinates of a specific location in Google Maps
Photo parameters:	

size	Max to 600 x 600, we will use 360 x 480 as default
heading	Adjust the heading of the camera, 0 and 360 are north, 90 is east, 180 is south, and 270 is west
Fov	For the horizontal field of view in Google Street View, we use 90
pitch	Adjust the camera's up and down angle, usually 90

Table 3-1-1 Google API

Defined all parameters for API request, it is possible to receive the needed picture. For this purpose, the following code is adopted:

```
Apiargs = {
        'location' : '45.044969, 7.64882',
        'size' : '640 × 640 ; 640 × 640',
        'heading' : '0 ; 90 ; 180 ; 270',
        'fov' : '90';
        'pitch' : '-90 ; 90',
        'key' : your keys applied
        }
```

The API of Google Street View has been verified effective through randomly sample in variant locations in a city, variant cities in a country, variant countries in world. The figure 3-2 shows some random position.

Then request the images through API. By this way, it is available to download images of one position and serial of positions as figure 3-3 shows.

The image request in Europe is completely effective, even in Iceland. While in Africa not at all. In Asia, apart from China, the other countries support Google Street View technology. But in China, the Baidu can support the Street View image by Baidu API.

Figure 3-1-2 panoramic images obtained

After we get the cube images concerning with the GSV, we need to stitch them based on the transform function between the global coordinates and Polar coordinates.

The goal is to determine the best estimate of the color at each pixel in the final spherical image given the 6 cubic texture images. The conversion process involves two main stages. The first stage is to calculate the polar coordinates corresponding to each pixel in the spherical image. The second stage is to use the polar coordinates to form a vector and find which face and which pixel on that face the vector (ray) strikes. In reality this process is repeated a number of times at slightly different positions in each pixel in the spherical image and an average is used in order to avoid aliasing effects.

If the coordinates of the spherical image are (i,j) and the image has width "w" and height "h" then the normalized coordinates (x,y) each ranging from -1 to 1 are given by:

$$X = 2j/w - 1$$

$$Y = 2j/h - 1 \qquad (3.1)$$

or $Y = 1 - 2j / h$ depending on the position of pixel 0

The polar coordinates theta and phi are derived from the normalised coordinates (x,y) below. theta ranges from 0 to 2 pi and phi ranges from -pi/2 (south pole) to pi/2 (north pole). Note there are two vertical relationships in common use, linear and spherical. In the former phi is linearly related to y, in the later there is a sine relationship.

$$\Theta = X * \pi$$

$$\Psi = Y * \pi / 2 \qquad (3.2)$$

or $\Psi = \text{asin}(y)$ for spherical vertical distortion

The polar coordinates (theta,phi) are turned into a unit vector (view ray from the camera) as below. This assumes a right hand coordinate system, x to the right, y upwards, and z out of the page. The front view of the cubic map is looking from the origin along the positive z axis.

$$x = \cos(\Psi)\cos(\Theta)$$

$$y = \sin(\Psi) \qquad (3.3)$$

$$z = \cos(\Psi))\sin(\Theta)$$

The intersection of this ray is now found with the faces of the cube. Once the intersection point is found the coordinate on the square face specifies the corresponding pixel and therefore colour associated with the ray.

The mapping of geometry showed as figure 3-4:

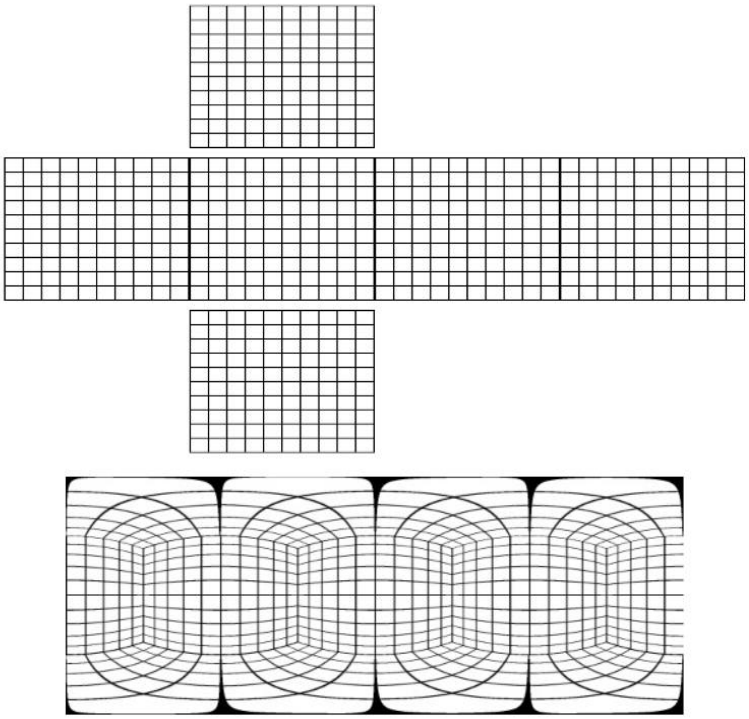


Figure 3-1-3 Cubes to Equi-rectangle

3.2 Deep learning for image segmentation

Deep learning is a branch of machine learning. It is an algorithm that attempts to use multiple processing layers that contain complex structures or consist of multiple nonlinear transformations to abstract data at a high level.

Before building segnet image segmentation model, we have two tasks to complete: understand the net model and how to realize in caffe. Being familiar with the network parameter setting in Caffe neural network, so the work should be carried out step by step.

S.1 the caffe profile configuration

There are two main profiles named “solver.prototxt” and “train_val.prototxt” that works for setting the net. “Solver.prototxt” mainly stores some super parameters used in model training, for example: net, defines the structure file of the model to be trained, i.e. train_ val.prototxt; base_Ir, defines basic learning rate etc.

Train_val.prototxt file is used to store the model structure, which mainly define the layers. We need to define the input and output of each layer of the net, as well as the motivate function, batch_ BN and other parameters. Part of the structure is shown in the Figure 3-2-1.

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to

capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

According to the Encoder-Decoder frame, we will build the net in train_val.prototxt file. And configure the train and test parameters

Encoder : Using pooling and convolution, the size of characteristic graph is reduced and the number of channels is increased – Subsample. VGG network removes the three-layer full connection layer, the model parameters are greatly reduced, and the model scale is greatly reduced. Batch normalization is added after each convolution.

Decoder : Each Decoder corresponds to an encoder, and a new upsample method is proposed, unpooling with indices.

The parameters that will be used when establist the net in caffe is:

net: is the net model that has been defined in train.prototxt;

net_param: collection of the net and layers name, and the source data path;

train_state: the state of the model progress, for example TRAIN state;

test_iter: the forward iteration of test of the net, also related with the size of batch_normalize;

test_interval: the internal step of the train and test;

base_lr: the initial learning rate;

max_iter: the maximum of the iteration;

The parameters that concerning to the optimization:

type: choose an optimal solution such as Adam;

momentum: the optimal system for weight upgrading;

weight_decay: the decay paramters to avoid overfitting;

regularization: L1, L2 used to avoid overfitting;

solver_mode: choose the CPU or GPU method(here use CPU in virtual machine)

We train the network to get the weight parameters through the training dataset. Fortunately, the dataset has been prepared online, the image set of target classification in the city. We can directly use the model paramters to train the model.

```
name: "VGG_ILSVRC_16_layer"
layer {
  name: "data"
  type: "DenseImageData"
  top: "data"
  top: "label"
  dense_image_data_param {
    source: "/SegNet/CamVid/tra
    batch_size: 4
    shuffle: true
  }
}
layer {
  bottom: "data"
  top: "conv1_1"
  name: "conv1_1"
  type: "Convolution"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    weight_filler {
      type: "msra"
    }
```

Figure 3-2-1 Net

Besides, the downloaded caffemodel should be put into the caffe root path "models/bvlc_reference_caffenet/". Meanwhile, we will encounter some problems in the process, the first is about the source path in configure file: text.txt and train.txt.

The files' name and architecture should be set as:

/SegNet-Tutorial/

CamVid/

test/

testannot/

train/

trainannot/

test.txt

train.txt

Since the virtualbox does not have GPU, we need to change the GPU to CPU in prototxt file. Train the model by terminal command with the command: /caffesegnet/build/tools/caffe train -solver/SegNetTutorial/Models/segnet_solver.prototxt.

Additionally, after having the generated the network structure, by your choice we can check whether the network structure is correct. We can intuitively express the diagram structure through the draw function in Caffe, as shown in the figure 3-2-2:

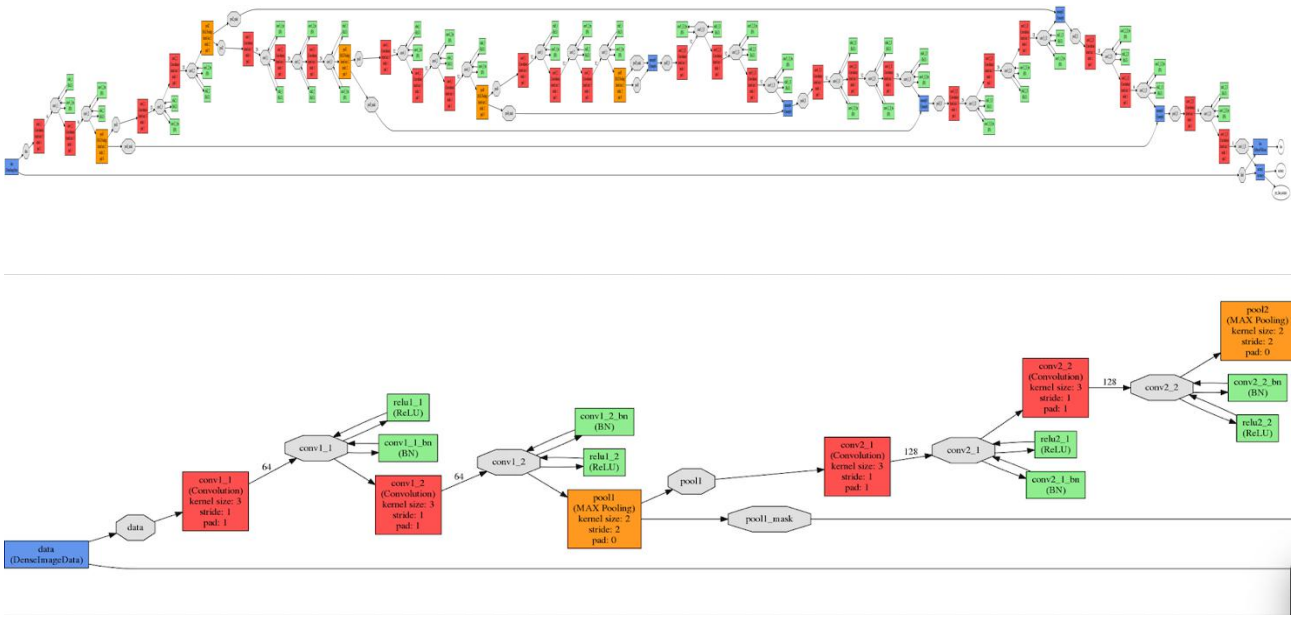


Figure 3-2-2 net frame output in caffe

The figure 3-2-2 shows the architecture of the SegNet, on top the line is the whole net frame, the button one is the part of the net in which is used to show more clearly.

Since the CPU works worse than GPU does, this stage will require long calculation time. After looping by 1000 times, the accuracy improves and the best configuration of the layers for segnet has been recorded.

When prediction is called through using the existing model configuration, just call the net stored in Caffe net and data stored in blob.

The first task is to train the segmentation model, some parameters are set before it start, such as the epoch of the train (better more than 30). Then the model can output the loss and iou score.

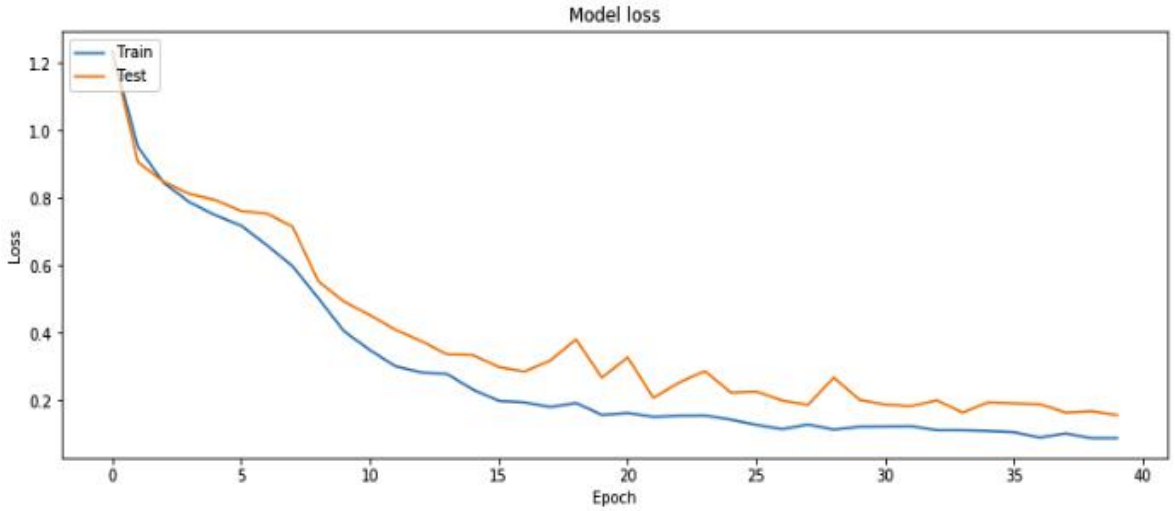


Figure 3-2-3 loss value during training

As the figure 3-2-3 shows, the loss goes well and close to zero at the end, which means the training for the model is good.

Intersection over union (IoU) is an important value that evaluate how the segmentation model is. From IoU, we can know whether the segmentation works good or bad, since the IoU is defined by the comparison between the real segmented image and the predicted segmented image. For each epoch, the model will be tested by the test data set to evaluate result of the train.

