Software Development for Producing Texture Images Mapped on a Building Surface of a 3D City Model Using Aerial Images

Ryuji Matsuoka [1], Masato Ishikawa [1], Tomoaki Inazawa [1], Yoshihiko Nakanishi [1], Futa Kawamata [1], Masahito Takada [1], Takuya Danjo [1]

[1] Kokusai Kogyo Co., Ltd., 2-24-1 Harumi-cho, Fuchu-shi, Tokyo 183-0057, Japan - {ryuji_matsuoka, masato_ishikawa, tomoaki_inazawa, yoshihiko_nakanishi, futa_kawamata, masahito_takada, takuya_danjo}@kk-grp.jp

Keywords: Automatic texture mapping, Building surface, 3D City Model, Aerial nadir image, Aerial oblique image, Occlusion.

Abstract

It is desirable that a 3D city model at level of detail 2 (LOD2) has texture images mapped on building surfaces. Owing to the cost of image collection, it would be the best way to use aerial images for texture mapping at present. Although aerial oblique images provide higher-resolution texture images, using aerial oblique images has a major issue of occlusion. Accordingly, we develop software for texture mapping to a 3D city model using aerial nadir and oblique images, aiming to minimize the impact of occlusion. The software designed to be used in ordinary operation includes the features of automatically detecting occlusions on building surfaces within images by utilizing the geometry of a 3D city model and automatically selecting appropriate oblique and nadir images for texture mapping. The major feature of the developed software is its ability to process grid by grid on a building surface. The validation experiment results confirm the software's satisfactory performance in practice. Moreover, the experiment results indicate that the performance of the software depends on the ability of a 3D city model to represent buildings. Since we have recognized that it would be effective if each pixel of a texture image has its own resolution, we plan to modify the software so that each pixel can have its own resolution.

1. Introduction

3D city model data in more than 250 Japanese cities is currently provided by Project PLATEAU, which is a project started by the Ministry of Land, Infrastructure, Transport and Tourism of Japan (MLIT) in 2020. The project aims to construct 3D city models as a platform for urban activities, provide 3D city model data as open access data, and promote the utilization of 3D city models (MLIT, 2025a).

Project PLATEAU recommends that a 3D city model at level of detail 2 (LOD2) should have texture images mapped on building surfaces to make the appearance of buildings of the 3D city model realistic (MLIT, 2025b). At present, the production of a 3D city model with texture in Japan utilizes mostly aerial nadir images, which provide the geometry of a 3D city model from the aspect of the production cost of a 3D city model.

The photorealistic 3D city models are utilized in various geospatial applications related to virtual city tourism, 3D GIS, urban planning, and real estate management. Texture images mapped on building surfaces of a 3D city model are utilized not only for providing a realistic appearance but also for enabling some analyses, such as investigating stories of a building and detecting doors and windows on a building façade.

Texture images of most 3D city model data in Japan have been produced using the aerial nadir images collected for producing 3D city models However, Ishikawa et al. (2025) demonstrated that the resolution of a texture image produced using such aerial nadir images would be insufficient for detecting doors and windows on a building façade. On the contrary, they suggested that the resolution a texture image produced using aerial oblique images would be sufficient using the resolution of a texture image produced using such aerial sufficient for detecting doors and windows on a building façade.

However, using an oblique image for texture mapping has a major issue that occlusion caused by buildings on the side of a camera in an oblique image may occur more frequently than in a nadir image. Additionally, occlusion would be unavoidable in texture mapping of building surfaces in a built-up area. Figure 1 illustrates an oblique image that exhibits occlusions.

Figure 1. Occlusions in an aerial oblique image

On ensuring the sufficient resolution of a texture image, the use of UAV images or terrestrial images would be one of the means to solve the issue of occlusion in texture mapping using aerial oblique images. However, owing to the cost of image collection, we consider that it would be the best way to use aerial images for texture mapping at present.

Accordingly, we decided to develop software that will use both aerial nadir and oblique images for texture mapping on a single surface, aiming to minimize the impact of occlusion. The software is designed to be used in ordinary operation. The software will include a feature that detects occlusions on building surfaces within images by utilizing the geometry of a 3D city model. It will also automatically select appropriate oblique and nadir images for texture mapping.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-4-2026-45-2026

There are some studies on producing a texture image of a 3D city model (He et al., 2022, Buyukdemircioglu and Oude Elberink, 2024). He et al. (2022) presented a framework to produce a texture image using terrestrial images. We judged that their framework would be too complicated and highly time-consuming to ensure the performance of the software using aerial images in ordinary operation. Buyukdemircioglu and Oude Elberink (2024) proposed an automated method for automated texture mapping of 3D city models from nadir and oblique aerial imagery. The proposed method selects the optimal image based on the visibility of the whole of the building surface. Since there are some building surfaces, a part of which is occluded in urban areas, we decided that our software does not adopt their proposed method.

The paper presents the outline of the processing of the developed software and the outline of an experiment conducted for investigating the validity of the developed software.

2. Outline of the processing of the developed software

Most current software for automatic texture mapping of building surfaces in a 3D city model using aerial images is designed to use only aerial nadir images. On the contrary, our developed software uses aerial nadir and oblique images collected by a multi-directional camera system such as the Vexcel Imaging UltraCam Osprey 4.1 and the Leica CityMapper-2 to reduce the effect of occlusion. Therefore, the software requires some complicated processing as mentioned below.

