Aligning and Updating Cadaster Maps with Aerial Images by Multi-Task, Multi-Resolution Deep Learning

Nicolas Girard[1], Guillaume Charpiat[2], and Yuliya Tarabalka[1]

[1] TITANE team, INRIA, Université Côte d'Azur, France
[2] TAU team, INRIA, LRI, Université Paris-Sud, France
firstname.lastname@inria.fr

Abstract. A large part of the world is already covered by maps of buildings, through projects such as OpenStreetMap. However when a new image of an already covered area is captured, it does not align perfectly with the buildings of the already existing map, due to a change of capture angle, atmospheric perturbations, human error when annotating buildings or lack of precision of the map data. Some of those deformations can be partially corrected, but not perfectly, which leads to misalignments. Additionally, new buildings can appear in the image. Leveraging multi-task learning, our deep learning model aligns the existing building polygons to the new image through a displacement output, and also detects new buildings that do not appear in the cadaster through a segmentation output. It uses multiple neural networks at successive resolutions to output a displacement field and a pixel-wise segmentation of the new buildings from coarser to finer scales. We also apply our method to buildings height estimation, by aligning cadaster data to the rooftops of stereo images. The code is available at https://github.com/Lydorn/mapalignment.

Keywords: Alignment · Registration · Multi-task · Multi-resolution

1 Introduction

Having a precise map of buildings is crucial for numerous applications such as urban planning, telecommunications and disaster management [9]. In the field of remote sensing, satellite or aerial images are used to retrieve meaningful geolocalized information. This may be the location of man-made objects, animals or plants, or delimitation perimeters of semantic areas such as forests, urban areas and buildings. The general goal is to automatically produce a map of the world.

One of the challenges in remote sensing is to deal with errors in the data. Each pixel in an image can be geolocalized by knowing the position and angle of the satellite, as well as the digital elevation model which gives the elevation of every point on earth. These data contain uncertainty errors, especially the

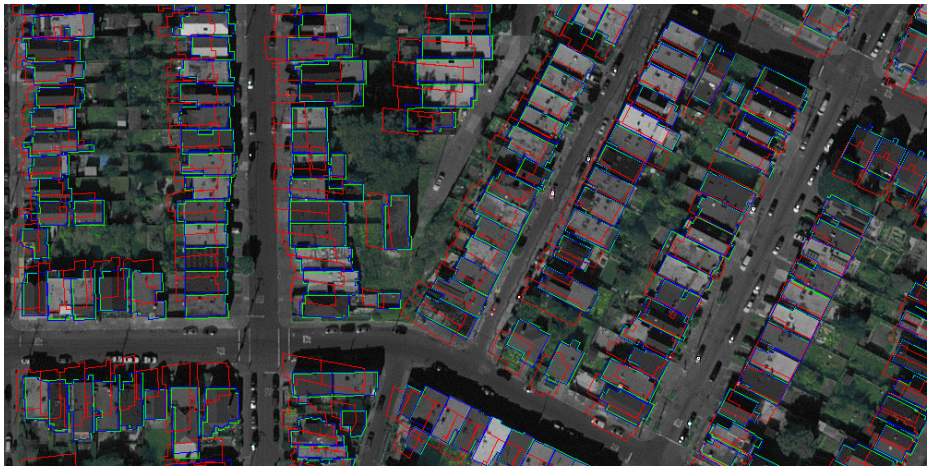


Fig. 1: Crop of one of the test images. Green buildings: ground truth; red: misaligned [input]; blue: aligned [our output].

digital elevation model which typically has a spatial resolution of 10 meters and a vertical resolution of 1 meter [14]. Also, atmospheric effects introduce errors, which might be reduced to an extent [15], but might still be there. Thus even if an object is accurately detected in an image, its geolocalization error may be high. Human-made maps may also suffer from geolocalization errors, due to human error [16] or to simply a lack of precision in the source material (scanned cadaster data from local authorities). Therefore, large misalignments frequently occur between a remote sensing image and an existing map. We observed displacements of up to 8 meters in OpenStreetMap [10], which translates to 27px in a 30cm/px image, and such displacements are not constant across the image.

There is a considerable amount of ground truth data available in the form of existing maps, that could potentially be used to learn the task of automatically mapping the world from remote sensing images, but it is useless in its current form because of such misalignments. We thus aim at correcting them.