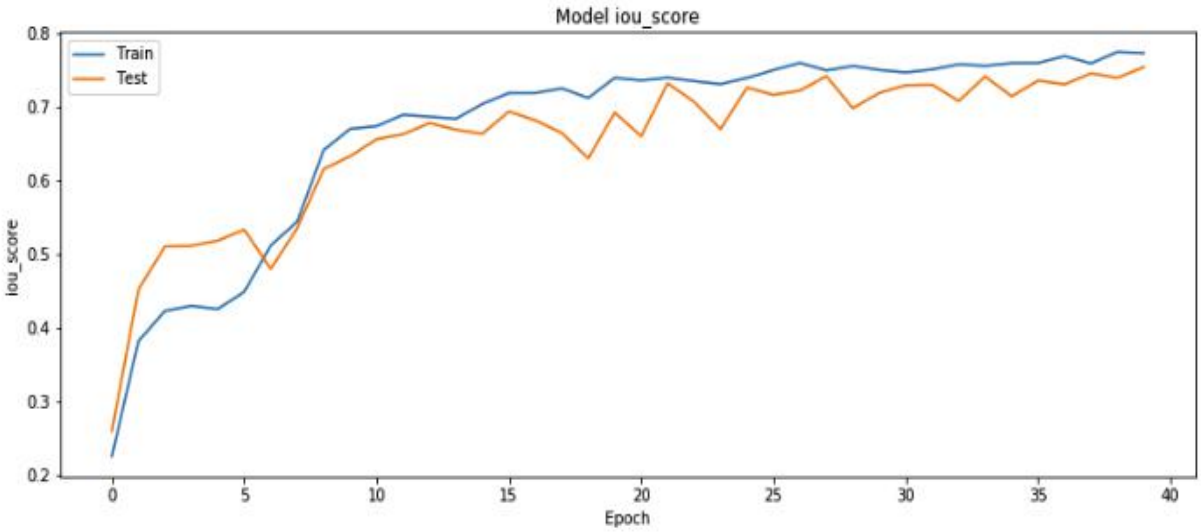


Figure 3-2-4 IoU score

According to the figure 3-2-4, the IoU of train and test can be seen similar. That means at the beginning, the model is trained not good, and after some epochs, they goes closer. Similar distribution means there is no over-fitting. And higher the IoU is, more precious the prediction is.

After the training work, next is prediction for the panoramic image so as to get the segmented image for the next step.

3.3 Fisheye image rebuilding and classification

The sky extraction is a requisite step to derive urban form information from hemi-spherical images. Therefore, before the analysis, we need to do some preparatory work. First, we need to convert the panoramic image to the fisheye image. Here we use the conversion of Cartesian coordinate and polar coordinates to convert the panoramic picture into a fish-eye picture of the ground perspective.

As the figure 3-3-1 showed, Xc and Yc are the coordinates of panoramic image which can be transferred into the (r, θ) in figure 3-8 that is the coordinates in fish-eye image.

Since the image in operation starts with (0, 0) on the left top corner, (Cx, Cy) is calculated by the middle of height and width, the center can be fixed.

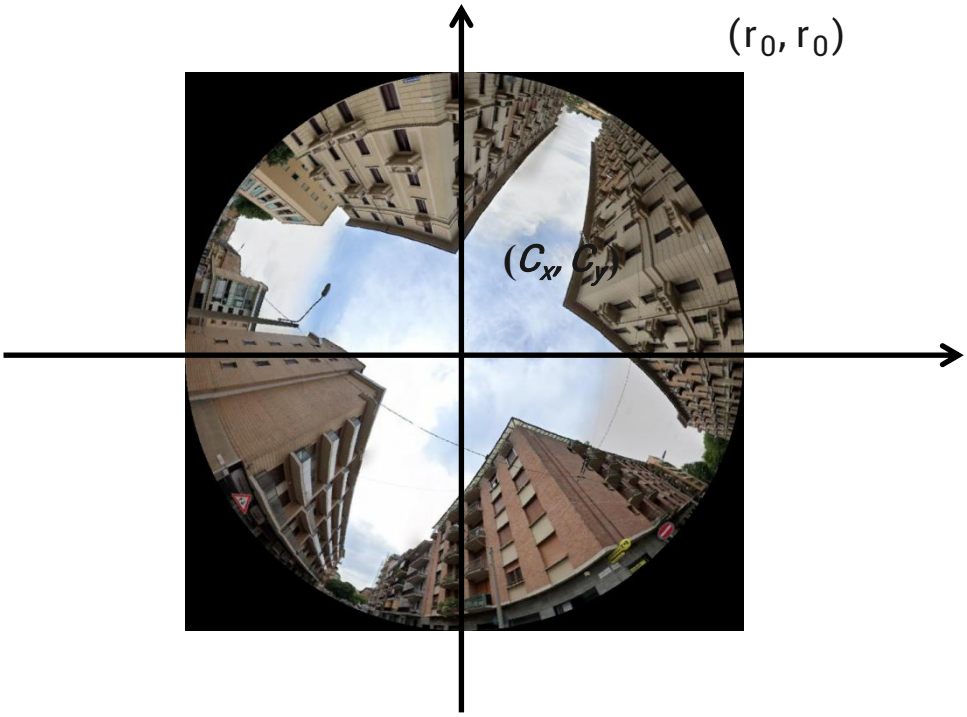


Figure 3-3-2 Fisheye Image

The formula for the transform is:

$$x_c = \frac{\theta}{2\pi} W_c \qquad (3.3.1)$$

$$y_c = \frac{r}{r_0} H_c \qquad (3.3.2)$$

$$r = \sqrt{(x_f - C_x)^2 + (y_f - C_y)^2} \qquad (3.3.3)$$

$$\theta = \begin{cases} \frac{3\pi}{2} - \arctan\left(\frac{y_f - C_y}{x_f - C_x}\right), & x_f < C_x \\ \frac{\pi}{2} - \arctan\left(\frac{y_f - C_y}{x_f - C_x}\right), & x_f > C_x \end{cases} \qquad (3.3.4)$$

(where: $C_x = C_y = \frac{W_c}{2\pi}$, $r_0 = \frac{W_c}{2\pi}$, $0 < x_f, y_f < \text{width}$ (from Using Google Street View for Street-Level Urban Form Analysis))

Next, the fish-eye picture is shown in the figure 3-3-3-a, which contains 12 classes. It is obviously to find the objects in image are classed and masked with different colors. The sky is gray covering most part of the image. What we need is only the class of the sky part, it is reasonable that we classify and process the picture again: to define the pixel of the sky part as 1 (white), and set the pixels of not sky part as 0 (black), then after loops for every pixel, the image showed as figure 3-3-3-b.

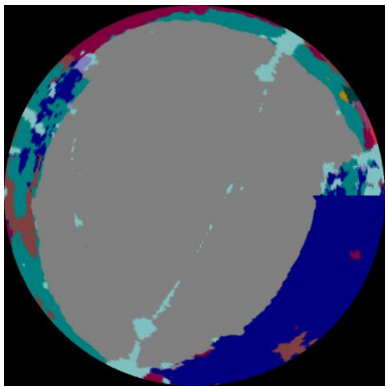


Figure 3-3-3-a fisheye image

Figure 3-3-3-b fisheye image with 0-1 pixel

3.4 Estimate the sky view factor

The sky view factor (SVF) is an important parameter of urban form. The SVF was proposed by urban climatologists to describe the amount of solar radiation reaching the street canyons. The SVF is defined as (Steyn 1980),

$$SVF = 1/\pi r_0^2 \int_{S^p} dS^p \qquad (3.4.1)$$

where r0 is the radius of the hemispheric radiating environment, Sp is the area of the circle sky area projected on the ground. When the sky is totally obstructed the SVF is zero, and the SVF is one when there is no obstruction for a site. In summer, building blocks and tree canopies act as major obstructions of sky in street canyons, while building block is the main obstruction in winter since trees are leafless. There- fore, in this study, we used the GSV-based photographic method and the building height model-based simulation method to estimate and SVF in summer and winter, respectively.

The photographic method is one of the standard methods for SVF estimation. In this study, we used the hemispherical images generated from GSV panoramas taken in summer to estimate the SVF in street canyons of Cambridge during summer. The photographic method (Steyn 1980; Johnson and Watson 1984) first divides the fisheye image into n concentric annular

rings of equal width, and then sums up all annular sections representing the visible sky. The SVF is then calculated as:

$$SVF = \frac{1}{2\pi} \sin\left(\frac{\pi}{2n}\right) \sum_{i=1}^{n} \sin\left(\frac{\pi(2i-1)}{2n}\right) \alpha_i \qquad (3.4.2)$$

Where:

n is the total number of rings;

i is the ring index;

α_i is the angular width in ith ring.

Based on previous studies (Chen et al. 2012), in this study, we set the n to 37. Since GSV panoramas used in this study were captured during leaf-on seasons, therefore, building blocks and street trees both act as obstructions of solar radiation in street canyons. The estimated SVF using GSV-based photographic method represents the openness of street canyons in summer.

3.5 Estimating Direct Sun-path in Street Canyons

The relative position of the Sun is a major factor in the heat gain of buildings and in the performance of solar energy systems. Accurate location-specific knowledge of sun path and climatic conditions is essential for economic decisions about solar collector area, orientation, landscaping, summer shading, and the cost-effective use of solar trackers. By locating where the sun is visible in the canyon, the light intensity can be obtained. This helps to analyze the comfort of the streets in the city, and can also obtain suggestions for the orientation and location of solar panels.

Sun path diagrams can show about how the sun will impact your site and building throughout the year. Stereographic sun path diagrams can be used to read the solar azimuth and altitude for a given location.

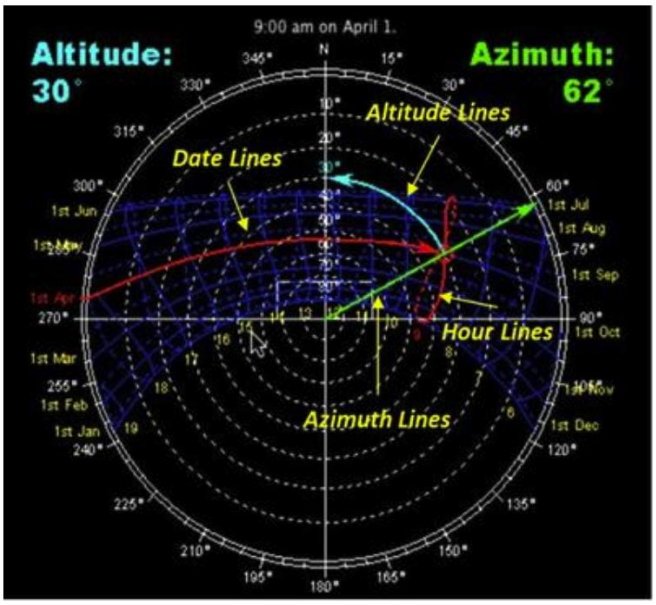


Figure 3-5-1

As the figure 3-5-1 shows:

- Azimuth Lines - Azimuth angles run around the edge of the diagram.
- Altitude Lines - Altitude angles are represented as concentric circular dotted lines that run from the center of the diagram out.
- Date Lines - Date lines start on the eastern side of the graph and run to the western side and represent the path of the sun on one particular day of the year.
- Hour Lines/ Analemma - Hour lines are shown as figure-eight-type lines that intersect the date lines and represent the position of the sun at a specific hour of the day. The intersection points between date and hour lines give the position of the sun.

The work is into two parts:

1. Fix the location and azimuth for the map so as to correct the sun path. According to the google street view API discription, the image named gsv_2, gsv_4, gsv_6, gsv_7, gsv_9 locates on the south-west, north-west, north-east, and south_east seperatedly.

2. Request the sunpath data through API of pvlib, organize the data and plot with the fisheye image.

In order to prominent the sun go through the sky open view, it is useful to plot the sun-path on classed image.

According to the sun-path data, the sun path at summer is arrange as figure 3--5-2. It can be used to predict the sun light coverage time when the position is fixed. The orange cure represents the angle of sun and horizon. If the apparent_elevation is over zero, that means the sun is rising above the horizon. And the blue cure represents the zenith. This graph shows that at Turin, the sun rises at around 5:30 am, and down at around 8:00 pm.

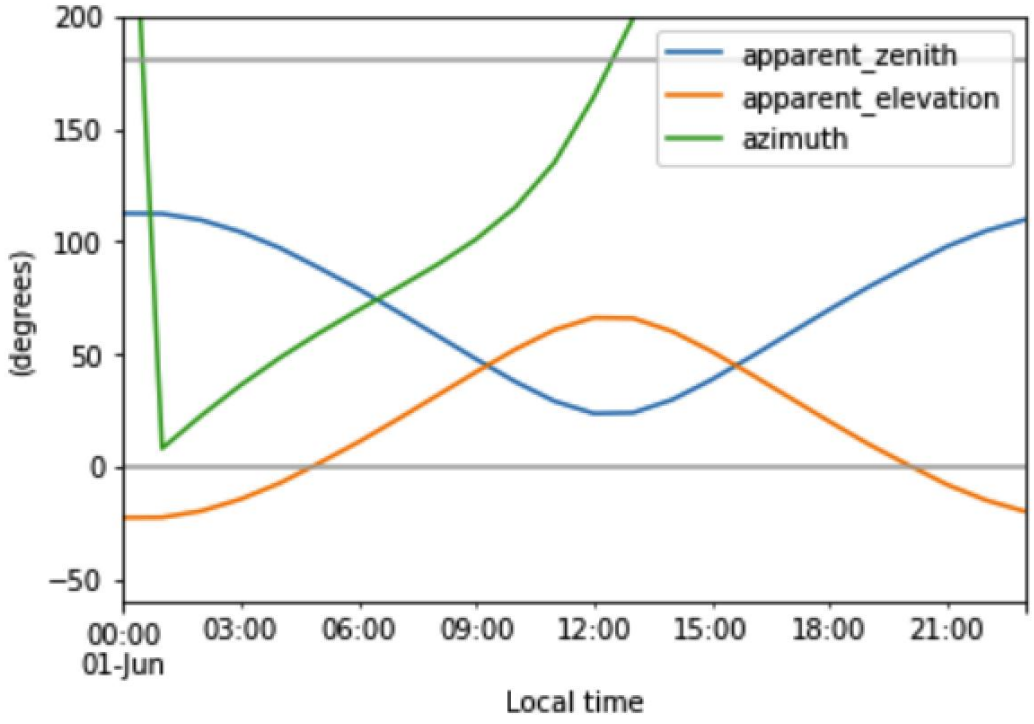


Figure 3-5-2 sun path

Then it is comfortable to locate data on the fish-eye image so that we are able to analyze the sun path clear. This is showed by figure 3-5-3.