Figure 2 shows the basic concept of using a nadir and oblique image for texture mapping against occlusion. The upper part of the texture image on the left façade of the right building is produced using an oblique image, while the lower part is produced using a nadir image.

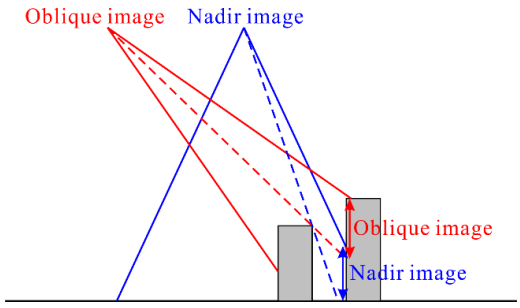


Figure 2. Using a nadir and oblique image for texture mapping

2.1 Processing flow of the developed software

Most current software selects an image to be mapped to a building surface polygon by polygon. Hence, a part of the neighbor building may be found in a mapped texture image with no consideration of occlusion. On the other hand, the developed software selects an image to be mapped to a building surface grid by grid that is established on each building surface.

Figure 3 shows the process flow of the developed software. The process flow shown in Figure 3 is a little complicated to reduce the computation time. [Image], [Polygon], and [Grid] in Figure 3 indicate the processing image by image, polygon by polygon, and grid by grid, respectively. The software establishes an equally spaced grid of points on each building surface where the visibility will be determined.

The key algorithms of the software include one that determines the visibility of a grid on a building surface and another that selects an image to be mapped to the grid.

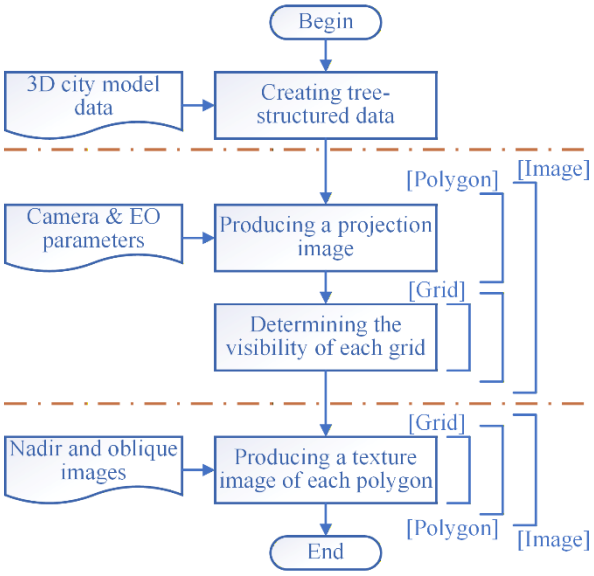


Figure 3. Process flow of the developed software

2.2 Determining the visibility of a point on a building surface

Determining the visibility of a point on a building surface in an urban area with densely distributed buildings is a highly time-consuming task.

2.2.1 Basic idea: There are two ways to determine whether a point on a building surface is visible from a camera. One is based on the idea that “whether we can see the camera or not” indicates “whether we are photographed by the camera or not.” The other is based on the idea that “whether the camera sees us or not” indicates “whether we are photographed by the camera or not.”

The former idea can be realized by ray tracing or by using a hemispherical viewshed, which is the angular distribution of sky obstruction. Figure 4 shows a hemispherical viewshed, which is a fisheye lens image viewed from a point on the earth. We call the method based on the former idea "Method A" in the paper.

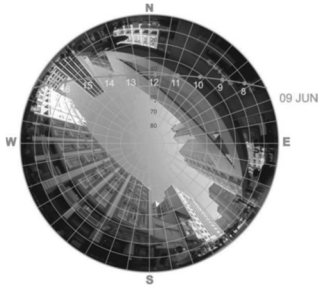


Figure 4. Fisheye image viewed from a point on the earth (Minella et al., 2011)

Conversely, we can implement the latter idea through the use of projection images captured by a camera. We call the method based on the latter idea "Method B" in the paper. From now on, we discuss the algorithm for determining visibility on a building surface.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-4-2026-45-2026 | © Author(s) 2026. CC BY 4.0 License.

Figure 5 shows an aerial image collected by an oblique camera with an off-nadir angle of 45°. Figure 6 shows a projection image corresponding to the real aerial image shown in Figure 5. The projection image in Figure 6 is created by using a 3D city model and the exterior orientation parameters of the aerial image shown in Figure 5. Each building surface in Figure 6 is colored according to its ID. From now on, an aerial image corresponding to a projection image is called a target image.

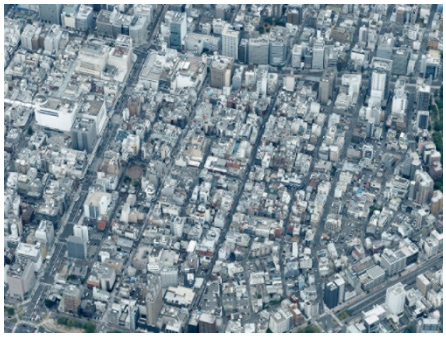


Figure 5. Aerial image collected by an oblique camera with the off-nadir angle of 45°

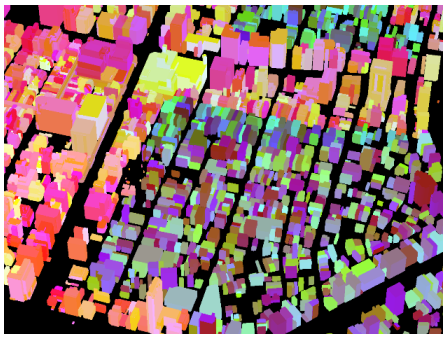


Figure 6. Projection image viewed by an oblique camera with the off-nadir angle of 45°

Using computer graphics (CG) techniques such as the depth buffer (Z-buffer) algorithm, which does not examine the intersection of the sun ray, can provide projection images viewed by the sun much more rapidly. Additionally, using another CG technique named the scanline algorithm for filling polygons can create a projection image more rapidly.

