Automatically drawing vegetation maps using digital time-lapse cameras in alpine ecosystems

A Preprint

Ryotaro Okamoto [*]
Doctoral Program in Biology
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8577 Japan.
okamoto.ryotaro.su@alumni.tsukuba.ac.jp

Hiroyuki Oguma
Biodiversity Division
National Institute for Environmental Studies
16-2 Onogawa, Tsukuba, Ibaraki 305-8506 JAPAN
oguma@nies.go.jp

Reiko Ide
Earth System Division
National Institute for Environmental Studies
16-2 Onogawa, Tsukuba, Ibaraki 305-8506 JAPAN
ide.reiko@nies.go.jp

July 26, 2022

Abstract

Enter the text of your abstract here.

***K*eywords** alpine ecology · deep learning · ecosystem monitoring

1 Introduction

The effects of climate change on terrestrial ecosystems are particularly significant in alpine regions ([1]). Alpine vegetation depends on severe conditions such as low temperatures and long snow-covered periods. Thus alpine areas have rare and unique species adapted to the extreme environments. Several studies have reported that recent global climatic changes, e.g., increasing temperatures and reducing snow-covered periods, have accelerated the invasion of non-native species into alpine areas (see [2]). In Japan, dwarf bamboo (*Sasa kurilensis*) has invaded alpine snow meadows, probably driven by the extension of the snow-free period ([3]). Also, climate change has affected the growth and phenologies of native species. For example, the growth of dwarf pine (*Pinus pumila*), a dominant species in Japanese alpine regions, has been affected by climatic conditions such as temperature and snowmelt ([4]). Effective conservation planning requires early detection and constant monitoring of such vegetation changes. Also, since the impact of climate change on alpine vegetation varies depending on species and the microhabitats ([5]), spatially high-resolution monitoring with a wide range is required.

Previous studies have mainly depended on field observations, yet it is hard to cover broad areas in alpine regions due to the poor accessibility and severe weather. Satellite, airborne, and Unmanned Aerial Vehicle (UAV) remote sensing methods seem to be alternatives. However, satellite imagery of alpine areas is rarely available due to cloud cover, and the spatial resolution is not enough to observe vegetation changes at the plant community scale. Airborne imagery can obtain high-resolution data, but its cost becomes a bottleneck for frequent monitoring. Although UAV methods have become a cost-effective tool for ecological monitoring ([6]), operating UAVs in alpine regions is challenging due to the strong wind and harsh topology.

On the other hand, researchers have also utilized automated digital time-lapse cameras mounted on the ground for monitoring green-leaf phenologies in forests ([7]), grasslands ([8]), and alpine meadows ([9]). Unlike

[*]https://github.com/0kam

satellite imagery, such cameras provide images free of clouds and atmospheric effects. Also, they can obtain high-resolution (i.e., sub-meter scale) and frequent (i.e., daily or hourly) images at a meager cost. These studies set some regions of interest (ROI) in the images and calculate the phenology index (e.g., excess greenness, [10]). However, few studies have utilized such images in monitoring vegetation distributions. This lack of studies seems to be because applying such repeat photography in monitoring vegetation distribution has two technical challenges.

First, unlike satellites' multispectral sensors, ordinal digital cameras can only obtain three bands (Red, Green, and Blue), making it harder to classify the vegetation. Second, since digital time-lapse cameras are mounted on the ground, transforming these images into geospatial data (e.g., orthoimage) is challenging. In other words, even if we classify the vegetation from such images, we cannot quantitatively measure and analyze it as geographic data. It is essential to treat ground-based images as geographic data to utilize them in conservation planning.

This study proposes an automated method for drawing vegetation maps with a digital time-lapse camera by solving these two challenges. We solved the first challenge by using time-lapse images to classify vegetation. Since the autumn leaf phenology varies among species, we utilized that information. We show the effectiveness of such phenological information in vegetation classification. We also developed a novel method for transforming a ground-based photograph into geographic data to tackle the second challenge. Finally, we show an example of a vegetation map drawn by the proposed method. We aim to use cheap but powerful digital time-lapse cameras in alpine ecosystem monitoring and conservation.

2 Materials and methods

2.1 Digital time-lapse camera imagery

We used repeat photography data owned by National Institute for Environmental Studies, Japan (NIES). All the images are publically available on NIES' webpage (`https://db.cger.nies.go.jp/gem/ja/mountain/station.html?id=2`). In 2010, NIES installed the digital time-lapse camera (EOS 5D MK2, Canon Inc., 21 M pixels) on a mountain lodge Murodo-sanso (about 2350 m a.s.l., above the forest limit), located at the foot of Mt. Tateyama (3015 m a.s.l.), in the Nothern Japanese Alps 1. The camera takes a photograph per hour, from 6 a.m. to 7 p.m. . The camera's field of view (FOV) includes Mt. Tateyama, which ranges from about 2350 m a.s.l. to 3015 m a.s.l. in elevation. The area has a complex mosaic-like vegetation structure because of its topography, including rocks, cliffs, curls, and moraines. From April to November, the camera has observed the snowmelt and seasonal phenology of evergreen(e.g., *Pinus pumila*) and deciduous (*Sorbus* sp., *Betula ermanii*) dwarf trees, dwarf bamboos (e.g., *Sasa kurilensis*), and alpine shrubs and herbaceous plants (e.g., *Geum pentapetalum*, *Nephrophyllidium crista-galli*). This study used images from the summer and fall of 2015 to classify vegetation with the temporal patterns of the leaf color.

2.2 Preprocessing

2.2.1 Selecting images

First, we selected images that are suitable for vegetation classification. We choose seven days with good weather from late summer to late fall of 2015 (8/25, 9/5, 9/12, 9/20, 9/26, 10/3, 10/10). We used the images of this season because we can separate vegetation from the patterns of autumn foliage coloration.