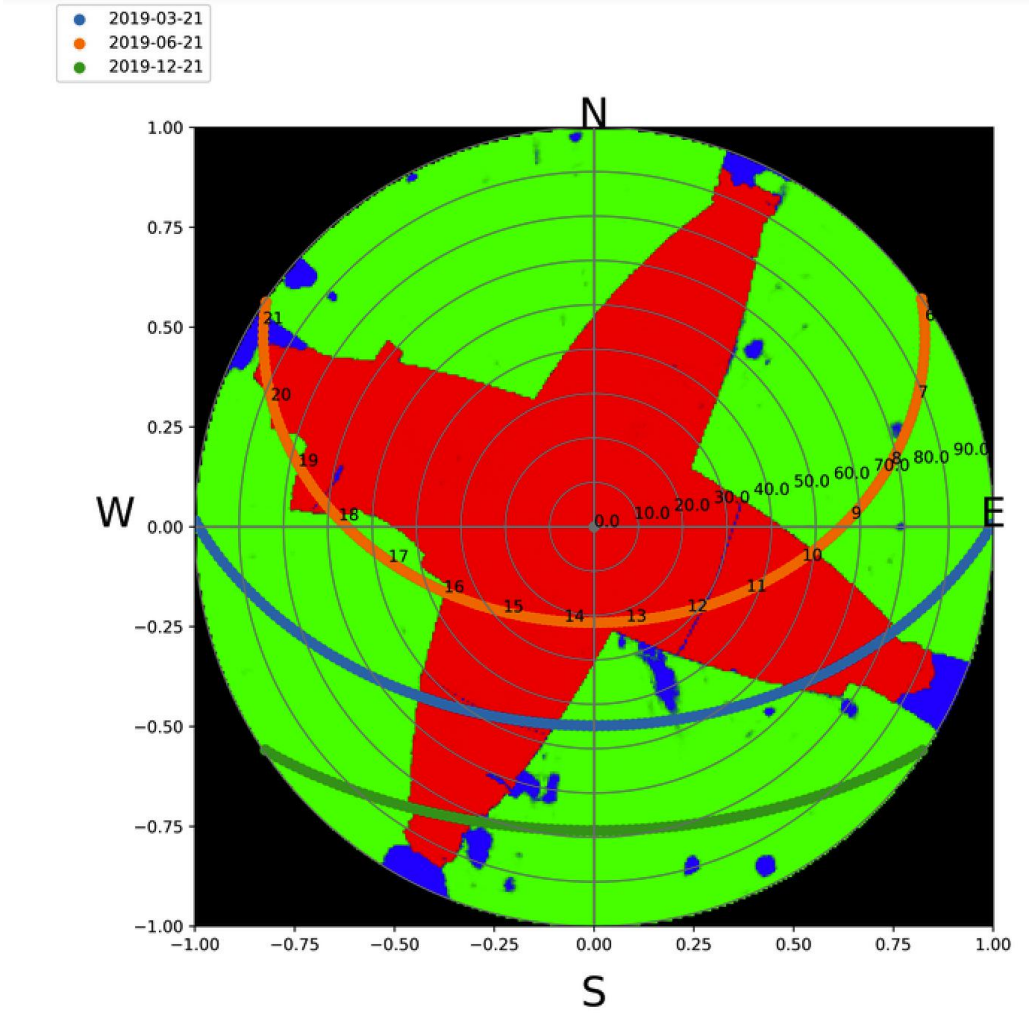


Figure 3-5-3 fisheye image with sun

From the figure 3-5-3, the daily sun-path can be found out, if the point is covering the building, we consider that the building obstructs the sun light. The colors indicates the range of movement of day lines from the summer to wind. Moreover, making the image binary with sun-path data is more simple to

analyzing. By this way, the image can be transformed into the image with 0-1 pixel (0: pixel with black color; 1: pixel with with color).

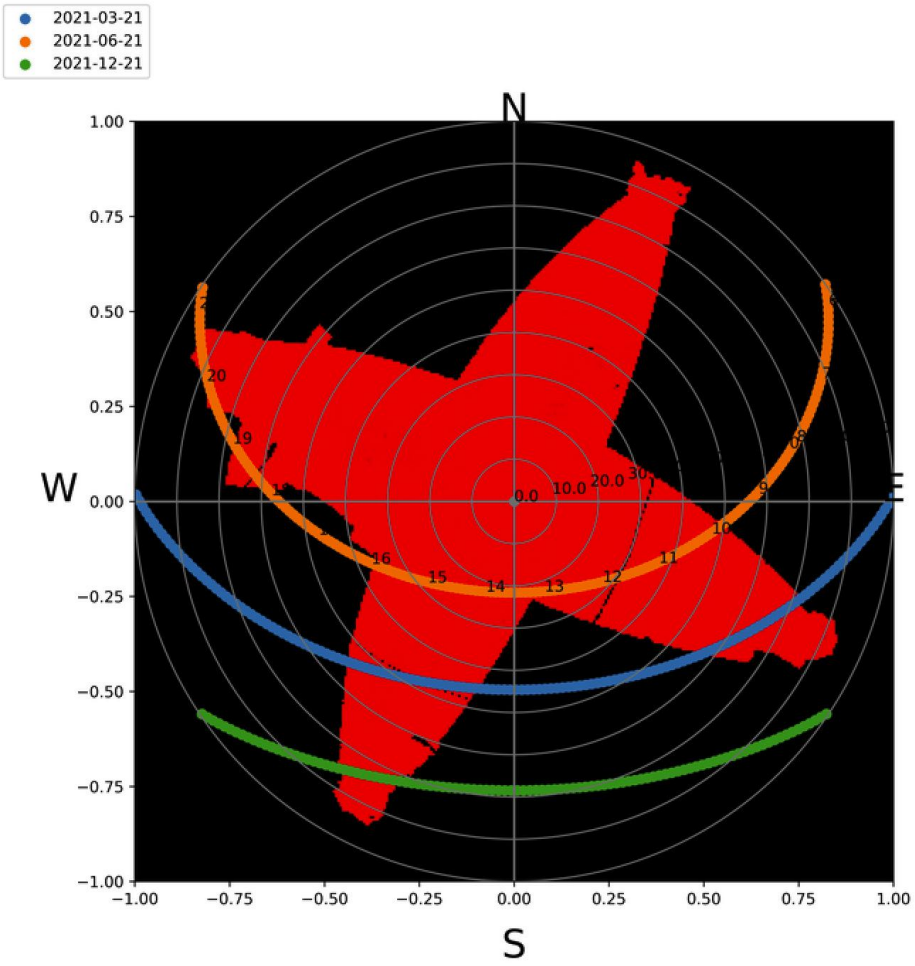


Figure 3-5-4 fish-eye image with sun path

As the figure 3-5-4 shows. The image only show the sky with red color and the rest with black, then it is easily to find the hours where sun position lies on the sky part.

From the graphs, we can directly find the point that Sun is shaded with time stamp since the number indicating the hourly locates the time.

Then it is useful to add the shaded/sunny data into the table in order to As the figure 3-5-6 shows, we transfer the sunny/shaded information into table, then locate the points in order to calculate the target data.

	apparent_zenith	zenith	apparent_elevation	elevation	azimuth	equation_of_time	ghi	dhi	sunny/shaded	shaded_value
2016-06-20 04:00:00	87.942758	88.240154	2.057242	1.759846	57.840135	-1.578974	0.000000	0.000000	shaded	0
2016-06-20 05:00:00	78.734621	78.815567	11.265379	11.184433	67.934787	-1.587977	0.745099	0.725975	shaded	0
2016-06-20 06:00:00	68.661661	68.704247	21.338339	21.295753	77.664048	-1.596980	78.325370	49.368292	shaded	0
2016-06-20 07:00:00	58.181769	58.208796	31.818231	31.791204	87.628996	-1.605982	234.913417	101.548427	shaded	0
2016-06-20 08:00:00	47.624540	47.642974	42.375460	42.357026	98.716243	-1.614983	406.096341	135.814156	sunny	1
2016-06-20 09:00:00	37.431760	37.444645	52.568240	52.555355	112.468219	-1.623984	565.920810	158.819591	sunny	1
2016-06-20 10:00:00	28.443131	28.452249	61.556869	61.547751	131.833266	-1.632984	698.832081	174.372295	sunny	1
2016-06-20 11:00:00	22.511163	22.518135	67.488837	67.481865	161.119130	-1.641984	793.851272	184.184649	sunny	1
2016-06-20 12:00:00	22.394303	22.401234	67.605697	67.598766	197.652038	-1.650983	843.631011	188.992464	sunny	1
2016-06-20 13:00:00	28.167154	28.176167	61.832846	61.823833	227.329140	-1.659982	844.427057	189.067719	sunny	1
2016-06-20 14:00:00	37.087893	37.100619	52.912107	52.899381	246.970140	-1.668980	796.180372	184.414262	shaded	0
2016-06-20 15:00:00	47.256152	47.274350	42.743848	42.725650	260.858533	-1.677977	702.518721	174.770652	shaded	0
2016-06-20 16:00:00	57.809131	57.835774	32.190869	32.164226	272.007852	-1.686973	570.675605	159.421764	shaded	0
2016-06-20 17:00:00	68.297306	68.339126	21.702694	21.660874	281.994905	-1.695969	411.508097	136.700065	shaded	0
2016-06-20 18:00:00	78.390048	78.468706	11.609952	11.531294	291.721258	-1.704964	240.382157	102.890811	sunny	1
2016-06-20 19:00:00	87.646598	87.922659	2.353402	2.077341	301.794664	-1.713958	82.672045	51.360782	shaded	0

Figure 3-5-6 data organized

After got the sunny/shaded part, we need to prepare the other part : the Global Horizontal Irradiation (GHI), so that we can evaluate the energy from sunlight in the area.

There are two ways to get the GHI:

1. Download the *EPW file from the Engergy-plus website;

2. Use the API of python-pvlib.

Both of them are acceptable, then comparison can be implemented between these two channels.

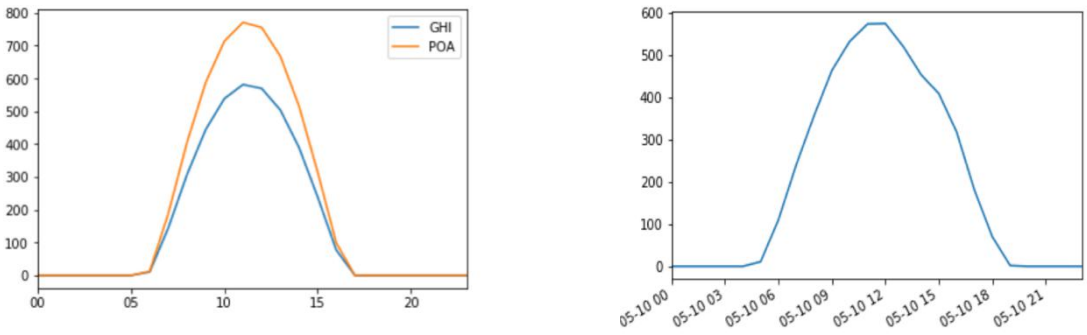


Figure 3-5-7 Plane of Array(POA) and GHI

We can see the figure 3-5-7, At the left one, there are two lines stands by the GHI and the POA separately, at the right one, there is only GHI lines. What we concentrate on is the GHI, so it is interesting to compare the GHI between the two graphs. The distribution of GHI in same day is highly similar, especially in the peak time.

Then the work moves on the different of GHIs between Summer and Winter. As the figure 3-5-8 shows, the winter and summer have different peak over the same day. In Summer, the value at peak is more than twice as much as that one in Winter. Also, the durations are obviously different, in Summer the duration is from around 6:00 am to 6:00 pm, that totally close to 12 hours, while in Winter, the it is from 7:30 am to 16:30 pm, that is shorted to 9 hours.

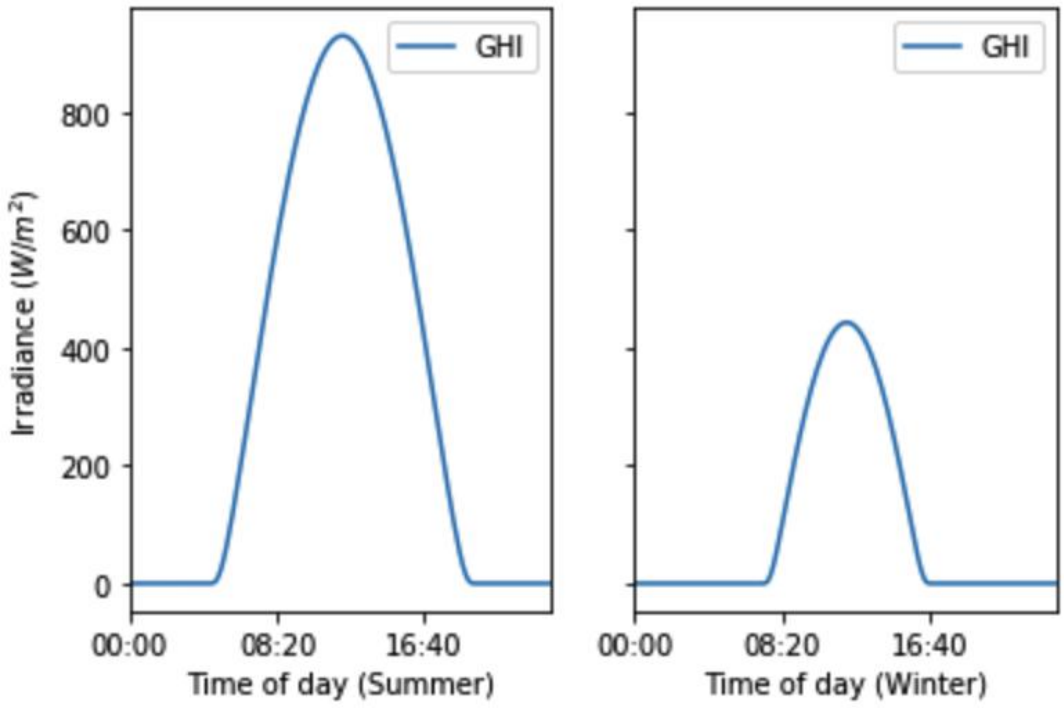


Figure 3-5-8 GHI distribution hourly

Not only the Through using the epw* file to analyze, now it is useful to overlook the distribution of values over the year, month, and days.