Roughly speaking, the computation time by Method A would be proportional to the square of the number of polygons of buildings in a 3D city model, while that by Method B would be proportional to the number of polygons. However, Method A directly provides information on the visibility of a point on a building surface, while a projection image in Method B cannot provide the information on the visibility directly. Method B requires additional processing to determine the visibility of each point on the surface except for producing a projection image. The computation time of the additional processing would be proportional to the number of polygons. Accordingly, we adopt Method B from the perspective of computation time.

2.2.2 Image geometry: Before showing the processing flow of Method B, we discuss image geometry. The image coordinate system is considered a planar rectangular coordinate system (x, y), while the object space coordinate system is regarded as a spatial rectangular coordinate system (X, Y, Z), as illustrated in Figure 7.

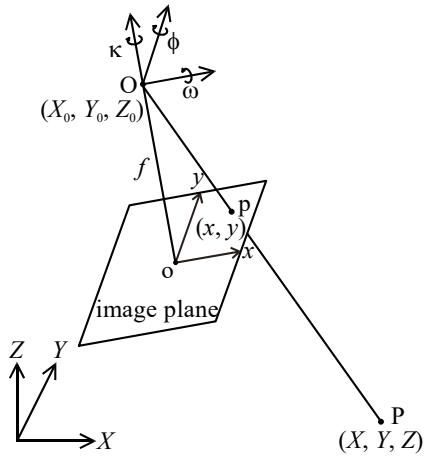


Figure 7. Image coordinate system (x, y) and object space coordinate system (X, Y, Z)

Consider that an image is taken by a camera with focal length f and pixel size δ. (X_O, Y_O, Z_O) and (ω, ϕ, κ) represent the position and attitude of the camera in the object space coordinate system (X, Y, Z). The rotation matrix $R[r_{ij}]$ of the camera is as follows:

$$R = R_\kappa R_\phi R_\omega \tag{1}$$

where

$$R_\omega = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \tag{2}$$

$$R_\phi = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{3}$$

$$R_\kappa = \begin{bmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Let the point $P(X, Y, Z)$ be projected to the image point $p(x, y)$ on an aerial image. Here the image point $p(x, y)$ is as follows:

$$x = -f\frac{u}{w} \tag{5}$$

$$y = -f\frac{v}{w} \tag{6}$$

where

$$u = r_{11}(X - X_O) + r_{12}(Y - Y_O) + r_{13}(Z - Z_O) \tag{7}$$

$$v = r_{21}(X - X_O) + r_{22}(Y - Y_O) + r_{23}(Z - Z_O) \tag{8}$$

$$w = r_{31}(X - X_O) + r_{32}(Y - Y_O) + r_{33}(Z - Z_O) \tag{9}$$

Here the viewing vector $\vec{V}(u, v, w)$ represents the vector from the exposure center O to the point P, and w expresses the depth.

Let (c_O, r_O) be the column and the row of the principal point in the image. The column c and the row r of the image point $p(x, y)$ are as follows:

$$c = \frac{x}{\delta} + c_O \tag{10}$$

$$r = -\frac{y}{\delta} + r_O \tag{11}$$

Let $\vec{N}(n_X, n_Y, n_Z)$ be the exterior normal vector of the plane on which a polygon lies. The equation of the plane can be expressed as follows:

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-4-2026-45-2026 | © Author(s) 2026. CC BY 4.0 License.

$$n_X X + n_Y Y + n_Z Z + d = 0 \quad (12)$$

The incident angle θ of the viewing vector $\vec{V}$ to the polygon can be expressed as follows:

$$\cos\theta = \frac{\vec{N}\cdot\vec{V}}{|\vec{N}||\vec{V}|} = \frac{n_X u + n_Y v + n_Z w}{\sqrt{n_X^2+n_Y^2+n_Z^2}\sqrt{u^2+v^2+w^2}} \quad (13)$$

The equation $w = F(c,r)$ for calculating the depth w of a pixel (c,r) on the projection image is as follows:

$$w = F(c,r) = \frac{g_1 f}{g_x\delta(c-c_O) - g_y\delta(r-r_O) + g_z f} \quad (14)$$

where

$$g_x = r_{11}n_X + r_{12}n_Y + r_{13}n_Z \quad (15)$$
$$g_y = r_{21}n_X + r_{22}n_Y + r_{23}n_Z \quad (16)$$
$$g_z = r_{31}n_X + r_{32}n_Y + r_{33}n_Z \quad (17)$$
$$g_1 = n_X X_o + n_Y Y_o + n_Z Z_o + d \quad (18)$$

2.2.3 Processing flow: Method B consists of three steps to obtain information on the visibility of a point on a building surface. The first step is the preprocessing step for preparing the following main steps. The second step is the step for producing a projection image, and the third step is the step for determining the visibility of each point using the produced projection images. Figure 8 shows the process flow of Method B for obtaining information on the visibility of a point on a building surface.