2.2.2 Automatic image-to-image alignment

Since images are slightly misaligned, we aligned them before processing. We implemented the program with the Python3 programming language and OpenCV4 (`https://opencv.org/`) image processing library. First, we set one image of 2015 as the alignment target. Next, we automatically found matching keypoints between the target and other images using the AKAZE local feature extractor ([11]) and K-nearest neighbor matcher. Then, we searched and applied the homography matrix that minimizes the distance between each pair of the matching points, using OpenCV's "findHomography" function. Applying the estimated homography matrix, we could align the images accurately (0.654 pixels in root mean square error of the matching points). Finally, we prepared an input mask to define image regions that should be ignored in the following procedure, such as the sky and the regions too close to the camera.

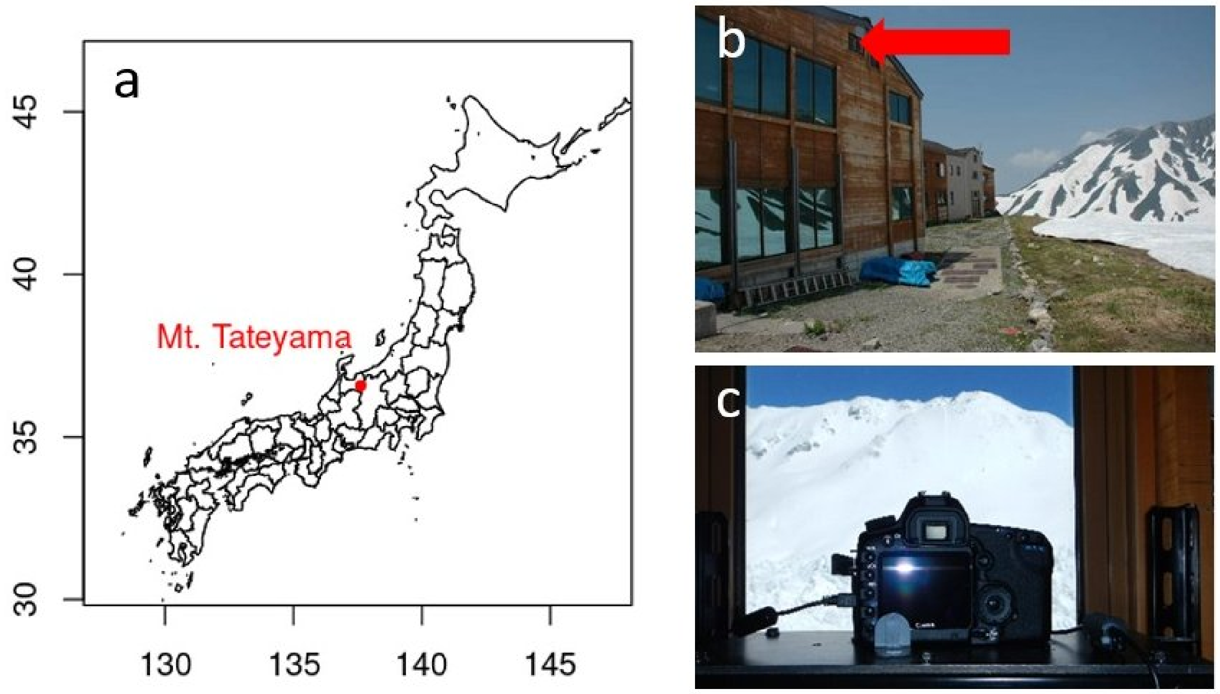

Figure 1: The study site. (a) The location of Mt. Tateyama. (b) The installation point of the camera in Murodo-sanso lodge. (c) The camera (EOS 5D MK2, Canon Inc.)

2.3 Automatic vegetation classification

Because we successfully aligned the images, we can stack the images and extract a time series of pixel values (Red, Green, and Blue, 2) for each pixel. Such pixel time series reflects the temporal patterns of leaf colors. Because deciduous plants have great differences in autumn phenology among species, researchers have used that information for vegetation classification with satellite imagery (e.g., [12], [13], [14]). However, no research has applied this technique to ground-based repeat photography imagery. We implemented Support Vector Machine (SVM) and Recurrent Neural Network (RNN) based vegetation classification methods in this study. We tested the effects of using pixel time series on the classification performance.

2.3.1 Model Architecture

We prepared two supervised models, SVM and RNN, to classify the pixel time series into vegetation categories. SVM is one of the most popular machine learning models in remote sensing and has many applications ([15]), including vegetation classification with multitemporal satellite imagery ([12]). RNN is a neural network that recognizes temporal or sequential data dynamics. Researchers have also utilized RNN for remote sensing tasks, such as land cover classification, with multi-temporal satellite imagery ([**?**], [16]). Amongst many variants of RNN, we used Long Short Time Memory (LSTM, [17]), one of the most well-known RNN architectures. We also classified the pixel of every single image separately using SVM classifiers to test whether using multi-temporal imagery increases the classification performance.

2.3.2 Dataset preparation

We set 7 vegetation classes: Dwarf Pine (*Pinus pumila*), Dwarf Bamboo (*Sasa kurilensis*), Rowans (*Sorbus sambucifolia*, *Sorbus matsumurana*), Birch (*Betula ermanii*), Alder (*Alnus viridis* subsp. *maximowiczii*), Other Vegetations (such as alpine shrubs and herbaceous plants), and No Vegetation. Using an open-source image annotation software (Semantic Segmentation Editor, Hitachi, `https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor`), an expert prepared a teacher dataset for each class. Then we validated the teacher dataset with a set of telephotos taken in the summer and fall of 2016.