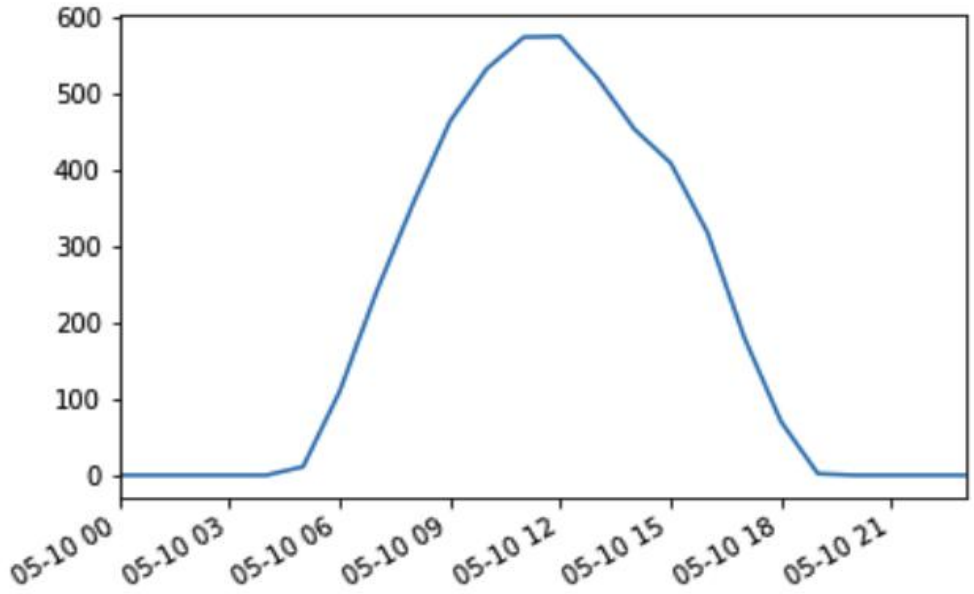


Figure 3-5-9 GHI distribution daily by epw*

From figure 3-5-9, it is easier to find some information from the special points. For example, the start time and end time for GHI is clear. Meanwhile, the value of peak can be read preciously, that is around 580W/m*2.

Not only the daily data can be figured to analyze, but the data for distribution on weekly and monthly can be also plotted. The data can be derived by mean of week and month.

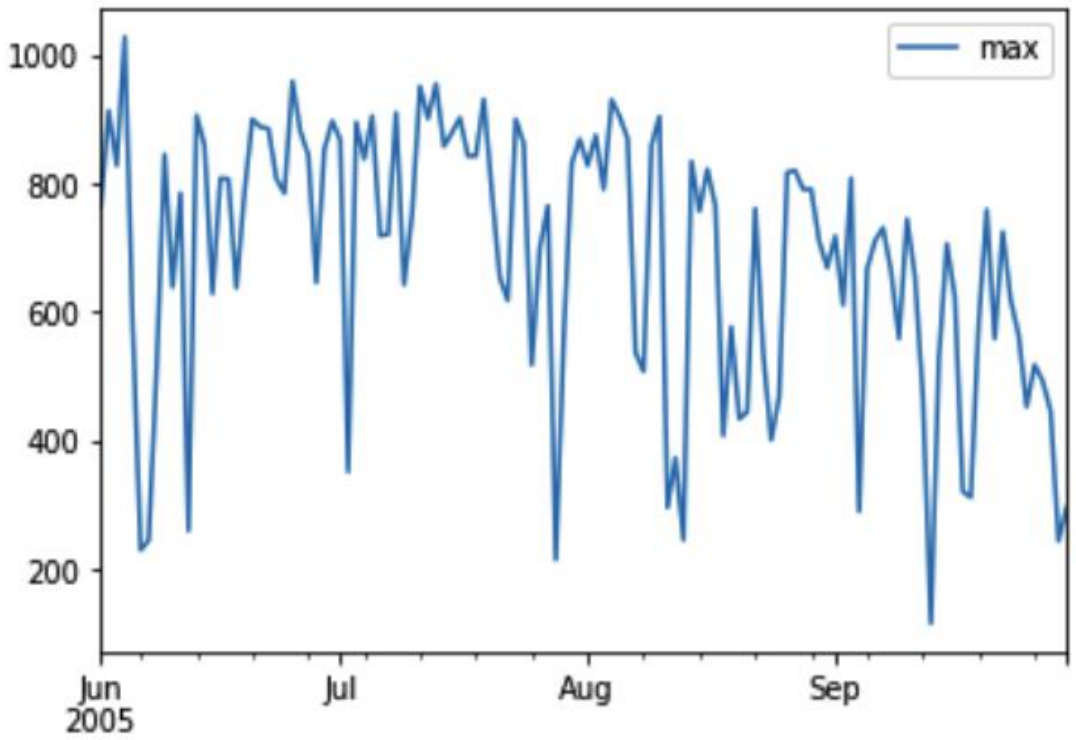


Figure 3-5-10 GHI distribution monthly (mean)

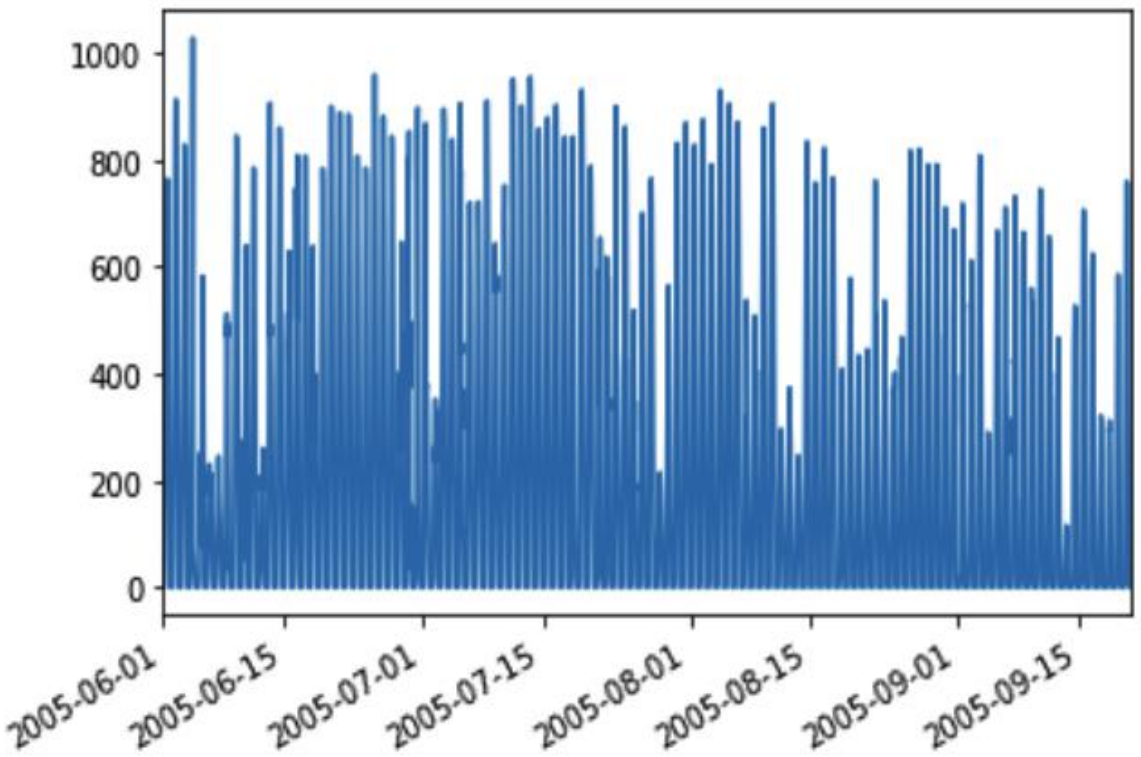


Figure 3-5-11 GHI distribution monthly a year (mean)

The figure 3-5-10 shows that samples into month, while 3-5-11 shows the distribution over a year. For monthly, the GHI is sampled by the max value (the peak) of everyday, that means the figure 3-5-10 represents how the the peak value goes. Similarly, figure 3-5-11 shows how that value changes over the year for everyday.

4. Sample application

In order to verify the feasibility of the method and application, sampling some coordinates of some urban streets will be randomly used as the input of the application, by this way, to test whether it can obtain the corresponding images, and whether it can process the image to panoramic one and do the segmentation. Meanwhile, check whether it can show the sun-path on the fish-eye image.

4.1 Sampling coordinates on map

Firstly, in order to test the feasibility of the Google Street View API, open the Google map and set the street view open, then in figure 4-1-1, the blue part represents the feasibility of downloading panoramic image.

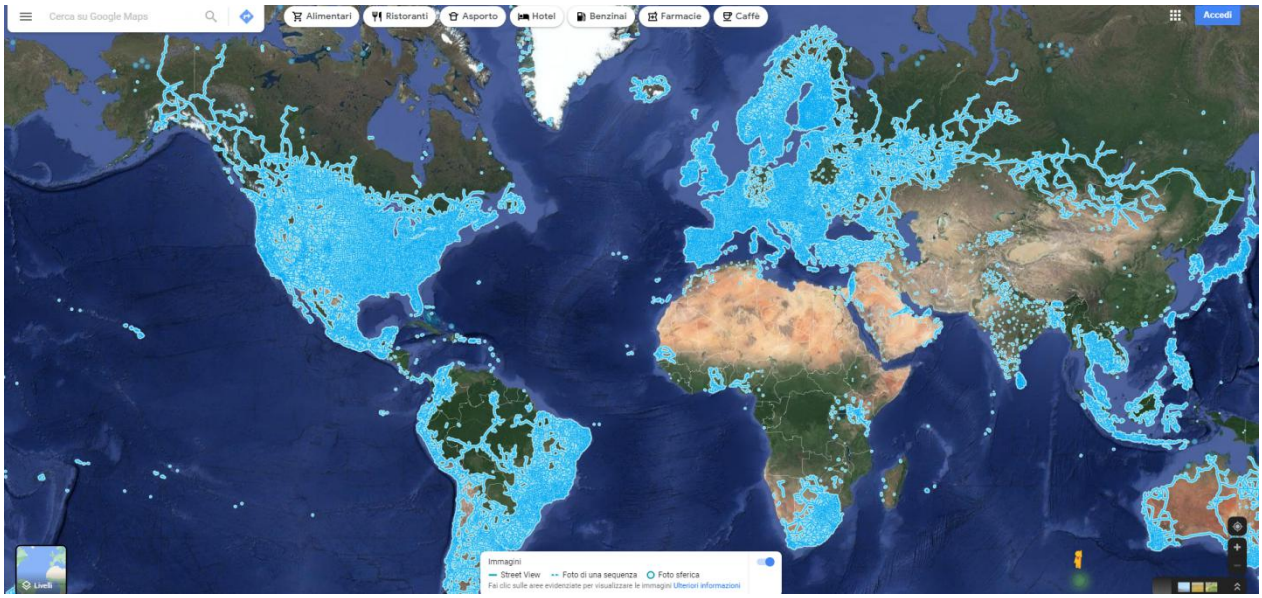


Figure 4-1-1 Google Street View acceptable area

Secondly, to verify the positions locating on the blue parts, randomly sample some points in cities of some countries. The random positions are sampled following the rule showed in figure 4-1-2:

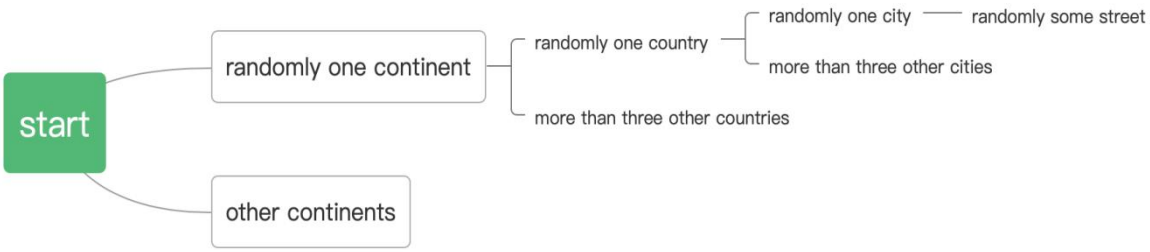


Figure 4-1-2 the structure of the sampling

As we can see, the positions are sampled randomly by countries, cities and streets. For one city, sampled 4 positions at least, and for one country, sampled 4 cities with 3 positions at least in it. For example, in this sample, we chose the five continents and chose Europe specially, then sampled some countries like Germany, Italy, France, Spain, Iceland and Greece. For each country, sampled more than tree cities. Meanwhile, chose a special countries randomly for sampling the cities and streets respectively with at least 6 positions.

The images will be requested by the chosen coordinates, and stored in special path with name of the coordinates. Go to the file gsv, the image can be found as figure 4-1-3:

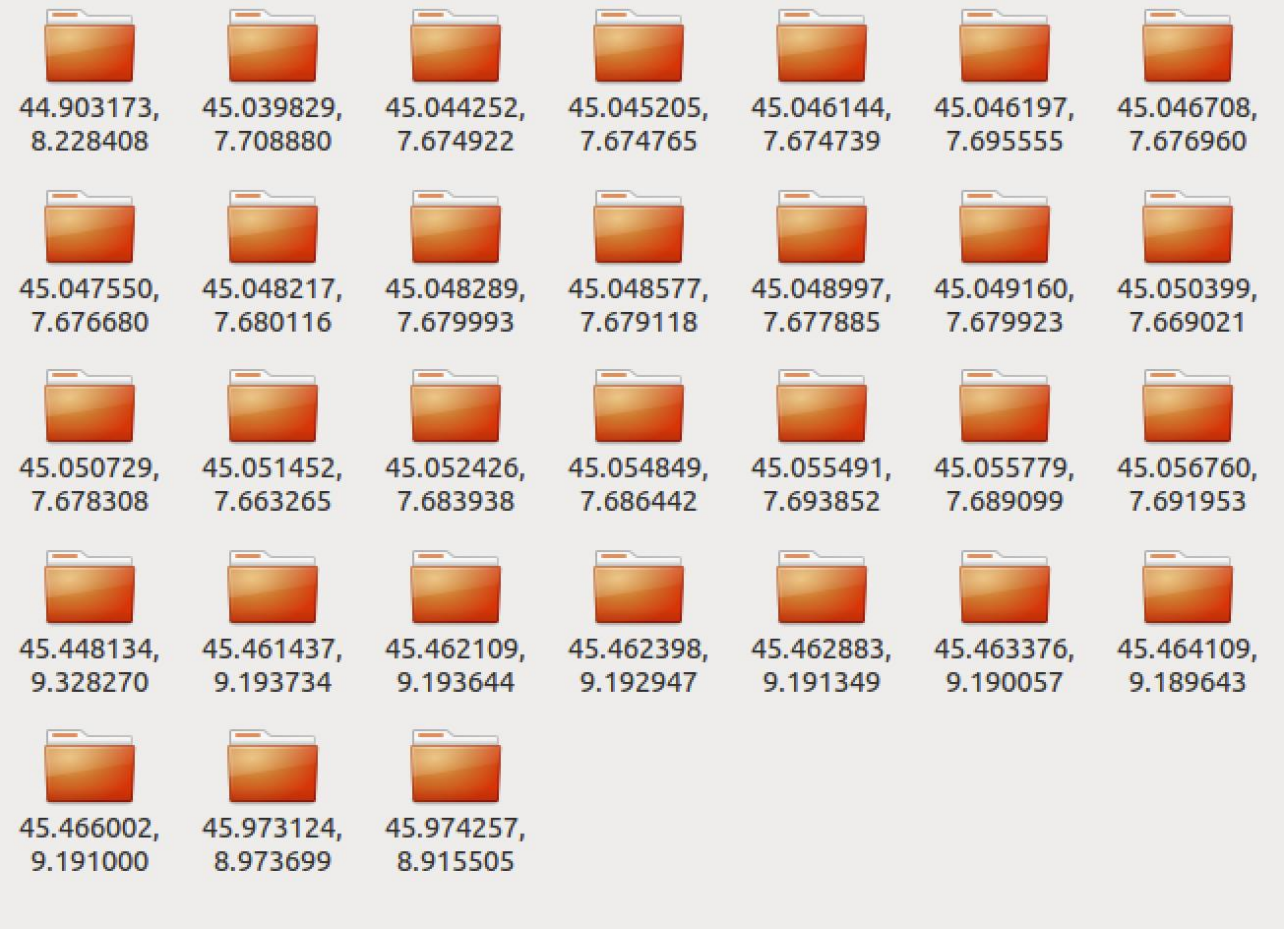


Figure 4-1-3 sampling image stored

In order to have an overlook on the sampled points on map, mark all of the points by python. The figure 4-1-4 shows the result. The blue points represent downloading Google Street View image is feasible, while the gray points represent not feasible.