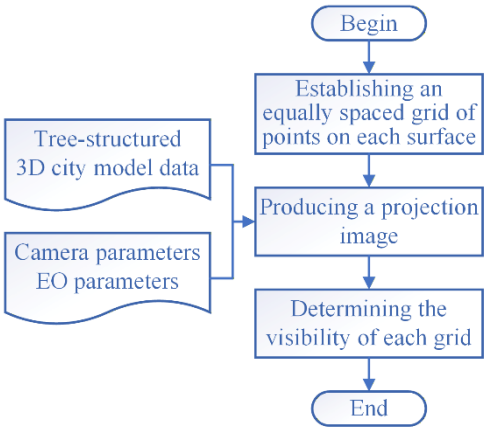


Figure 8. Process flow for obtaining information on the visibility of a point on a building surface

Step I: Preprocessing

1) Establish an equally spaced grid of points on each building surface where the visibility will be determined.
2) Prepare a matrix for storing the information used to select a suitable image for texture mapping and produce a texture image as to each grid. The information stored in the visibility information matrix, which is shown in Figure 9, is as follows:
   - ID of the image to be selected for texture mapping
   - Image coordinates of the selected image to be used in producing a texture image
   - Information for selecting a suitable image

   The information for selecting a suitable image in the visibility information matrix will be discussed later.

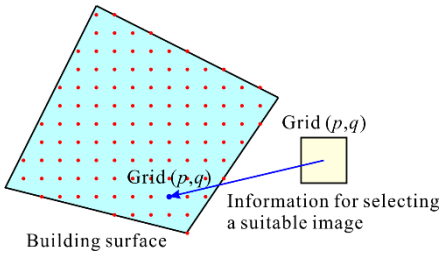


Figure 9. Visibility information matrix

Step II: Producing a projection image

1) Obtain an equation $w = F(c,r)$ for calculating the depth w of a pixel (c,r) on the projection image as to each polygon on building surfaces of a 3D city model.
2) Prepare two matrixes with the same size as the target image for storing intermediate results. One is for storing the polygon ID, and the other is for storing the depth w of each pixel (c,r) of the target image. The former is called an ID matrix, while the latter is called a depth matrix. Both matrices are shown in Figure 10. The ID matrix is initially filled with NULL.

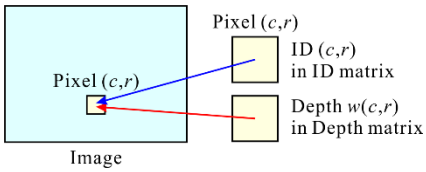


Figure 10. ID matrix and depth matrix

3) Calculate image coordinates (c,r) on the projection image of the vertexes (X,Y,Z) of a polygon of a building surface using the exterior orientation parameters $(X_O, Y_O, Z_O, \omega, \phi, \kappa)$ of the target image.
4) Fill the polygon on the temporal projection image with its ID using the scanline algorithm.
5) Calculate the depth w of each filled pixel (c,r) on the projection image using the equation $w = F(c,r)$ derived at first.
6) Store the polygon ID and the depth w of each pixel (c,r) in the ID matrix and the depth matrix, respectively, if one of the following conditions is satisfied:
   a) The polygon ID stored in the ID matrix is NULL.
   b) The polygon ID stored in the ID matrix is not NULL, and the depth in the depth matrix is larger than the calculated depth.
7) Go to Processing 3), if there remain unprocessed polygons; otherwise, producing a projection image storing the polygon IDs is finished. This is the depth buffer (Z-buffer) algorithm.

Step III: Determining the visibility of each point

Determining the visibility of each point is executed by back projection as follows:

1) Obtain the viewing vector $\vec{V}(u,v,w)$ from the exposure center O to a grid (X,Y,Z) of a polygon of a building surface using the exterior orientation parameters $(X_O, Y_O, Z_O, \omega, \phi, \kappa)$ of the target image.
2) Calculate the incident angle of the viewing vector $\vec{V}$ to the current polygon using the equation of the plane on which a polygon lies. If the calculated incident angle is larger than the predefined threshold, go to Processing 1) as to the next grid for avoiding undesirable results.
3) Calculate the image coordinates (c,r) on the projection image of the current grid.

4) Examine the polygon ID stored in the ID matrix. If the stored polygon ID is the same as the current ID, the current grid of the current polygon is visible; otherwise, the grid is not visible. If the grid is visible, select an image suitable for texture mapping. The detailed algorithm for selecting a suitable image is described in the next section.
5) Execute Processing 1) through 4) as to every grid of every building.

2.3 Selecting a suitable image for texture mapping

If a grid of a polygon of a building surface is photographed by more than one image, we need to choose the most suitable one for the grid. Figure 11 illustrates the current algorithm utilized in the developed software for selecting the appropriate image.