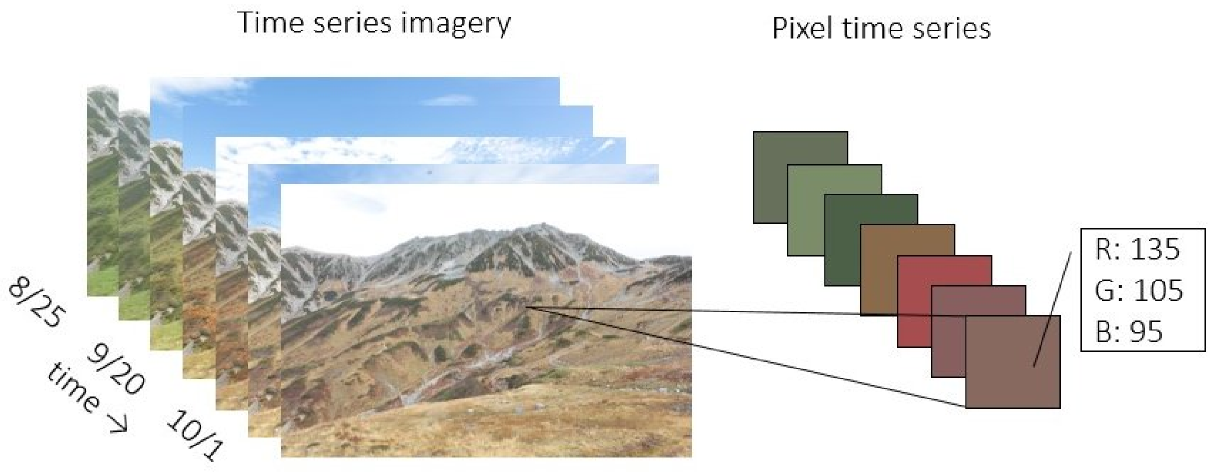

Figure 2: Pixel time series acquired from the time lapse camera. You can obtain a time series of pixel values (i.e., Red, Green, and Blue) for each pixel on the photographs. Such pixel time series reflects the autumn phenology of the vegetation.

2.3.3 Implementation and model training

We implemented the classifier with Python3 language, ThunderSVM library (`https://github.com/Xtra-Computing/thundersvm`) for the SVM classifier, and PyTorch deep neural network library (`https://pytorch.org/`) for the RNN classifier. All source codes are publically available via GitHub (`https://github.com/0kam/xxxx`).

2.4 Automatic georectification

Then, we developed a novel method to convert ground-based landscape imagery into GIS-ready geographical data. This process is called georectification. Georectification of ground-based images has been a difficult task, and this causes the underuse of potentially rich information in ground-based imagery. In plain words, georectification means aligning images onto Digital Surface Models (DSMs) so that every pixel of a image gets a geographical coordinate. You can do this by modeling the camera. A camera is considered as a function that transforms 3D geographical coordinates (e.g., X, Y and height in a Universal Transverse Mercator coordinate system (UTM)) into 2D image coordinates (locations of each pixel in the image). Estimating the parameters of this function (such as the camera location, pose, and the FoV), you can simulate how each point of the DSM appears in the image. Usually, georectification has three steps:

1. Finding Ground Control Points (GCPs) in the image.
2. Estimating the camera parameters such as camera poses and field of view using GCPs.
3. Projecting the DSM into the image using the camera parameters.

Recently, researchers have developed some georectification methods to use ground-based photographs in glaciology ([18]) and snow cover studies ([19]). Especially, [19] is worth mentioning for its semi-automatic method using mountain silhouettes as GCPs. However, this silhouette-based method has a drawback in the projection accuracy. It only uses limited areas (silhouettes) of images in the image-to-DSM alignment, and also it ignores lens distortion. Because our target site has a complex vegetation distribution and our camera has considerable lens distortion, we needed a more accurate method.

2.5 Modeling and estimating camera parameters

As we mentioned before, we considered a camera as a function that transforms geographical coordinates of the target mountain to the image coordinates in the acquired image. We implemented this procedure using the OpenGL framework to accelerate it with graphical processing units (GPUs). In OpenGL, we can separate this process into three operations:

1. Transforming the world (geographical) coordinates to the view coordinates (coordinates seen from the camera's point of view) using camera's extrinsic parameters (i.e., the location and angles of the camera).

2. Distorting the view coordinates using the lens distortion parameters.

3. Transforming the view coordinates to the screen (image) coordinates using camera's intrinsic parameters (i.e., the FoV and aspect ratio of the camera).

First, we transformed the absolute geographic coordinates (also known as the world coordinates) to the view coordinates that are relative to the camera's position and direction. We applied a 4×4 view matrix M_{view} (2.5) to the geographic coordinates in this step. The view matrix represents the position and the direction (pan, tilt, roll) of the camera. Note that the geographic coordinate system must be cartesian (such as the UTMs).

$$M_{view} = \begin{bmatrix} \cos roll & -\sin roll & 0 & 0 \\ \sin roll & \cos roll & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos tilt & -\sin tilt & 0 \\ 0 & \sin tilt & \cos tilt & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos pan & 0 & \sin pan & 0 \\ 0 & 1 & 0 & 0 \\ -\sin pan & 0 & \cos pan & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -z \\ 0 & 0 & 1 & -y \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Where $pan, tilt, roll$ are the Euler angles of the camera pose and x, y, z are the camera location in the geographic coordinate system. Then, we can transform the geographic coordinates of the DSM $[X_{geo} \quad Z_{geo} \quad Y_{geo} \quad 1]$ to the view coordinates $[X_{view} \quad Z_{view} \quad Y_{view} \quad 1]$ applying the view matrix M_{view} (2.5). In the OpenGL's view coordinate system, X_{view}, Z_{view} and Y_{view} represents horizontal, vertical, and depth positions respectively.

$$\begin{bmatrix} X_{view} \\ Z_{view} \\ Y_{view} \\ 1 \end{bmatrix} = M_{view} \begin{bmatrix} X_{geo} \\ Z_{geo} \\ Y_{geo} \\ 1 \end{bmatrix} \tag{2}$$