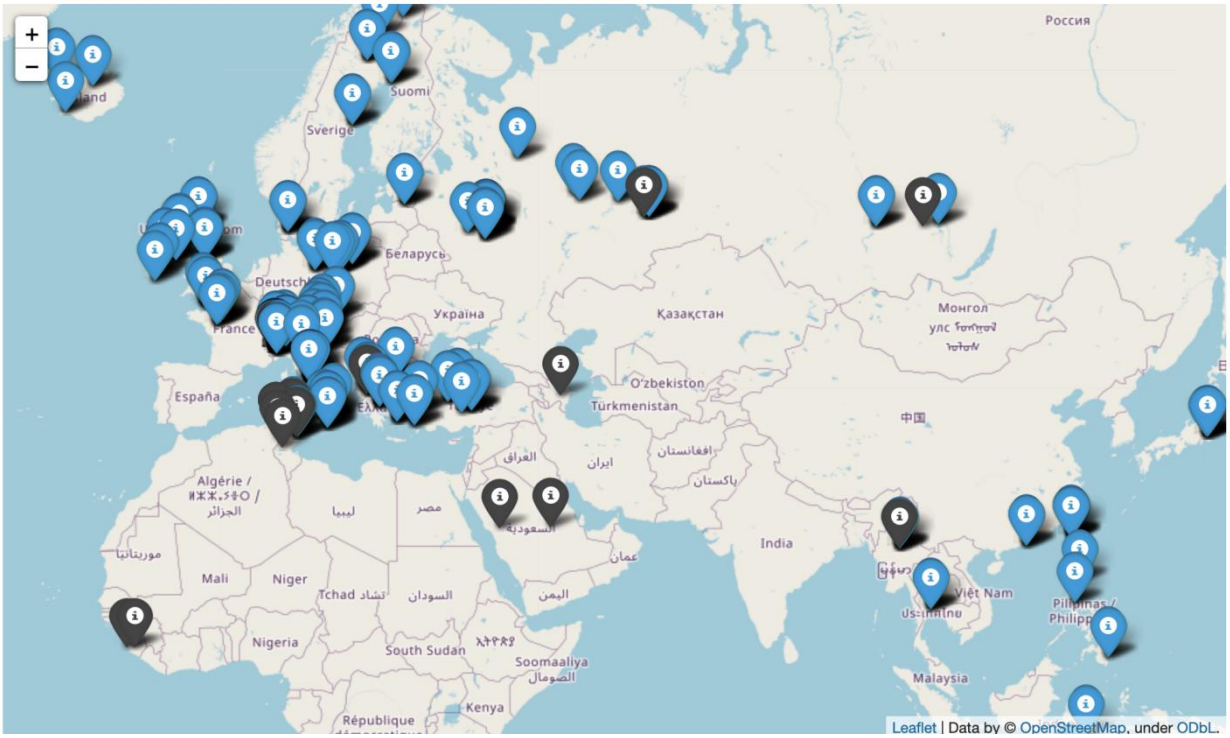


Figure 4-1-4 sampled points on map

By the way above, the method for requesting the images is feasible, but when we test in different countries, there are some permission. For example, in Africa and Asia, most request are forbidden in countries of Asia, except in Philippines, Singapore and Japan. While in Africa, there is no one feasible. In Europe and North America, the Google image in almost countries and cities are feasible.

4.2 test the application

In order to test whether the application can obtain the panoramic images and process them into segmented and re-fished images, we use some sampled coordinates as input.

The application is work as: as long as the user input the specific coordinates, and the Google API key, the application will request images and process them as the panoramic images, users are not required to do some additional work. The work is showed as figure 4-2-1

Figure 4-2-1 run to get segmentation

After that, the panoramic images are still stored. Then the segmented image will be transformed into fish-eye image. This step is also automatically by one line of code. As the figure 4-2-2 shows, the image can be converted into fisheye image. And figure 4-2-3 shows that after filter the special color representing the sky pixel, the image can be transformed into 1-0 image, the sky is with color red, while the others are with black.

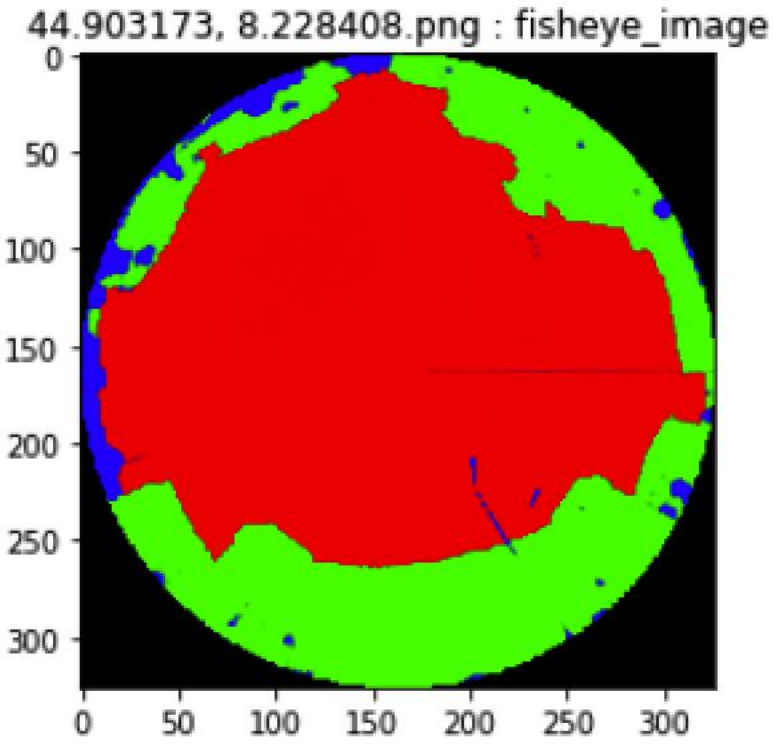


Figure 4-2-2 re-fisheye image

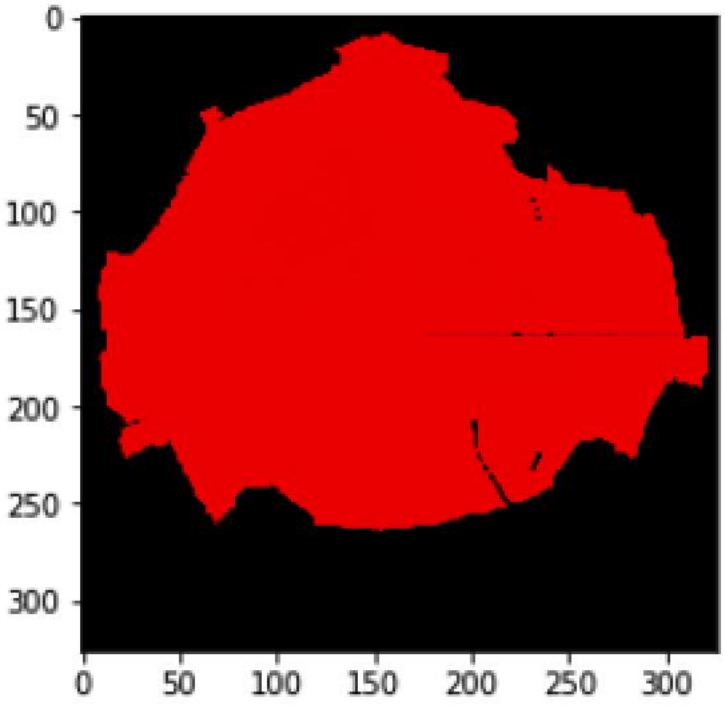


Figure 4-2-3 1-0 fish eye image

4.3 sun-path and analysis

The sun path can also be test by input the fisheye image path, and set the date you are interested in, then the sun path will be ploted on the fisheye images. The test is as figure 4-3-1.

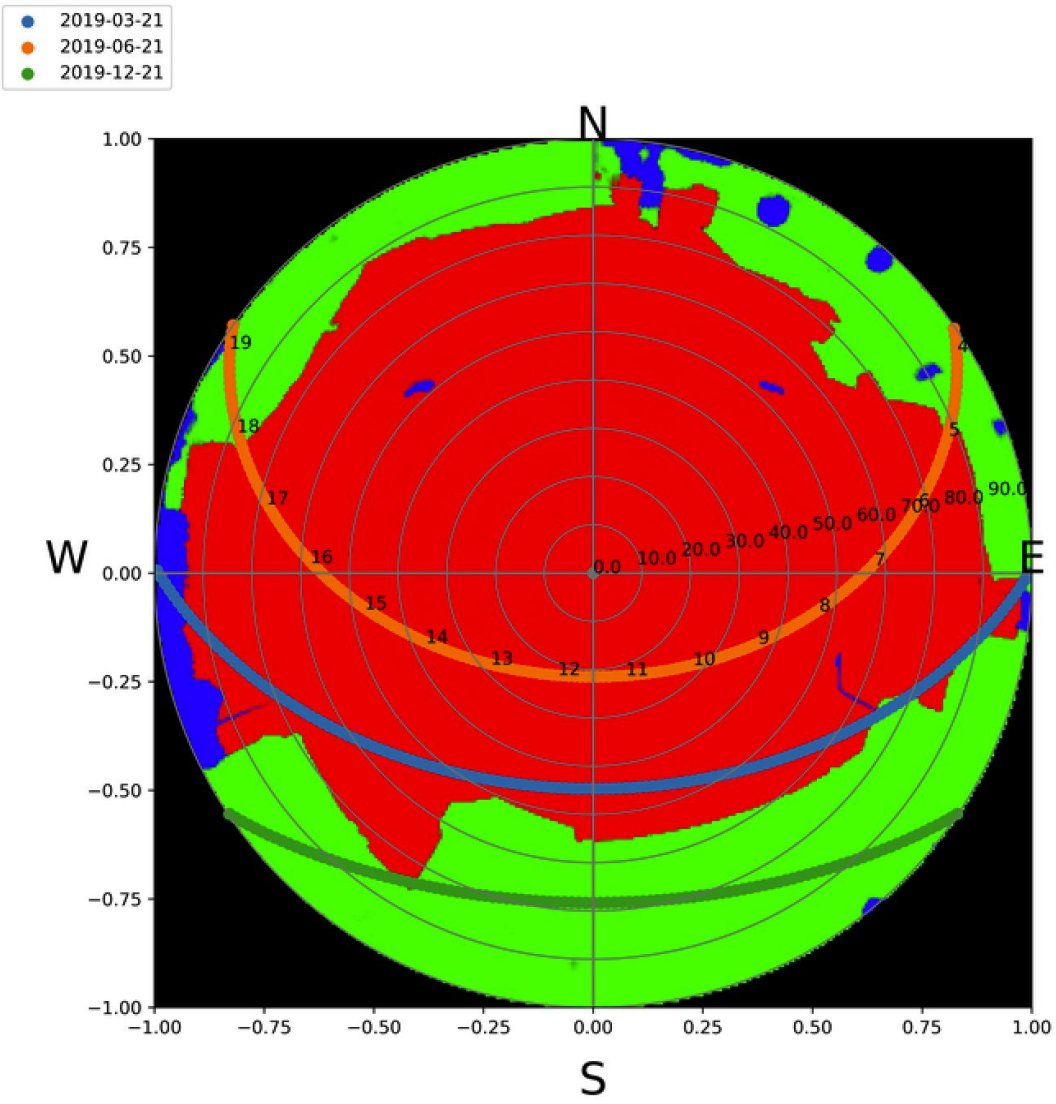


Figure 4-3-1 fisheye image with sun path

When the sun path is marked on the fisheye image, the shaded/shine hours can be figured out. Then organize the Global Horizontal Irradiation (GHI), Diffuse Horizontal Irradiation (DHI) and the sun path data like figure 4-3-2.

	apparent_zenith	zenith	apparent_elevation	elevation	azimuth	equation_of_time	ghi	dhi	sunny/shaded	shaded_value	
2016-06-20 04:00:00	87.688455	87.967363	2.311545	2.032637	58.238585	-1.578974	0.000000	0.000000	shaded	0	0.57
2016-06-20 05:00:00	78.409203	78.487985	11.590797	11.512015	68.287324	-1.587977	1.624225	1.534334	sunny	1	0.57
2016-06-20 06:00:00	68.291561	68.333369	21.708439	21.666631	77.985094	-1.596980	92.441428	51.907546	sunny	1	0.57
2016-06-20 07:00:00	57.777962	57.804574	32.222038	32.195426	87.940202	-1.605982	256.091998	97.202641	sunny	1	0.57
2016-06-20 08:00:00	47.197816	47.215978	42.802184	42.784022	99.055352	-1.614983	430.344707	126.195780	sunny	1	0.57
2016-06-20 09:00:00	36.998978	37.011663	53.001022	52.988337	112.913834	-1.623984	591.283455	145.776310	sunny	1	0.57
2016-06-20 10:00:00	28.044274	28.053241	61.955726	61.946759	132.559890	-1.632984	724.112569	159.145879	sunny	1	0.57
2016-06-20 11:00:00	22.248126	22.255007	67.751874	67.744993	162.366250	-1.641984	818.220921	167.620287	sunny	1	0.57
2016-06-20 12:00:00	22.389042	22.395972	67.610958	67.604028	199.114284	-1.650983	866.493935	171.721562	sunny	1	0.57
2016-06-20 13:00:00	28.376146	28.385239	61.623854	61.614761	228.445371	-1.659982	865.364284	171.627259	sunny	1	0.57
2016-06-20 14:00:00	37.411603	37.424479	52.588397	52.575521	247.759247	-1.668980	814.915303	167.333843	sunny	1	0.57
2016-06-20 15:00:00	47.639246	47.657689	42.360754	42.342311	261.454550	-1.677977	718.878151	158.653253	sunny	1	0.57
2016-06-20 16:00:00	58.223968	58.251040	31.776032	31.748960	272.495831	-1.686973	584.525441	145.040431	sunny	1	0.57
2016-06-20 17:00:00	68.727122	68.769849	21.272878	21.230151	282.424919	-1.695969	422.631938	125.125144	sunny	1	0.57
2016-06-20 18:00:00	78.820469	78.902005	11.179531	11.097995	292.126581	-1.704964	248.235613	95.577941	shaded	0	0.57
2016-06-20 19:00:00	88.040419	88.345507	1.959581	1.654493	302.201133	-1.713958	86.006920	49.363475	shaded	0	0.57

Figure 4-3-2 data organized

As the figure 4-3-2 shows, the values of different element have timestamp as the index. By this way, it is comfortable to plot the value of a specific day as figure 4-3-3.