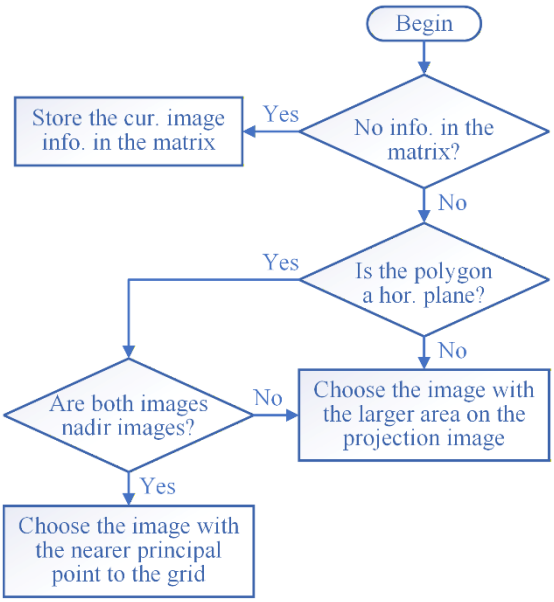


Figure 11. Algorithm for selecting the appropriate image

1) If finding an image photographing the grid, investigate the content stored in the visibility information matrix.
2) If no information is stored in the visibility information matrix, store the information of the current image in the visibility information matrix.
3) If any information is stored in the visibility information matrix, classify the polygon into a nearly horizontal plane or a nearly vertical plane surface according to its normal vector. The current classification criterion is based on the elevation of the normal vector, which is set at 45°.
4) As for a nearly horizontal surface, give priority to a nadir image. If both images are nadir images, select the image whose principal point is nearer to the grid. Consequently, the distance between the grid and the principal point of the selected image is recorded in the visibility information matrix.
5) As for a nearly vertical surface, compare the areas of the polygon on the projected image and select the image with the larger area. Consequently, the area of the polygon on the projected image of the selected image is recorded in the visibility information matrix.

The current information for selecting a suitable image stored in the visibility information matrix is the distance between the grid and the principal point of the selected image and the area of the polygon on the projected image of the selected image.

The image selection for each grid is performed one image at a time to minimize the size of the visibility information matrix.

2.4 Producing a texture image of each polygon

After finishing determining the visibility of grids on the surfaces of all buildings, the software produces a texture image for each polygon. The software produces a texture image by resampling, utilizing the selected image number and the corresponding image coordinates stored in the visibility information matrix.

The production of the texture image of each polygon is performed one aerial image at a time to minimize the time of loading aerial images.

3. Experiment

3.1 Outline of the experiment

We conducted an experiment for investigating the validity of the developed software. In the experiment we used aerial images collected by the Leica CityMapper-2 (CM-2), which is one of the most popular aerial multi-directional camera systems. The specifications of the CM-2 are shown in Table 1 (Leica Geosystems AG, 2025). Figure 12 shows the ground footprints of a nadir camera and four oblique cameras of the CM-2 (Leica Geosystems AG, 2025).

	CM-2 Nadir
Image size	14,192 × 10,640 pixels
Physical pixel size r	3.76 μm
Lens system focal length f	71 mm
Field of view θ across track × along track	41.2° × 31.5°
Flying height H for 10 cm GSD	1890 m
	CM-2 Oblique
Image size	14,192 × 10,640 pixels
Physical pixel size r	3.76 μm
Lens system focal length f	189 mm
Field of view θ image column × image row	16.1° × 12.1°

Table 1. Specifications of the CM-2
(Leica Geosystems AG, 2025)

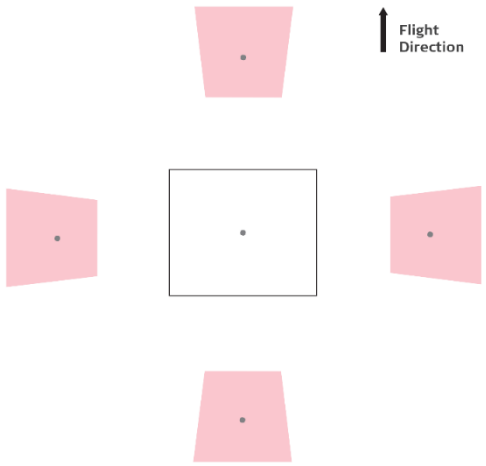


Figure 12. Ground footprint of the CM-2
(Leica Geosystems AG, 2025)

Figure 13 shows the area of interest (AOI) and the projection centers of collected nadir images in Hiroshima City. There are 4,741 LOD2 buildings for texture mapping in the AOI of 3.85 km^2. The images used in the experiment were collected based on the image collection plan of 10 cm GSD, 80% end lap, and 60% side lap. The collection included 606 nadir images and 4 × 606 oblique images. The resolution of a texture image on a building surface is 10 cm × 10 cm in the experiment. Producing texture images for 4,741 buildings took approximately four hours with a CPU core at 3.6 GHz.

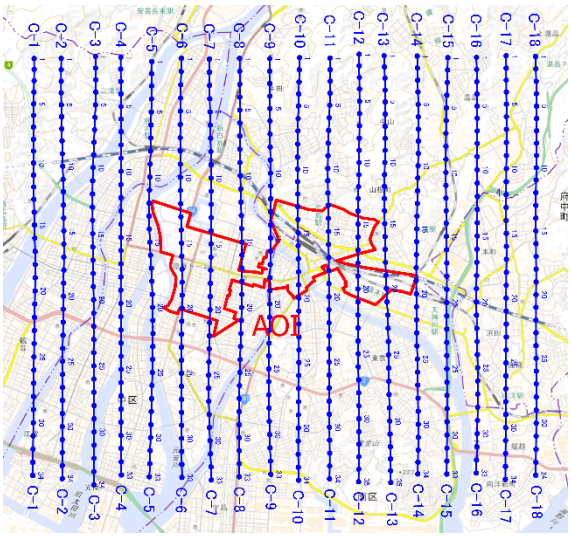


Figure 13. AOI and projection centers of collected nadir images

3.2 Results and discussion

Table 2 shows the image utilization rate and occlusion rate of building roofs and façades in three cases using only nadir images, only oblique images, and both nadir and oblique images. As for the image utilization rate, Ishikawa et al. (2025) suggested that the volume of data collected by a multi-directional camera system for texture mapping might be massive. In the experiment, only about one-third of the collected images were used for texture mapping.