Second, we distorted the camera coordinates to simulate the lens distortion. We modeled the lens distortion (2.5) based on [20] and OpenCV's implementation (`https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html`), where X_{norm} and Z_{norm} are the Y-normalized view coordinates. This model distorts points' locations according to the distance from the center of the image when projected to the image surface. Our model includes radial ($k1$~$k6$), tangental ($p1$, $p2$), thin prism ($s1$~$s4$) distortion, and unequal pixel aspect ratio ($a1$, $a2$). See [20] for the details of lens distortion modeling.

$$X_{norm} = \frac{X_{camera}}{Y_{camera}} \tag{3}$$

$$Z_{norm} = \frac{Z_{camera}}{Y_{camera}} \tag{4}$$

$$r^2 = {X_{norm}}^2 + {Z_{norm}}^2 \tag{5}$$

$$\begin{bmatrix} X_{dist\_norm} \\ Z_{dist\_norm} \end{bmatrix} = \begin{bmatrix} X_{norm} \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1x'y' + p_2(r^2 + 2x'^2) + s_1r^2 + s_2r^4 \\ Z_{norm} \frac{1+a_1+k_1r^2+k_2r^4+k_3r^6}{1+a_2+k_4r^2+k_5r^4+k_6r^6} + p_1(r^2 + 2y'^2) + 2p_2x'y' + s_3r^2 + s_4r^4 \end{bmatrix} \tag{6}$$

$$X_{dist} = X_{dist\_norm} Y_{camera} \tag{7}$$

$$Z_{dist} = Z_{dist\_norm} Y_{camera} \tag{8}$$

$$\tag{9}$$

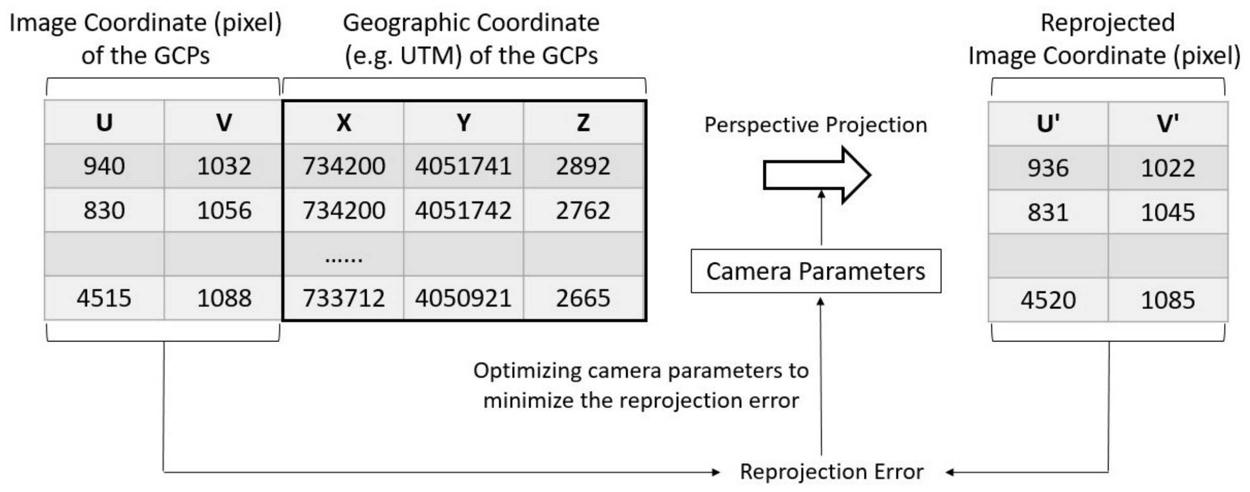
Image Coordinate (pixel) of the GCPs
Geographic Coordinate (e.g. UTM) of the GCPs
Perspective Projection
Camera Parameters
Reprojected Image Coordinate (pixel)
Optimizing camera parameters to minimize the reprojection error
Reprojection Error

U	V	X	Y	Z
940	1032	734200	4051741	2892
830	1056	734200	4051742	2762
		......		
4515	1088	733712	4050921	2665

U'	V'
936	1022
831	1045
4520	1085

Figure 3: Work flow of the camera parameter optimization. We estimated the camera parameters by minimizing the GCP's reprojection error.

Next, we transformed the distorted view coordinates to the image coordinates by perspective projection. During the perspective projection, a closer object is drawn larger. We used a 4×4 projection matrix M_{proj} (2.5), that represents the camera's horizontal (fov_x) and vertical (fov_z) FoV. Finally, we can get the image coordinates of the points as X_{image}, Z_{image} (2.5).

$$f_x = \frac{1}{\frac{\tan fov_x}{2}} \tag{10}$$

$$f_z = \frac{1}{\frac{\tan fov_z}{2}} \tag{11}$$

$$M_{proj} = \begin{bmatrix} f_x & 0 & 1 & 0 \\ 0 & f_z & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} X_{image} \\ Z_{image} \\ Y_{image} \\ 1 \end{bmatrix} = M_{proj} \begin{bmatrix} X_{dist} \\ Z_{dist} \\ Y_{dist} \\ 1 \end{bmatrix} \tag{13}$$

Applying these procedures to the GCPs' geographical coordinates, we can calculate the reprojected image coordinates with a set of camera parameters (the camera location, pose, FoV, and the lens distortions). Then, we can major the distance from the actual image coordinates of GCPs (U, V of 3) to the reprojected image coordinates (U', V' of 3). We optimized the camera parameters (except camera location) by minimizing this distance using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES, [21], 3). We could not estimate the camera location because it makes the problem too complicated (e.g., a telephoto taken from a distance and a wide-angle taken from a close look similar).