As figure 4-3-3 shows, there are five lines representing five elements:

GHI : Global Horizontal Irradiation;

DHI : Diffuse Horizontal Irradiation;

direct_hr : Direct Horizontal irradiation;

direct_only : only the direct irradiation component;

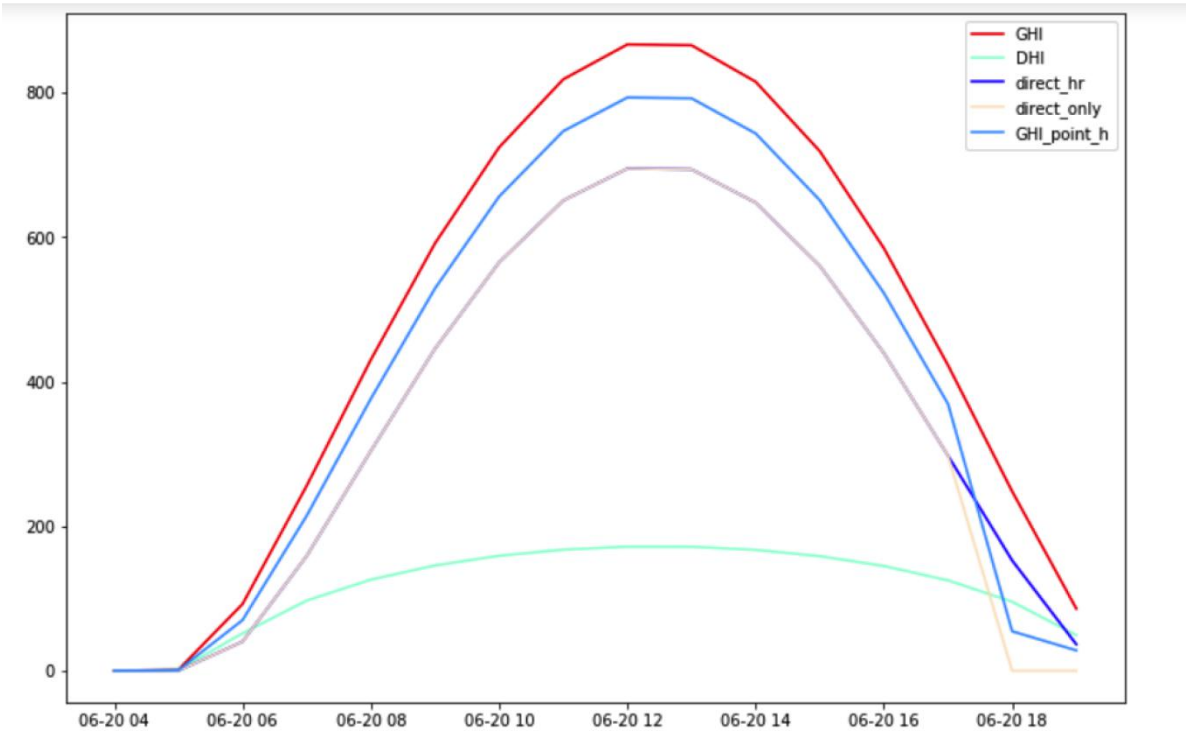


Figure 4-3-3

As we can see, GHI includes DHI and direct_hr, so the GHI is on top of others. GHI_point_h and direct_only are from SVF, GHI and sun path, so the results are effected by the sun path, in other word, by the time of the day.

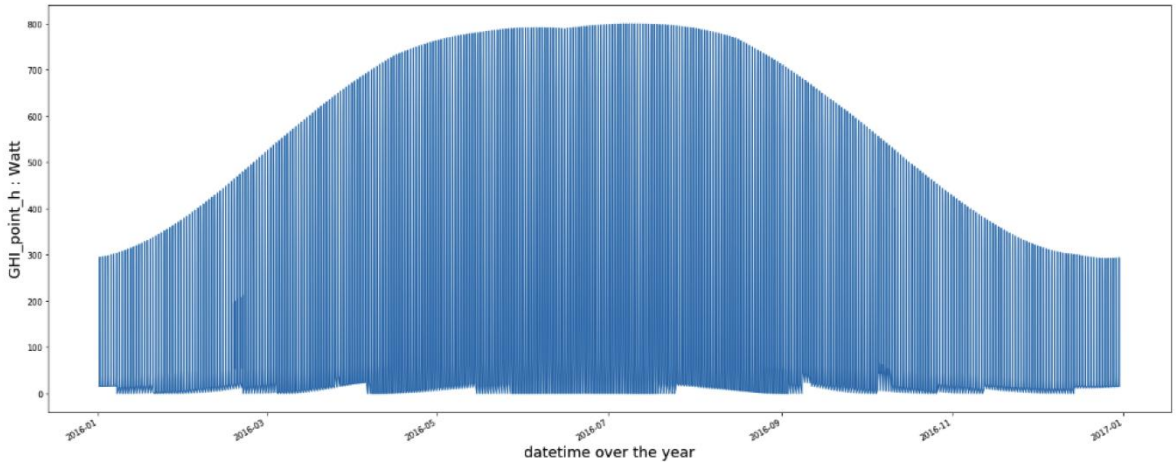


Figure 4-3-4

The last result is the yearly results. For example, the GHI_point_h can be showed as figure 4-3-4:

4.4 Multipoint analysis

The sampled point are in London, Madrid, New York, Paris and Rome respectively. Then use the application to analyze the points.

Firstly, the figure 4-4-1 to figure 4-4-5 shows the results of fisheye image with sun path and energy analysis.

According to the result of fisheye image with sun path, the parameters are figured out and plotted as figures following. The figure a shows the sun path, which help us to find the shaded/shined hours over a day, and the figure b shows the distribution of each parameter depending on the shined hours.