Camera	Image utilization rate	Occlusion rate	
		Roof	Façade
Nadir	41.1%	1.5%	14.4%
Oblique	34.2%	1.8%	22.1%
Total	35.5%	1.1%	12.7%

Table 2. Image utilization rate and occlusion rate

We investigated the produced texture images by human inspection to validate the performance of the developed software in the experiment.

As for roofs, the texture mapping using only nadir images would be able to provide sufficient resolution images, and we considered that the issue of occlusion would be negligible. On the contrary, as for façades, the resolution of texture images produced using oblique images is higher than that using nadir images. However, the effect of occlusion using oblique images is greater than that using nadir images, and the region that is not texture-mapped is larger. In the experiment, we investigated whether using both nadir and oblique images would be able to reduce the area that is not texture-mapped, keeping the higher resolution using oblique images.

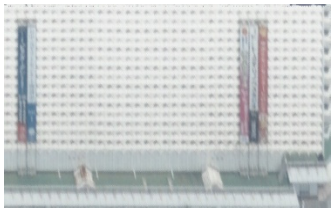
The façades under investigation can be classified into the following types:

a) The whole of a façade is photographed using an oblique image. Texture images of this type can be produced using oblique images.
b) The part of a façade is not photographed by any oblique image, and the whole of a façade is photographed by some nadir or oblique image. Texture images of this type can be produced using nadir and oblique images.
   This type is divided into the following subtypes:
   i) The whole of a texture image would be acceptable.
   ii) The part of a texture image would not be acceptable.
c) The whole of a façade is not photographed by any oblique image.

Most of the façades are combinations of the above basic types.

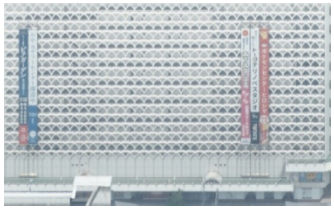
3.2.1 Type-A façade: The entire Type-A façade is photographed by an oblique image. Texture images of this type can be produced using oblique images. The resolution of the texture image of a Type-A façade would be expected to be high.

A Type-A façade shown in Figure 14 indicates that the resolution of a texture image produced using an oblique image is higher than that using a nadir image. (a-1) shows the texture image produced using C05-18 Nadir shown in (a-2), while (b-1) shows the texture image produced using C06-24 Forward shown in (b-2). The resolution of (a-1) is much higher than that of (b-1).



(a-1) Texture image produced using C05-18 Nadir

(a-2) C05-18 Nadir



(b-1) Texture image produced using C06-24 Forward

(b-2) C06-24 Forward

Figure 14. Texture images produced using C05-18 Nadir (a) and C06-24 Forward (c)

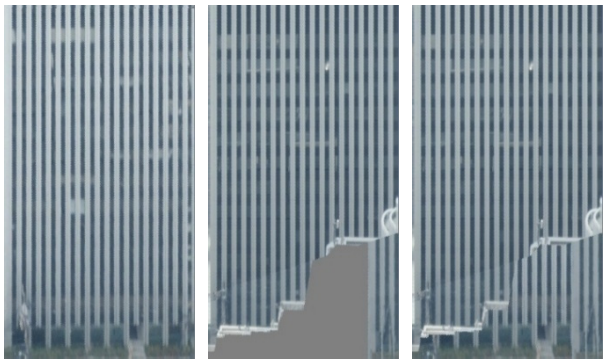
3.2.2 Type-B1 façade: The part of a façade is not photographed by any oblique image, and the whole of a façade is photographed by some nadir or oblique image. Texture images of this type can be produced from nadir and oblique images. The whole texture image of this type would be acceptable. The resolution of the texture image of this type depends on the source image of the texture image. If a source image is an oblique image, the resolution would be expected to be high.

Figure 15 shows a Type-B1 façade. (a), (b), and (c) in Figure 15 show the texture image produced using C08-19 Nadir, five oblique images, and both C08-19 Nadir and five oblique images, respectively.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-4-2026-45-2026 | © Author(s) 2026. CC BY 4.0 License.

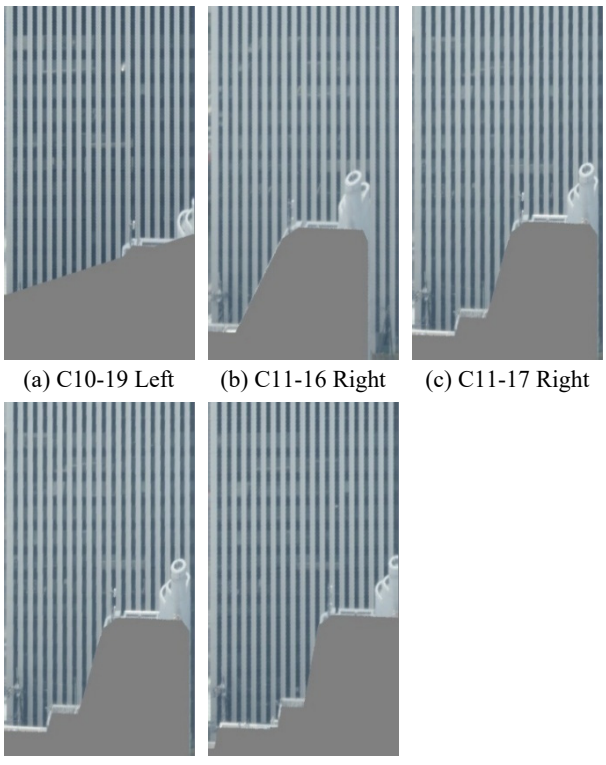
Figure 16 shows the texture images produced using each oblique image: C10-19 Left and C11-16 to C11-19 Right. Figure 17 shows three source images for texture mapping. Since the whole of the façade of interest was not photographed by C10-19 Left, C11-16 to C11-19 Right are used for producing the texture images (b) and (c) shown in Figure 15.