2.6 Image-matching-based aqcuisition of GCPs

To get matching points between images and DSMs on a broader area, we additionally used an orthorectified airborne image. Applying the camera model to an airborne image, a DSM, and a set of initial camera parameters, we rendered a simulated landscape image 4. Then we got matching points of an original image and the simulated image using AKAZE local feature matcher 4. Since these matching points have both geographical coordinates (from the DSM) and image coordinates (from the original image), we used these

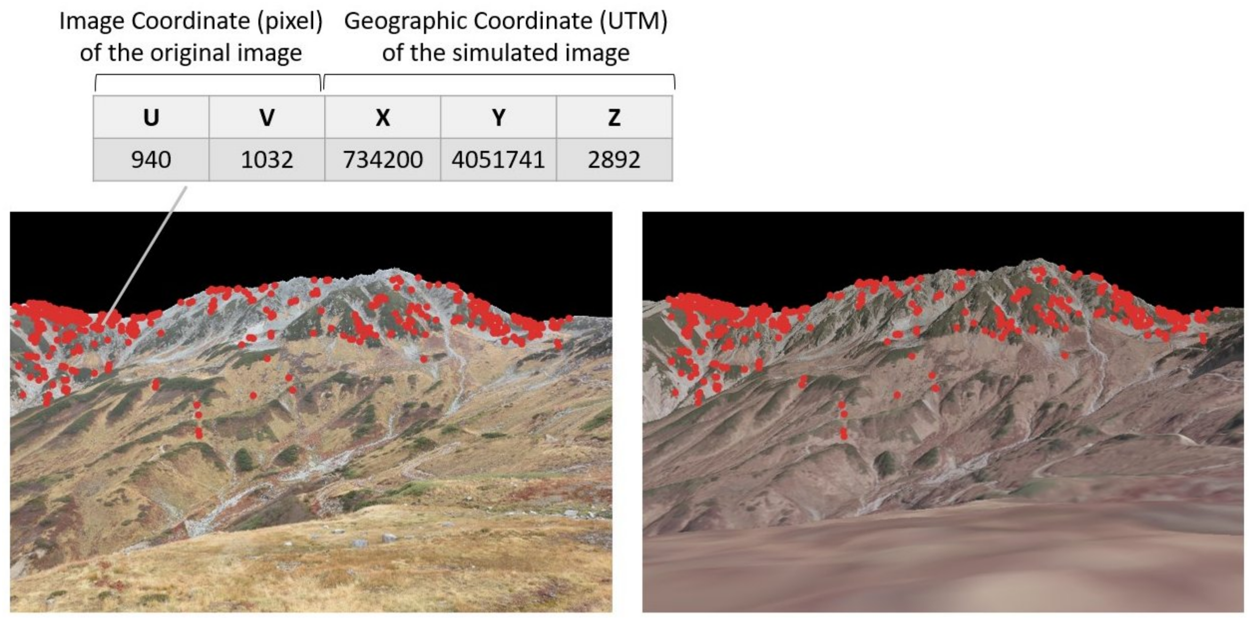
Image Coordinate (pixel) of the original image
Geographic Coordinate (UTM) of the simulated image

U	V	X	Y	Z
940	1032	734200	4051741	2892

Figure 4: The original image (left) and the simulated image (right). The simulated image was rendered with an orthophoto, DSM, and initial camera parameters. Red points shows the matching points found by the AKAZE local feature matcher. We used these points as GCPs.

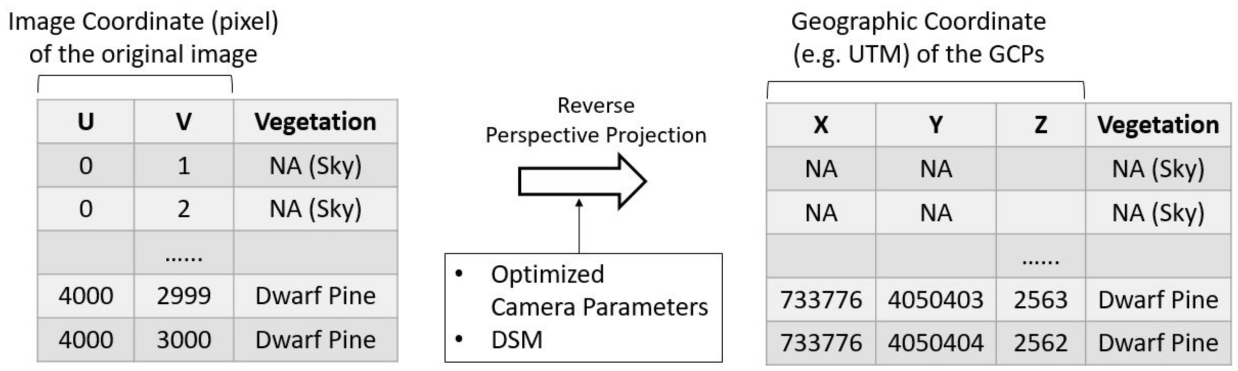
Image Coordinate (pixel) of the original image
Reverse Perspective Projection
• Optimized Camera Parameters
• DSM
Geographic Coordinate (e.g. UTM) of the GCPs

U	V	Vegetation
0	1	NA (Sky)
0	2	NA (Sky)
	......	
4000	2999	Dwarf Pine
4000	3000	Dwarf Pine

X	Y	Z	Vegetation
NA	NA		NA (Sky)
NA	NA		NA (Sky)
		......	
733776	4050403	2563	Dwarf Pine
733776	4050404	2562	Dwarf Pine

Figure 5: Our georectification procedure. We applied the optimized camera parameters to the DSM and the vegetation classification result to get a vegetation map.

points as GCPs. Our procedure requires the camera's exact location and initial camera parameters to render the simulated image. Note that the orthorectification accuracy of the airborne photograph may affect the georectification accuracy.

2.7 Georectification of the vegetation map

Finally, we georectified the vegetation classification result using the optimized camera parameters 5. At this point, we got point data of vegetation (as shown in the table on the right of 5) that every row represents a pixel of the original image. To measure the area of each vegetation class, we converted the point data to a raster data. We rasterized the point data in 1 m resolution, then interpolated holes up to 2 m since the maximum pixel foot print was about 2 m on the ridge of the mountain. We implemented these procedure using R language [22], `stars` package [23] and `terra` [24] package. See `https://github.com/0kam/xxx/xxx` for the source code.