Figure 4-4-1-a panoramic image point in New York

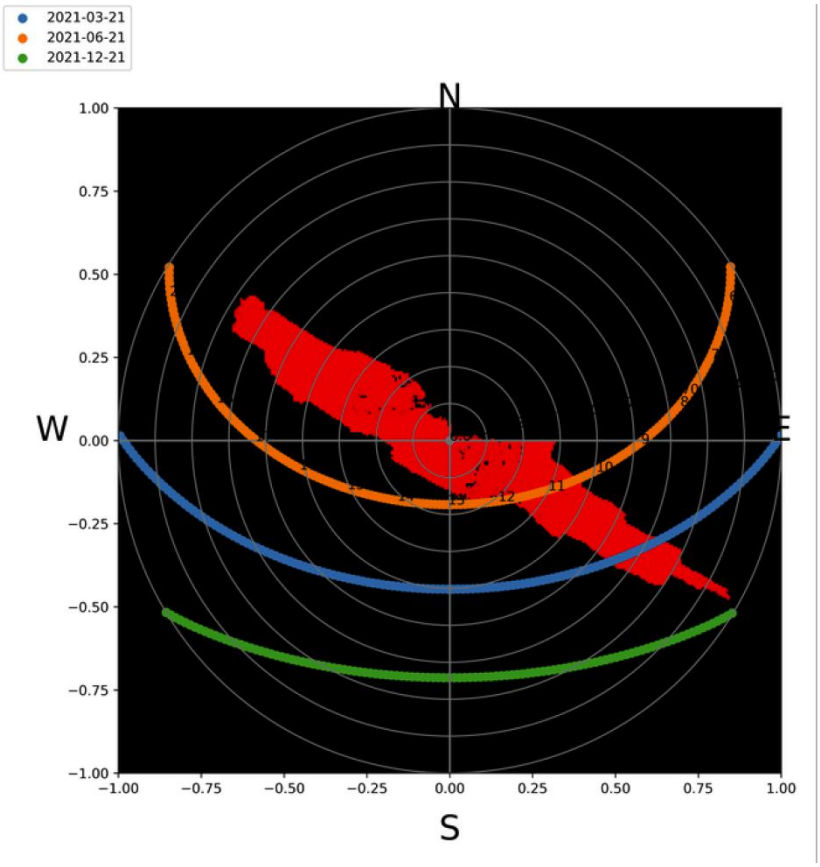


Figure 4-4-1-b point in New York

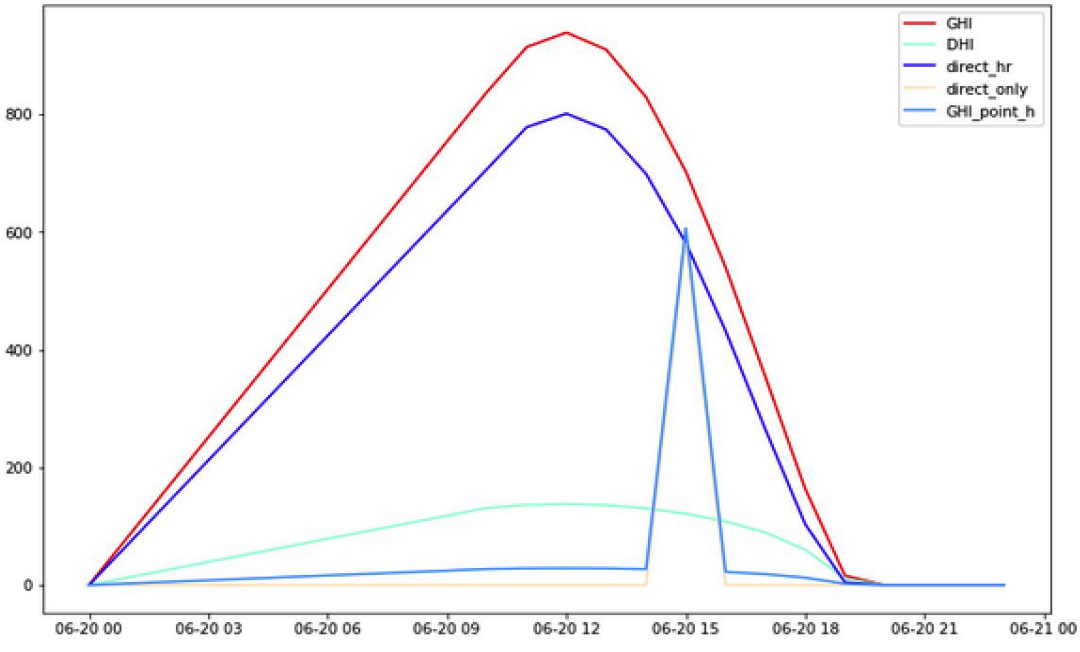


Figure 4-4-1-c point in New York

Figure 4-4-2-a panoramic image point in Paris

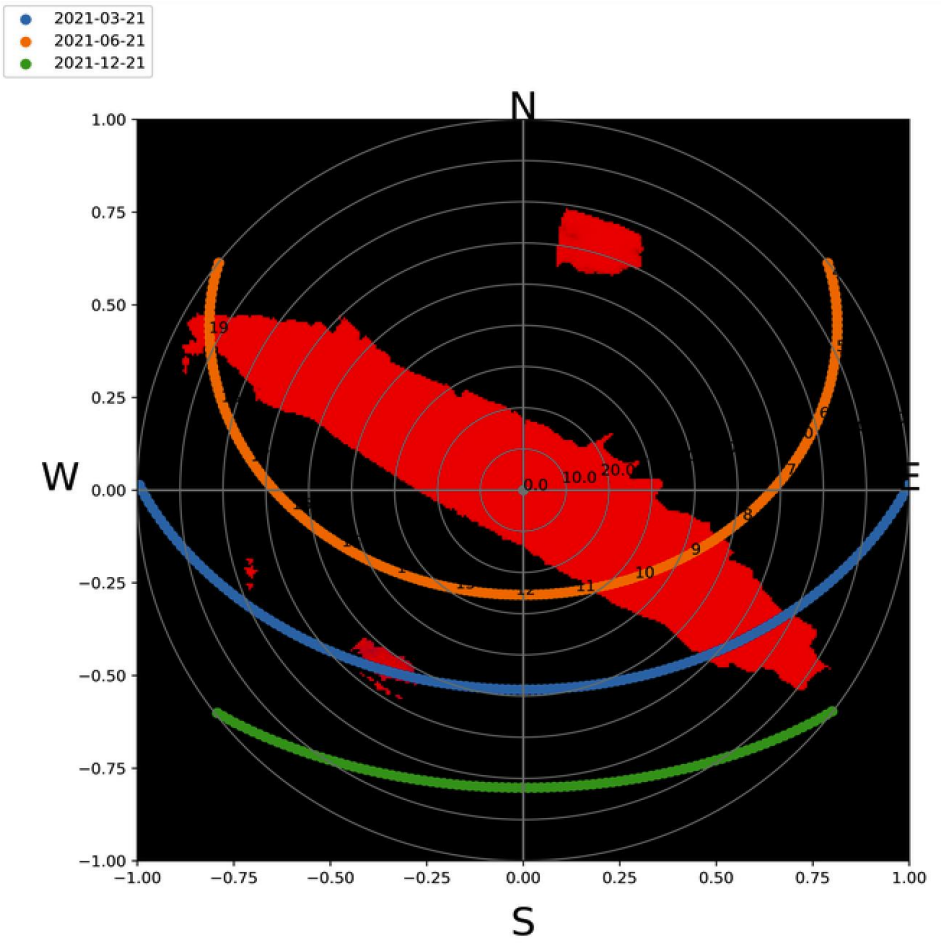


Figure 4-4-2-b point in Paris

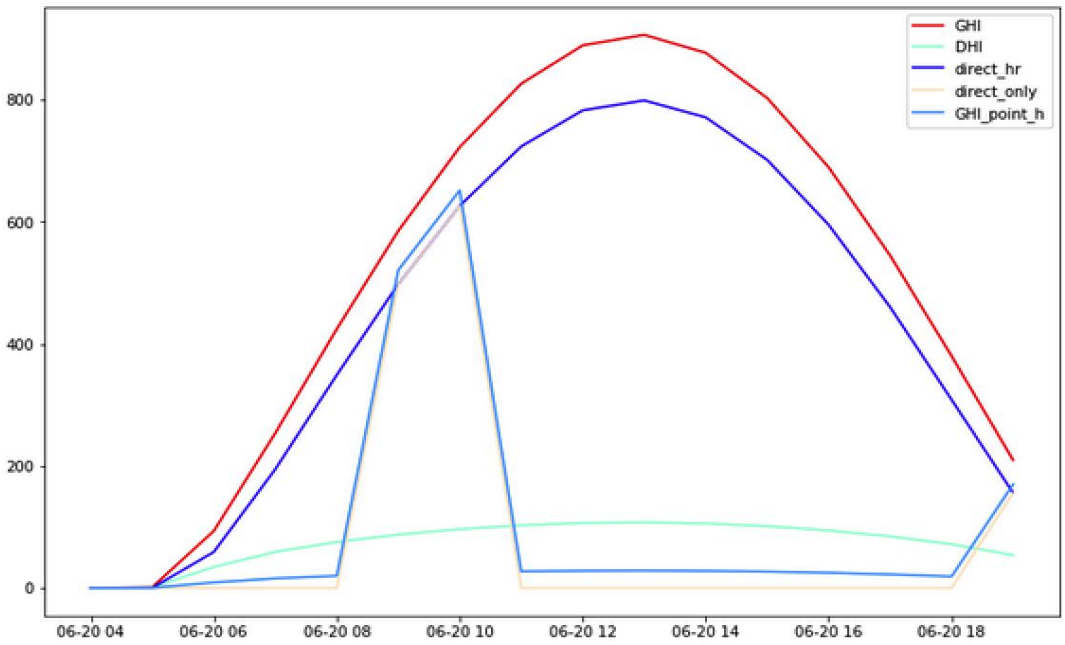


Figure 4-4-2-c point in Paris

Figure 4-4-3-a panoramic image point in Madrid

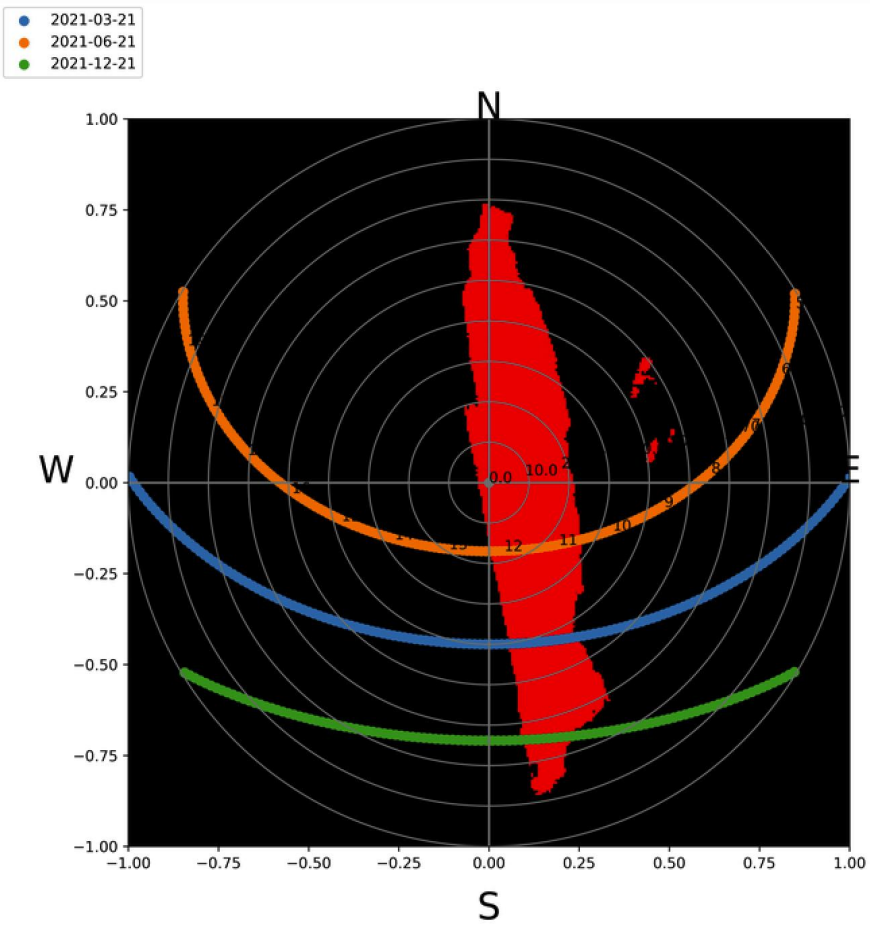


Figure 4-4-3-b point in Madrid

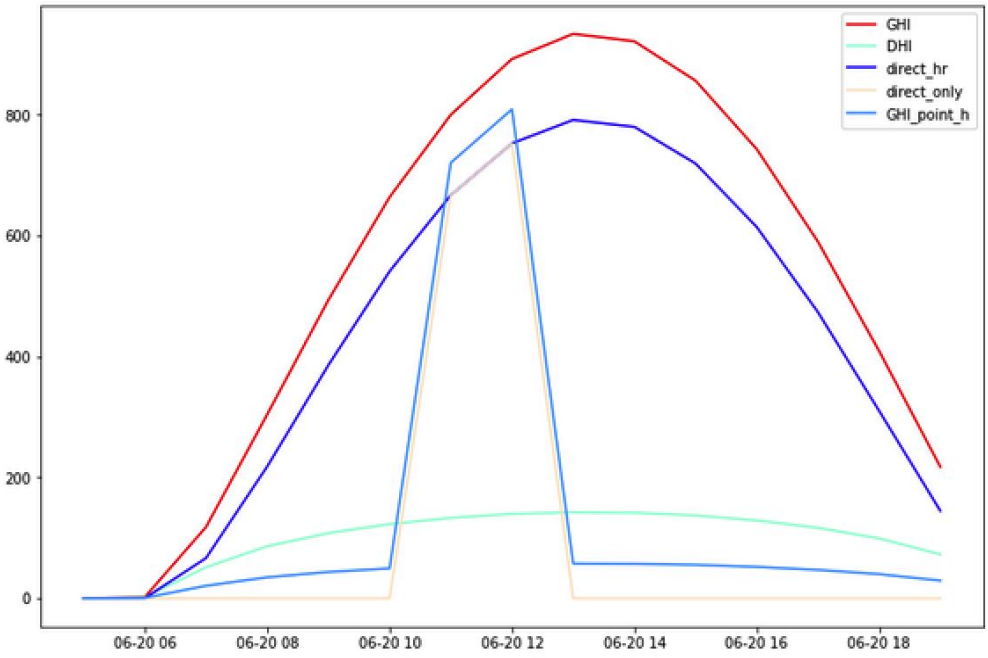


Figure 4-4-3-c point in Madrid

Figure 4-4-4-a panoramic image point in London

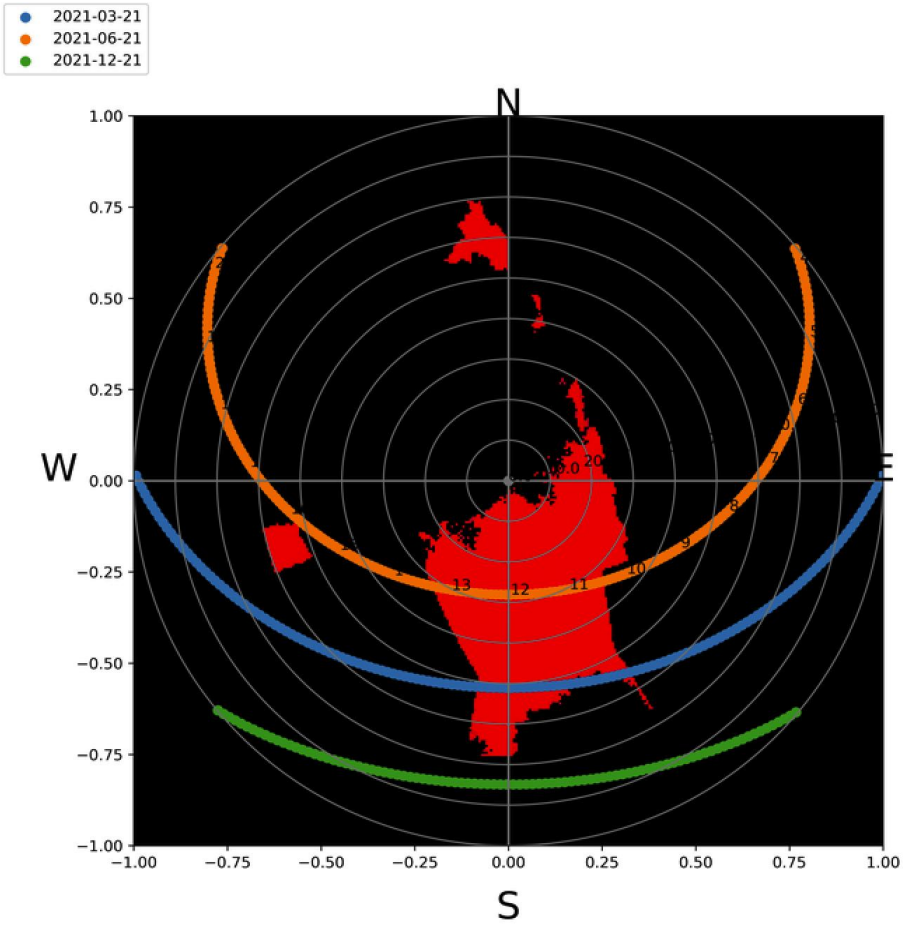


Figure 4-4-4-b point in London

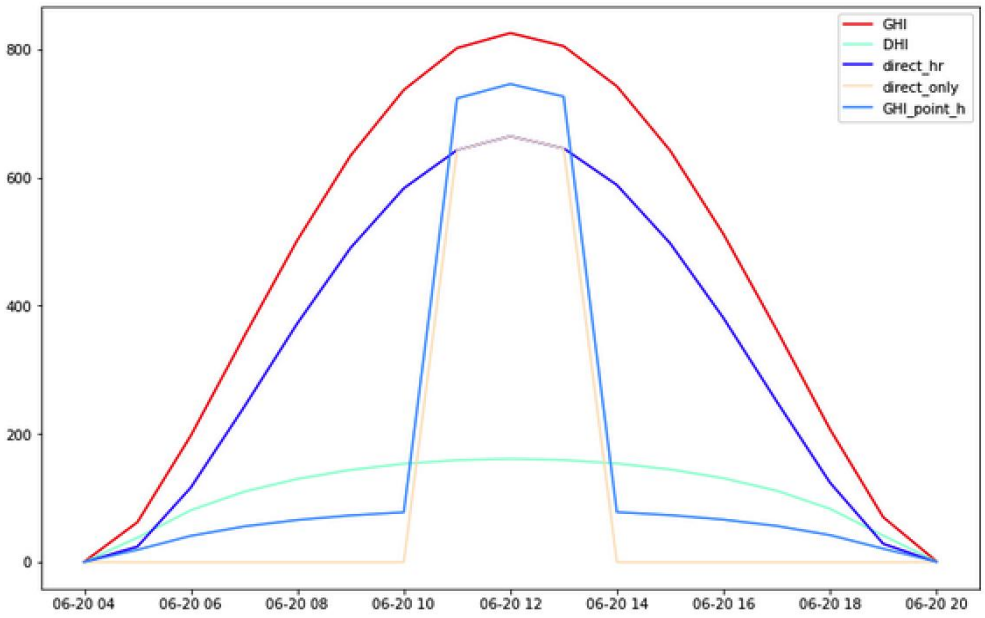


Figure 4-4-4-c point in London

Figure 4-4-5-a panoramic image point in Roma

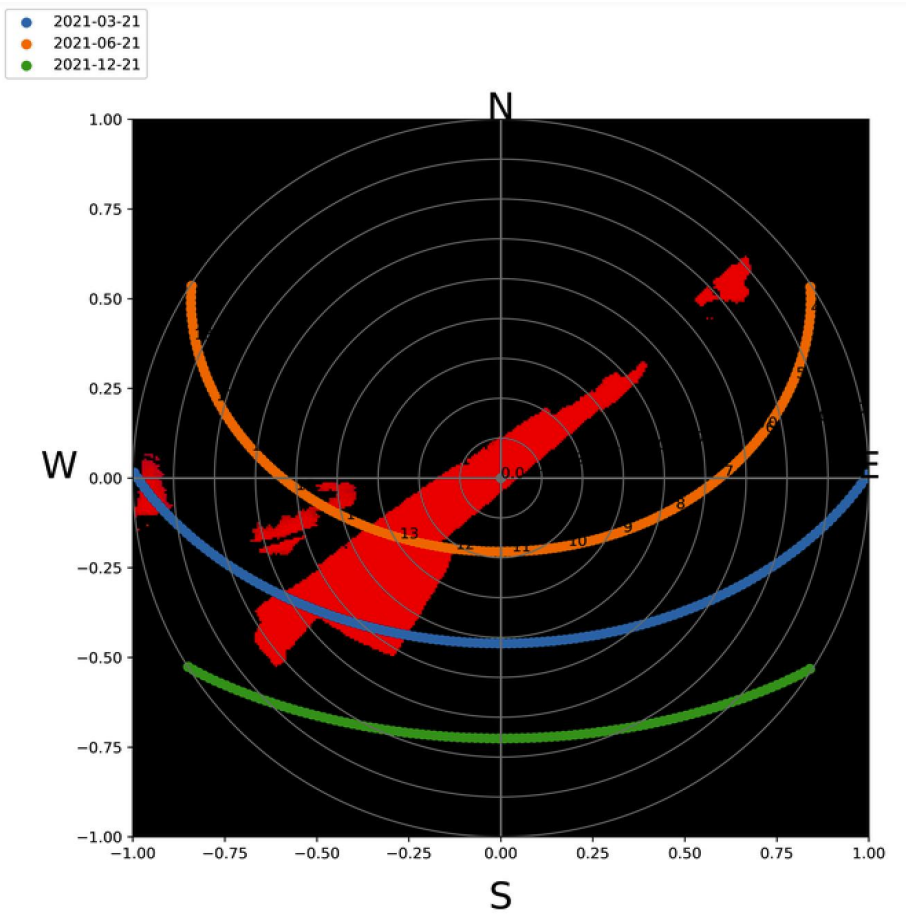


Figure 4-4-5-a point in Roma

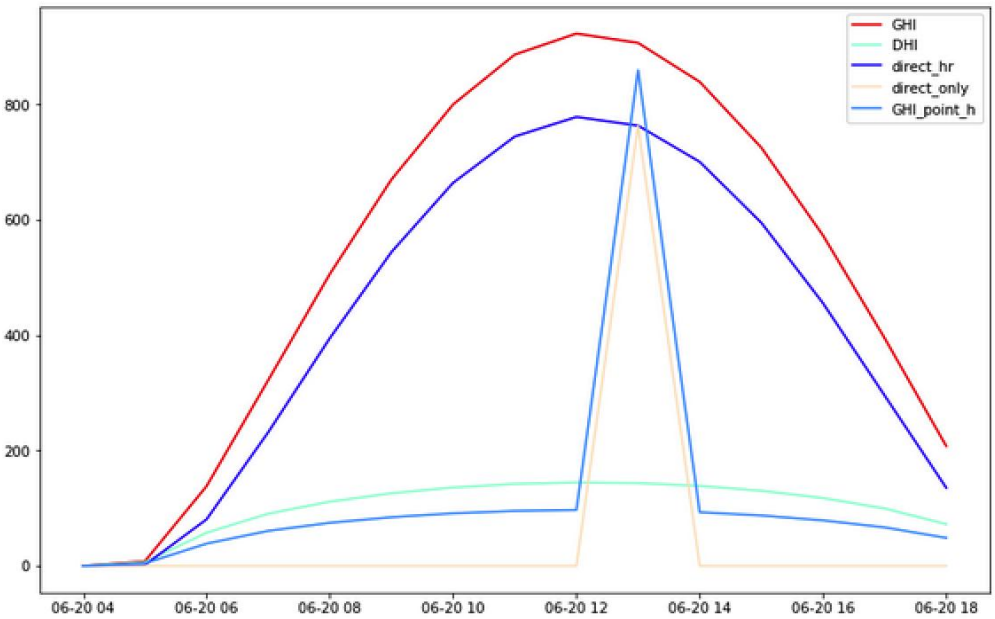


Figure 4-4-5-b point in Roma

5. Discuss and Conclusions

Presented works and methodology have been developed and verified. At step 1, the Google map downloading is tested with a good results, for every city in Europe, the Google Street View is able to provide a panorama. In Africa and most Middle East countries, the GSV dose not work. In the future, with the cooperation between Google and the government, the GSV may cover the cities in those countries. Then at step 3, segmentation, the work on the images requested from GSV, also has the accuracy to evaluate the predicted result. That can be seen in figure 3-2-4 and figure 3-2-5. At the last step, the data of solar radiation, we compared the method between by requesting data from pvlib and by epw database. Both of them are feasible. Nevertheless, not all

location has a file epw, but furthermore, through the tool and API, service may produce the *epw interpolated data by coordinates.

By building this tool, the most direct benefit is to realize SVF analysis more conveniently. Firstly, it solves the problem that only a single SVF can be analyzed in a specific location for once, that means if there is any demand for various points, people do not need to collect the data point by point in city. And also it makes the user free from the constraints of physical devices and in site of picture collection.

But there are still some problems to be considered, the system is able to provide the static data, not the domain. It can meet the needs that only need simple SVF data, for example, the SVF distribution over the whole city, or some area.

Considering this issue, in the future, it would be possible to upgrade existing urban comfort and environmental monitoring kits - e.g. the recent developed IoT urban mobile monitoring comfort kit, see (Chiesa et al. 2021) - by including the possiblity to retrieve the SVF. It is, in fact, possible to use the methodology introduced in this thesis, using both the coordinate-driven google image elaboration, or including a fisheye camera, to allow the automatic detection of SVF values near to other urban comfort variables, as temperature, humidity, wind velocity and direction, long and short wave radiation, etc. People would be able to carry the kit to collect data in city. So the work should move on the camera segmentation that is called instance segmentation.

Acknowledge

This is a very interesting job. From the beginning to the preparation of the thesis, it gives me a lot of experience and let me learn more knowledge and skills. Here, I would like to express my warmest thanks to my tutor, professor Giaccamo Chiesa. He is in wisdom and erudited. With his help, most problem concerning the theory and the questions about the thesis are solved. Additionally, he gave me a lot of good suggestions. Moreover, I would like to thank our ICT's professors for giving me a reserve of skills and laying the foundation for my practice.

Reference:

[1]Souza, L.C.L.; Rodrigues, D.S.; Mendes, J.F.G. Sky-view factors estimation using a 3D-GIS extension. In Proceedings of the 8th International IBPSA Conference, Eindhoven, The Netherlands, 11–14 August 2003.

[2]Sky View Factor Calculation in Urban Context: Computational Performance and Accuracy Analysis of Two Open and Free GIS Tools.

[3]C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in CVPR, pp. 1–9, 2015.

[4]K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” CoRR, vol. abs/1409.1556, 2014.

[5]LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436–444.

[6]Souza, L.C.L.; Rodrigues, D.S.; Mendes, J.F.G. Sky-view factors estimation using a 3D-GIS extension. In Proceedings of the 8th International IBPSA Conference, Eindhoven, The Netherlands, 11–14 August 2003.

[7]Oke, T.R. Canyon geometry and the nocturnal urban heat island: Comparison of scale model and field observations. J. Climatol. 1981, 1, 237–254. doi:10.1002/joc.3370010304.

[8]Bradley, A. V., Thornes, J. E., & Chapman, L. (2001). A method to assess the variation of urban canyon geometry from sky view factor transects. Atmospheric Science Letters, 2(1–4), 155–165.

[9]Chapman, L., & Thornes, J. E. (2004). Real-time sky-view factor calculation and approximation. Journal of Atmospheric and Oceanic Technology, 21(5), 730– 741.

[10]Chapman, L., Thornes, J. E., & Bradley, A. V. (2001). Rapid determination of canyon geometry parameters for use in surface radiation budgets. Theoretical and Applied Climatology, 69(1), 81–89.

[11]Chen, L., Ng, E., An, X., Ren, C., Lee, M., Wang, U., & He, Z. (2012). Sky view factor analysis of street canyons and its implications for daytime intra-urban air temperature differentials in high-rise, high-density urban areas of Hong Kong: A GIS-based simulation approach. International Journal of Climatology, 32(1), 121– 136.

[12]Gál, T., Lindberg, F., & Unger, J. (2009). Computing continuous sky view factors using 3D urban raster and vector databases: comparison and application to urban climate. Theoretical and Applied Climatology, 95(1), 111–123.