The oblique images, some of which are shown in Figure 17, captured an object on the neighbor's building's roof, which the 3D city model did not represent. As a result, there is an unexpected malfunction in the texture image produced using each oblique image. However, we judged that the quality of the produced texture image would be acceptable. Since some buildings have penthouses, chimneys, and cooling towers on flat rooftops, we consider that texture images produced from aerial images would be unable to prevent such malfunctions.



(a) Nadir (b) Oblique (c) Nadir + Oblique

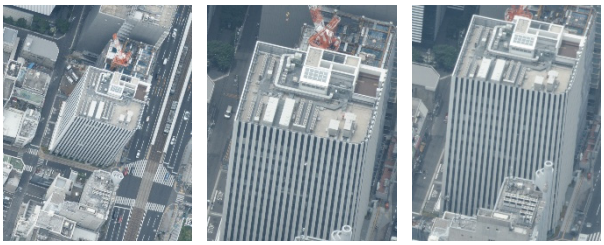
Figure 15. Texture images produced using a nadir image (a), oblique images (b), a nadir and oblique images (c)



(a) C10-19 Left (b) C11-16 Right (c) C11-17 Right

(d) C11-18 Right (e) C11-19 Right

Figure 16. Texture images produced from each oblique image



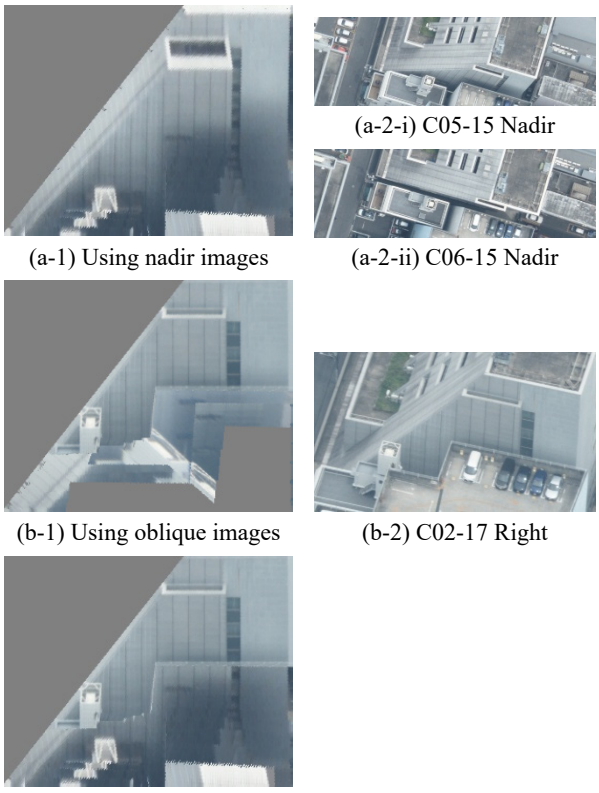
(a) C08-19 Nadir (b) C10-19 Left (c) C11-19 Right

Figure 17. Source images used for texture mapping

3.2.3 Type-B2 façade: The part of a façade is not photographed by any oblique image, and the whole of a façade is photographed by some nadir or oblique image. Texture images of this type can be produced from nadir and oblique images the same as a Type-B1 façade. The difference between a Type-B1 and Type-B2 façade is the resolution of the texture image produced. The part of the texture image for a Type-B2 façade would not have a permissible resolution.

Figure 18 shows a Type-B2 façade. (a-1), (b-1), and (c) in Figure 18 show the texture image produced using nadir images, oblique images, and both nadir and oblique images, respectively.

The texture image (a-1) produced using nadir images has no occlusion. (a-2) in Figure 18 shows two nadir images mainly utilized for producing the texture image (a-1). Although these images are free from occlusion or with slight occlusion, the area of the façade of interest photographed in the collected image is too small to produce a texture image with the permissible resolution.