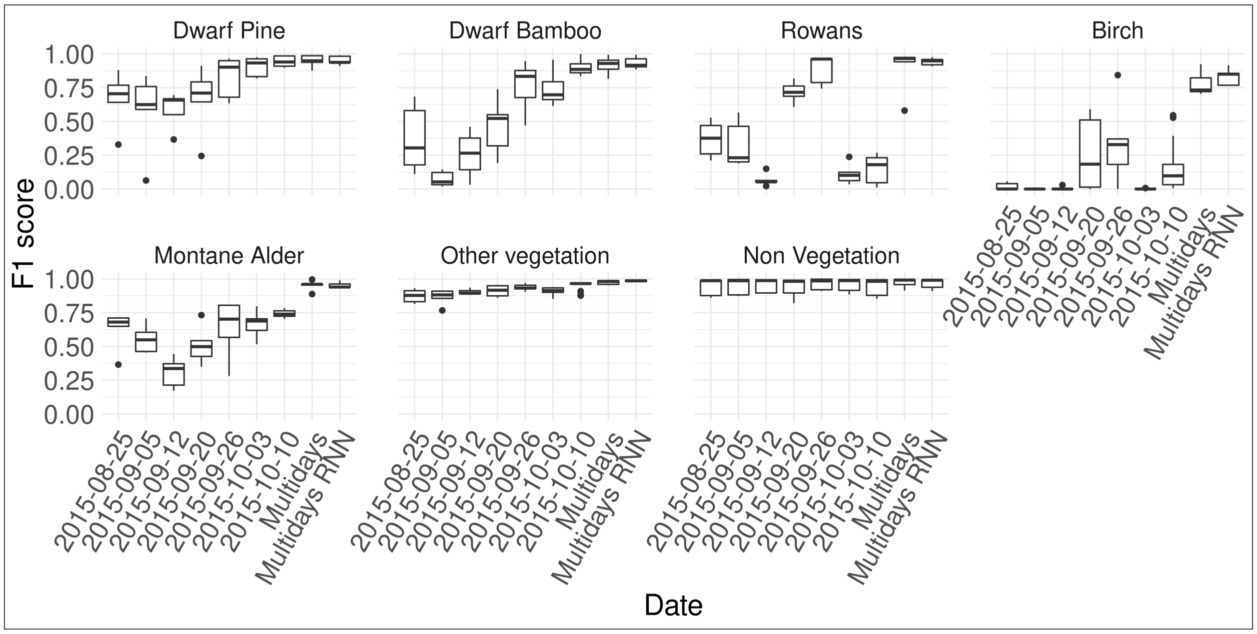

Figure 6: F1-scores of vegetation classification. F1-score was calculated for each vegetation class. Each box shows the results of 5-fold cross validation. We classified vegetation using single image (shown as the shooting date) with SVM classifiers, and all images with SVM (shown as "Multidays") and RNN (shown as "Multidays RNN") classifiers.

2.8 Implementation and data set

We implemented the algorithm with Python3 language and published it as an open-source package via GitHub (`https://github.com/0kam/alproj`). You can try it with your data. We used an airborne photograph taken in the November of 201X with a spatial resolution of 1.0 m. Also, we used the 1m resolution Digital Surface Model that were also used in the orthorectification process of the airborne photograph.

3 Results

3.1 Vegetation classification accuracy

We evaluated the performance of the vegetation classifier using a 5-fold cross-validation design. Each fold was stratified with the vegetation categories. We used F1-score, a standard metrics in machine learning evaluation (3.1), where TP, FP, FN is true positives, false positives, and false negatives, respectively.

$$precision = \frac{TP}{TP + FP} \tag{14}$$

$$recall = \frac{TP}{TP + FN} \tag{15}$$

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{16}$$

$$\tag{17}$$

6 shows the results of the 5-fold cross validation. Focusing on the single image classification (shown as the shooting date), the best date for classification was different among classes. For example, 26th Sep was the best for identifying rowans, but 10th Oct was for dwarf pines and dwarf bamboos. Note that the autumn colorization of rowan leaves was at the best on 26th Sep, and the most vegetation except dwarf bamboo and dwarf pine were blasted at 10th Oct. No single image can classify birch and montane alder accurately.

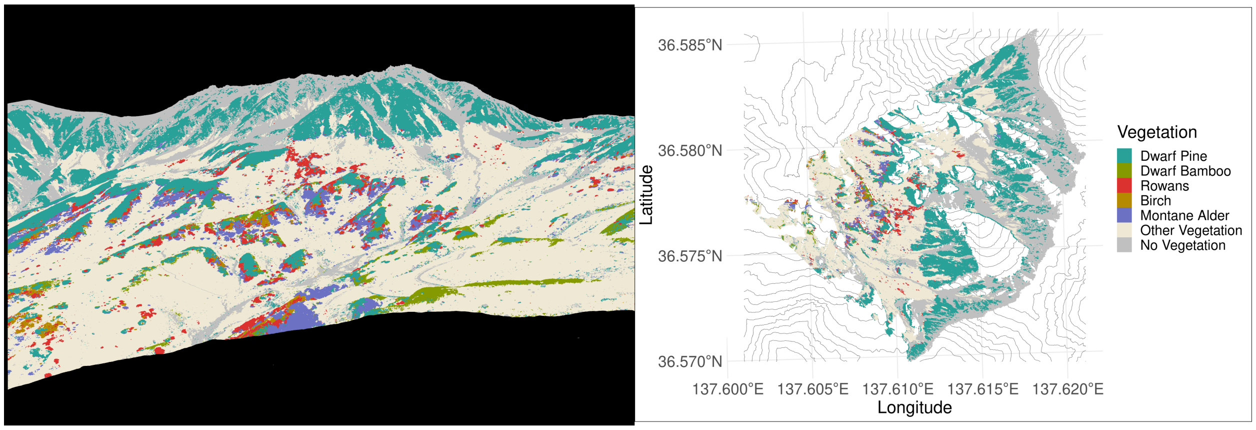

Figure 7: Vegetation classification results of the RNN model. The masked area, i.e., the sky and the regions too close to the camera, is shown in black.