[13]Gál, T., & Unger, J. (2014). A new software tool for SVF calculations using building and tree-crown databases. Urban Climate, 10, 594– 606.

[14]Grimmond, C. S. B., Potter, S. K., Zutter, H. N., & Souch, C. (2001). Rapid methods to estimate sky-view factors applied to urban areas. International Journal of Climatology, 21(7), 903– 913.

[15]Hämmerle, M., Gál, T., Unger, J., & Matzarakis, A. (2011). Comparison of models calculating the sky view factor used for urban climate investigations. Theoretical and [16]Applied Climatology, 105(3), 521– 527.

[16]Helbig, N., Löwe, H., & Lehning, M. (2009). Radiosity approach for the shortwave surface radiation balance in complex terrain. Journal of the Atmospheric Sciences, 66(9), 2900– 2912.

[17]Helbig, N., Mott, R., Herwijnen, A., Winstral, A., & Jonas, T. (2017). Parameterizing surface wind speed over complex topography. 122(2), 651– 667.

[18]Johnson, G. T., & Watson, I. D. (1984). The determination of view-factors in urban canyons. Journal of Climate and Applied Meteorology, 23(2), 329– 335.

[19]Kokalj, Ž., Zakšek, K., & Oštir, K. (2011). Application of sky-view factor for the visualisation of historic landscape features in lidar-derived relief models. Antiquity, 85(327), 263– 273.

[20]Liang, J., Gong, J., Sun, J., & Liu, J. (2017). A customizable framework for computing sky view factor from large-scale 3D city models. Energy and Buildings, 149, 38– 44.

[21]Lin, T.-P., Matzarakis, A., & Hwang, R.-L. (2010). Shading effect on long-term outdoor thermal comfort. Building and Environment, 45(1), 213– 221.

[22]Lindberg, F., & Grimmond, C. S. B. (2010). Continuous sky view factor maps from high resolution urban digital elevation models. Climate Research, 42(3), 177– 183.

[23]Lindberg, F., Grimmond, C. S. B., Gabey, A., Huang, B., Kent, C. W., Sun, T., Theeuwes, N. E., Järvi, L., Ward, H. C., Capel-Timms, I., Chang, Y., Jonsson, P., Krave, N., Liu, D., Meyer, D., Olofson, K. F. G., Tan, J., Wästberg, D., Xue, L., & Zhang, Z. (2018). Urban Multi-scale Environmental Predictor (UMEP): An integrated tool for city-based climate services. Environmental Modelling & Software, 99, 70– 87.

[24]Lindberg, F., Holmer, B., & Thorsson, S. (2008). SOLWEIG 1.0—Modelling spatial variations of 3D radiant fluxes and mean radiant temperature in complex urban settings. International Journal of Biometeorology, 52(7), 697– 713.

[25]Park, S., & Tuller, S. E. (2014). Advanced view factor analysis method for radiation exchange. International Journal of Biometeorology, 58(2), 161–178.

Scarano, M., & Mancini, F. (2017). Assessing the relationship between sky view factor and land surface temperature to the spatial resolution. International Journal of Remote Sensing, 38(23), 6910– 6929.

[26]B. Holmer, U. Postgård & M. Eriksson.(2001). Sky view factors in forest canopies calculated with IDRISI. Theoretical and Applied Climatology volume, 68, 33–40.

[27]Matzarakis A (2001) Die thermische Komponente des Stadtklimas. Berichte des Meteorologischen Institutes der Universität Freiburg. Nr. 6

[28]Lin T-P, Matzarakis A, Hwang RL (2010) Shading effect on long-term outdoor thermal comfort. Build Environ 45:213–211

[29]Hämmerle, M.; Gál, T.; Unger, J.; Matzarakis, A. Comparison of models calculating the sky view factor used for urban climate investigations. Theor. Appl. Climatol. 2011, 105, 521–527.

[30]Matzarakis, A.; Matuschek, O. Sky View Factor as a parameter in applied climatology—Rapid estimation by the SkyHelios Model. Meteorol. Z. 2011, 20, 39–45.

[31]Lo, C.-M., Lee, C.-F., & Keck, J. (2017). Application of sky view factor technique to the interpretation and reactivation assessment of landslide activity. Environmental Earth Sciences, 76(10).

[32]Chen, L., Ng, E., An, X., Ren, C., Lee, M., Wang, U., & He, Z. (2012). Sky view factor analysis of street canyons and its implications for daytime intra-urban air temperature differentials in high-rise, high-density urban areas of Hong Kong: A GIS-based simulation approach. International Journal of Climatology, 32(1), 121– 136.

[33]Helbig, N., Mott, R., Herwijnen, A., Winstral, A., & Jonas, T. (2017). Parameterizing surface wind speed over complex topography. 122(2), 651– 667.

[34]Blennow, K. (1995). Sky view factors from high-resolution scanned fish-eye lens photographic negatives. Journal of Atmospheric and Oceanic Technology, 12(6), 1357– 1362.

[35]Bradley, A. V., Thornes, J. E., & Chapman, L. (2001). A method to assess the variation of urban canyon geometry from sky view factor transects. Atmospheric Science Letters, 2(1–4), 155– 165.

[36]Grimmond, C. S. B., Potter, S. K., Zutter, H. N., & Souch, C. (2001). Rapid methods to estimate sky-view factors applied to urban areas. International Journal of Climatology, 21(7), 903– 913.

[37Chapman, L., & Thornes, J. E. (2004). Real-time sky-view factor calculation and approximation. Journal of Atmospheric and Oceanic Technology, 21(5), 730– 741.

[38]Zhong-Hu Jiao,Huazhong Ren,Xihan Mu,Jing Zhao,Tianxing Wang,Jiaji Dong, （2019） Evaluation of Four Sky View Factor Algorithms Using Digital Surface and Elevation Model Data

[39]Matzarakis, A., Rutz, F., & Mayer, H. (2010). Modelling radiation fluxes in simple and complex environments: Basics of the RayMan model. International Journal of Biometeorology, 54(2), 131– 139.

[40]Manners, J., Vosper, S. B., & Roberts, N. (2012). Radiative transfer over resolved topographic features for high-resolution weather prediction. Quarterly Journal of the Royal Meteorological Society, 138(664), 720– 733.

[41]Dozier, J., & Frew, J. (1990). Rapid calculation of terrain parameters for radiation modeling from digital elevation data. IEEE Transactions on Geoscience and Remote Sensing, 28(5), 963– 969.

[42]Giacomo Chiesa, Luo Yingjun, Sheng Yuxuan, Zhang Bolun, (2021). Development and initial tests of an urban comfort monitoring system
Development and initial tests of an urban comfort monitoring system