(a-1) Using nadir images (a-2-i) C05-15 Nadir

(a-2-ii) C06-15 Nadir

(b-1) Using oblique images (b-2) C02-17 Right

(c) Using Nadir & Oblique

Figure 18. Texture images produced using nadir images (a), oblique images (b), nadir and oblique images (c)

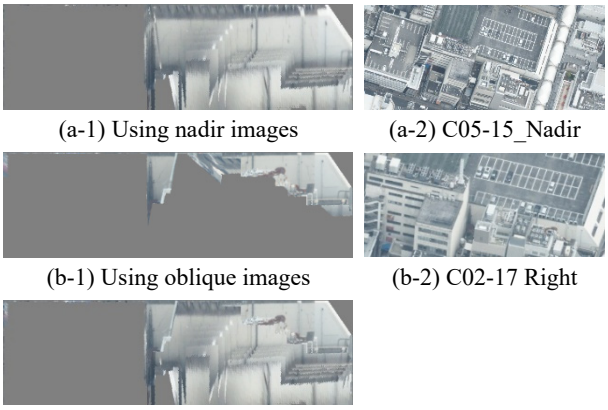
This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-4-2026-45-2026

On the other hand, the texture image (b-1) in Figure 18 produced from oblique images has occlusion. (b-2) in Figure 18 shows an oblique image mainly utilized for producing the texture image (b-1). As a result, the resolution of the part of the texture image (c) produced from nadir and oblique images would not be permissible.

3.2.4 Type-C façade: The whole of a façade is not photographed by any oblique image. If the whole of a façade is not photographed by any nadir image, a texture image of the façade is unable to be produced. In the other case, a texture image of the façade can be produced using some nadir images.

Figure 19 shows a façade that is composed of a Type-B2 and a Type-C façade. (a-1), (b-1), and (c) in Figure 19 show the texture image produced using nadir images, oblique images, and both nadir and oblique images, respectively.

The left hand on the texture image is a Type-C, while the right hand on the texture image is a Type B2. (a-2) shown in Figure 19 is a nadir image utilized for producing the texture image (a-1), while (b-2) is an oblique image utilized for producing the texture image (b-1). These images indicate that the façade of interest is closely approaching the adjacent building, and a texture image with a permissible quality would be unable to be produced.



(a-1) Using nadir images (a-2) C05-15_Nadir

(b-1) Using oblique images (b-2) C02-17 Right

(c) Using Nadir & Oblique

Figure 19. Texture images produced using nadir images (a), oblique images (b), nadir and oblique images (c)

We consider that the software should have the function to reject a texture image with a resolution lower than the predefined threshold. There are two issues in adding the function. One is how to set the threshold value, and the other is how to estimate a texture image's resolution.

4. Conclusion

We developed software that automatically produces texture images. The software uses 3D city model data, aerial nadir and oblique images collected by a multi-directional camera system, and sets of exterior orientation parameters. The software determines the visibility of a point on a building surface using projection images viewed by a camera.

We conducted an experiment for investigating the validity of the developed software. The area of interest in the experiment is 3.85 km^2 and has 4,741 LOD2 buildings for texture mapping. We investigated texture images produced in the experiment by human inspection to validate the performance of the developed software. The experiment results indicate that the developed software would be practical enough.

We discovered that an unrepresented object on the neighbor's building roof causes an unexpected malfunction in the texture image generated from an oblique photograph. Since some buildings have penthouses, chimneys, and cooling towers on flat rooftops that are not represented in the 3D city model, we consider that texture images produced from aerial images would be unable to prevent such malfunctions.

We plan to modify the software so that each pixel of a texture image can have its own resolution. Using the resolution, the updated software will be able to reject a texture image with a resolution lower than the predefined threshold. Moreover, the resolution would be able to assist some analyses, such as investigating stories of a building and detecting doors and windows on a building façade. Additionally, we examine adopting some image processing techniques in resampling to provide a higher resolution for a texture image, such as cubic convolution (Meijering, 2002).

Acknowledgement

The motivation of the study came from the Nverse software of Precision Lightworks, LLC, USA. We would like to thank the people of Precision Lightworks for reading the manuscript and providing valuable comments.

References

Buyukdemircioglu, M. and Oude Elberink, S.,2024. Automated texture mapping CityJSON 3D city models from oblique and nadir aerial imagery, *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, X-4/W5-2024, 87–93, https://doi.org/10.5194/isprs-annals-X-4-W5-2024-87-2024, 2024.

He, H., Yu, J., Cheng, P., Wang, Y., Zhu, Y., Lin, T., Dai, G., 2022. Automatic, Multiview, Coplanar Extraction for CityGML Building Model Texture Mapping, *Remote Sensing*, 14, 50, https://doi.org/10.3390/rs14010050.

Ishikawa, M., Inazawa, T., Nakanishi, Y., Kawamata, F., Takada, M., Danjo, T., and Matsuoka, R., 2025. Resolution of a Texture Image Mapped on a Building Surface of a 3D City Model, *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, X-4/W6-2025, 89–96, https://doi.org/10.5194/isprs-annals-X-4-W6-2025-89-2025.

Leica Geosystems AG, 2025. Leica CityMapper-2, Leica CityMapper-2 DS.pdf, https://leica-geosystems.com/products/airborne-systems/hybrid-sensors/leica-citymapper-2

Meijering, E., 2002. A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing, *Proceedings of the IEEE*, vol. 90, no. 3, 319–342.

Minella, F., Rossi, F. A., Krüger, E., 2011. Analysis of the daytime effect of the sky view factor on the microclimate and on thermal comfort levels in pedestrian streets of Curitiba, *Ambiente Construído*, 11(1), 123-143.

Ministry of Land, Infrastructure, Transport and Tourism of Japan (MLIT), 2025a. https://www.mlit.go.jp/plateau (April 7, 2025).

Ministry of Land, Infrastructure, Transport and Tourism of Japan (MLIT), 2025b. Standard Data Product Specification for 3D City Model, https://www.mlit.go.jp/plateau/file/libraries/doc/plateau_doc_0001_ver04.pdf.