In contrast, using all images (shown as "Multidays", "Multidays RNN", representing the SVM and RNN classifiers respectively) resulted high F1 score in every classes. Also, the RNN classifier outperforms the SVM classifier (0.937 and 0.918 in mean macro average F1 score). 7 shows the products of the RNN classifier. We can observe the distribution of dwarf pine and dwarf bamboo at the plant-community scale.

3.2 Georectification accuracy

We tested the accuracy of our georectification method and evaluated the effect of its two features: using image matching to acquire GCPs, and modeling the lens distortion. We manually searched 83 matching points between the simulated photograph and the original photograph and treated them as the test GCPs: GCPs that are not used in the georectification process. We additionally prepared two comparison methods to test the performance of our proposed method. The first one is the silhouette based matching method (hereinafter called silhouette) used in previous studies ([19]). The second one is the proposed method without the lens distortion model (hereinafter called no distortion). We evaluated the projection error of the test GCPs using these three methods. The proposed method achieves accurate projection (3.45 m as the root mean square error) while the other two did not (16.1 m with silhouette, 23.6 m with no distortion). Note that we estimated the lens distortion parameters in the silhouette method. 8 shows the relationship between the projection error of the proposed method and the test GCPs' distance from the shooting point. The projection error was especially large in the GCPs near to the shooting point.

3.3 Vegetation maps

4 Discussion

We suggested a fully automated procedure to transform time-lapse imagery into georeferenced vegetation maps at a meager cost. This task is challenging because of 1. the difficulty of classifying vegetation with ordinal digital camera imagery and 2. the difficulty of georectification. We solved these issues by 1. using the temporal information of autumn leaf colors for vegetation classification and 2. developing a novel method for accurate image georectification. Our vegetation classification performance (with an accuracy of 0.87) and georectification methods (with an RMSE of $rmse$ m) are practical.

4.1 The benefit of using time-series imagery for vegetation classification

One of the shortcomings of ordinal digital cameras is that they only have three bands (Red, Blue, and Green). We made up for this lack of information by using the rich temporal information that time-lapse cameras can obtain. Many plant species have characteristic phenologies, such as flowering and autumn foliages. Observing this requires long-term (such as year-round) monitoring with a high frequency, and digital time-lapse cameras are suitable. This study only focused on the two species (dwarf bamboo and stone pine) that are easy to classify, even with a single image. However, like the previous studies ([12], [13], [14]), we expect our method

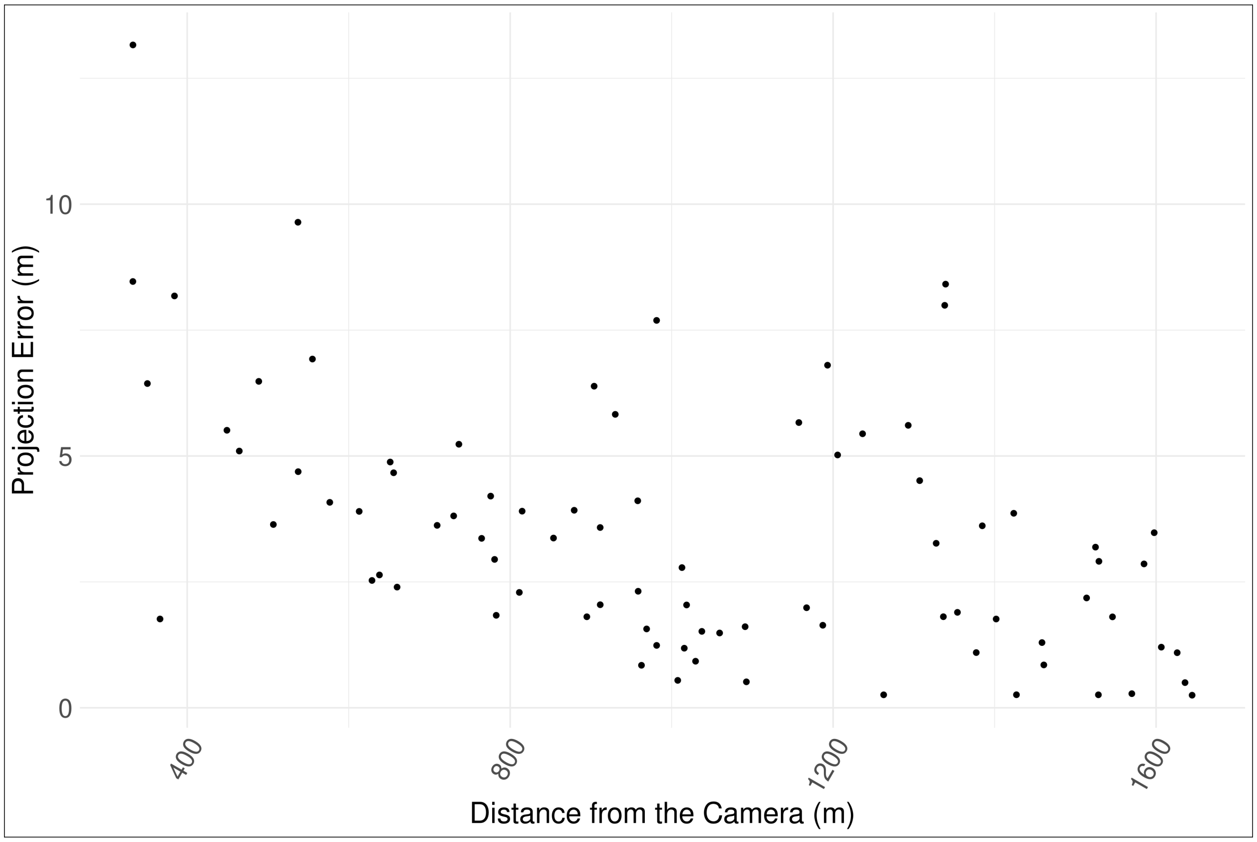

Figure 8: Vegetation classification results of the RNN model. The masked area, i.e., the sky and the regions too close to the camera, is shown in black.

to classify more vegetation (e.g., dwarf deciduous trees and alpine herbaceous plants) using the temporal information of leaf colors. We also used hourly images for each day. Fig. x shows that this additional information made our model robust against transient noises, such as shadows. Since alpine ecosystems have rough topologies, shadows are a considerable problem. Time-lapse cameras are also beneficial for this reason.

4.2 The performance of the georectification method and its limitation.

4.3 Future application and conclusion

References

[1] Intergovernmental Panel on Climate Change (IPCC). *Climate Change 2007: ThePhysical Science Basis: Contribution of Working Group I to the Fourth AssessmentReport of the IPCC.* Cambridge University Press, 2007.

[2] Jake M. Alexander, Jonas J. Lembrechts, Lohengrin A. Cavieres, Curtis Daehler, Sylvia Haider, Christoph Kueffer, Gang Liu, Keith McDougall, Ann Milbau, Aníbal Pauchard, Lisa J. Rew, and Tim Seipel. Plant invasions into mountains and alpine ecosystems: current status and future challenges. *Alpine Botany*, 126:89–103, 10 2016.

[3] Gaku Kudo, Yukihiro Amagai, Buho Hoshino, and Masami Kaneko. Invasion of dwarf bamboo into alpine snow-meadows in northern japan: pattern of expansion and impact on species diversity. *Ecology and Evolution*, 1:85–96, 9 2011.

[4] Yukihiro Amagai, Masami Kaneko, and Gaku Kudo. Habitat-specific responses of shoot growth and distribution of alpine dwarf-pine (pinus pumila) to climate variation. *Ecological Research*, 30:969–977, 11 2015.

[5] Gaku Kudo, Mitsuhiro Kimura, Tetsuya Kasagi, Yuka Kawai, and Akira S. Hirao. Habitat-specific responses of alpine plants to climatic amelioration: Comparison of fellfield to snowbed communities. *Arctic, Antarctic, and Alpine Research*, 42:438–448, 11 2010.

[6] Susana Baena, Justin Moat, Oliver Whaley, and Doreen S. Boyd. Identifying species from the air: Uavs and the very high resolution challenge for plant conservation. *PLOS ONE*, 12:e0188714, 11 2017.

[7] Andrew D. Richardson, Bobby H. Braswell, David Y. Hollinger, Julian P. Jenkins, and Scott V. Ollinger. Near-surface remote sensing of spatial and temporal variation in canopy phenology. *Ecological Applications*, 19:1417–1428, 9 2009.

[8] Dawn M. Browning, Jason W. Karl, David Morin, Andrew D. Richardson, and Craig E. Tweedie. Phenocams bridge the gap between field and satellite observations in an arid grassland ecosystem. *Remote Sensing 2017, Vol. 9, Page 1071*, 9:1071, 10 2017.

[9] Reiko Ide and Hiroyuki Oguma. A cost-effective monitoring method using digital time-lapse cameras for detecting temporal and spatial variations of snowmelt and vegetation phenology in alpine ecosystems. *Ecological Informatics*, 16:25–34, 7 2013.

[10] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen. Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the ASAE*, 38:259–269, 1 1995.

[11] Pablo F Alcantarilla, Jesús Nuevo, and Adrien Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *Proceedings of the British Machine Vision Conference*, 2013.

[12] Jan Tigges, Tobia Lakes, and Patrick Hostert. Urban vegetation classification: Benefits of multitemporal rapideye satellite data. *Remote Sensing of Environment*, 136:66–75, 9 2013.

[13] Nguyen Thanh Son, Chi Farn Chen, Cheng Ru Chen, Huynh Ngoc Duc, and Ly Yu Chang. A phenology-based classification of time-series modis data for rice crop monitoring in mekong delta, vietnam. *Remote Sensing 2014, Vol. 6, Pages 135-156*, 6:135–156, 12 2013.

[14] Katharina Heupel, Daniel Spengler, and Sibylle Itzerott. A progressive crop-type classification using multitemporal remote sensing data and phenological information. *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 86:53–69, 4 2018.

[15] Giorgos Mountrakis, Jungho Im, and Caesar Ogole. Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66:247–259, 5 2011.

[16] Atharva Sharma, Xiuwen Liu, and Xiaojun Yang. Land cover classification from multi-temporal, multi-spectral remotely sensed imagery using patch-based recurrent neural networks. *Neural Networks*, 105:346–355, 9 2018.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[18] A. Messerli and A. Grinsted. Image georectification and feature tracking toolbox: Imgraft. *Geoscientific Instrumentation, Methods and Data Systems*, 4:23–34, 2 2015.

[19] Celine Portenier, Fabia Hüsler, Stefan Härer, and Stefan Wunderle. Towards a webcam-based snow cover monitoring network: Methodology and evaluation. *Cryosphere*, 14:1409–1423, 4 2020.

[20] Juyang Weng, Paul Cohen, and Marc Herniou. I i ieee transactions camera calibration with distortion models and accuracy evaluation. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 14, 1992.

[21] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11:1–18, 3 2003.

[22] R Core Team. R: A language and environment for statistical computing, 2022.

[23] Edzer Pebesma. stars: Spatiotemporal arrays, raster and vector data cubes, 2022. https://r-spatial.github.io/stars/, https://github.com/r-spatial/stars/.

[24] Robert J Hijmans. terra: Spatial data analysis, 2022. https://rspatial.org/terra/.