Solving this alignment problem, also known as non-rigid image registration, can be formulated as the search for a dense displacement field between two given images: here, a cadaster map and a satellite photography. Each pixel of the first image is thus associated with a 2D displacement vector describing where it is mapped to in the second image. Non-rigid registration also arises notably in medical imaging, where different scans need to be aligned (scans of different patients, or of the same patient at different times, or of different modalities). Classically, alignment problems are classified in two categories: mono-modal or multi-modal, depending on whether the two images to be registered are taken by the same type of sensor or not.

Classical methods for mono-modal image alignment use image similarity measures or key-point descriptors such as SIFT [7] or HOG [3] to match parts of the

2 images with each other [11]. More recent methods use CNNs to predict the optical flow between two images [4]. These methods rely intrinsically on appearance similarities and thus cannot be extended to the multi-modal setting, which is our case of interest. Note e.g. that trees can cover part of the objects of interest in the RGB image while trees are not indicated on the cadastral map, so direct point-to-point appearance matching would fail. However a recent method considered structural similarity [18] to define a new feature descriptor for key-point matching.

These methods are not specific to certain types of images or objects, which is why they are widely used. However machine learning and more recently deep learning methods have achieved state-of-the-art performance on many computer vision problems by learning the best features for the task at hand. Several deep learning methods for image registration have been proposed, for instance Quicksilver [17] learns an image similarity measure directly from image appearance, to predict a dense deformation model for applications in medical imaging. However it works best for rather small displacements and with 2 image-like modalities. Also from the field of medical imaging, the fully-convolutional neural network U-Net [12] has been widely-used for image segmentation. Its use of skip-connections at intermediate levels of the network performs very well for predicting a spatially precise output that corresponds pixel-to-pixel to the input image. A new deep learning method for remote sensing imaging using a U-Net-like model has been proposed by [19], in a multi-resolution approach to predict large displacements. Indeed, solving for increasing resolutions iteratively has proved successful in a number of applications [5,2].

Another success in machine learning has been the use of multi-task learning [13]. The idea is to learn multiple tasks at the same time with a single neural network instead of one network per task. This allows the network to train better as it learns common features from all tasks while still being able to learn task-specific features. It also has the advantage of producing a single neural network, smaller and faster than the compilation of individual task-specific networks.

We propose a deep learning method that uses the multi-resolution approach from [19] and aims to improve the results by training the network with a multi-task objective. The primary objective is to directly compute a dense displacement map (or flow) that aligns the building cadaster to the image (for example building cadaster from OpenStreetMap). See Fig. 1 for a visual result of our alignment method. The second objective is to output a segmentation of the buildings from the image (otherwise known as pixel-wise classification) to help train the network and detect new buildings as well which can be used to update a map with missing buildings or recently-built buildings. The contributions of this work are:

(i) the design of a fully-convolutional neural network able to correct and update existing cadastral maps. Multi-task learning is used to improve alignment performance and to provide new building detection at no additional cost;
(ii) the use of intermediate losses inside the network to help gradients flow and improve final performance on both objectives;
(iii) random dropping of input polygons, to force the detection of new objects.

After presenting the methodology in Section 2, we present our experimental setup in Section 3; we apply our method to the building alignment task and evaluate the results in Section 4. We show another applicative potential of our method on building height estimation, from a pair of images and misaligned building cadaster data, in Section 5.

2 Methodology

2.1 Objective functions

Mathematical modeling. Given two images A_1 and A_2 of same size $H \times W$, but of different modalities, e.g. with A_1 an RGB image (picture from a satellite) and A_2 a binary image (cadaster, indicating for each pixel whether it belongs to a building or not), the alignment problem aims at finding a deformation, i.e. a 2D vector field $\mathbf{f}$ defined on the discrete image domain $[1, H] \times [1, W]$, such that the warped second image $A_2 \circ (\mathrm{Id} + \mathbf{f})$ is well registered with the first image A_1. To do this, in a machine learning setting, we will provide examples of image pairs (A_1, A_2) as inputs, and ask the estimated deformation $\hat{\mathbf{f}}$ to be close to the ground truth deformation $\mathbf{f}_{\mathrm{gt}}$.

For the segmentation problem, only the RGB image A_1 is given as input, and the desired output is an image of same size, expressing building presence probability for each pixel. The ground truth segmentation is thus the perfectly-registered cadaster $A_2 \circ (\mathrm{Id} + \mathbf{f}_{\mathrm{gt}})$.

Displacement field map cost function. The displacement field map loss function is the mean squared error between the predicted displacement field map ($\hat{\mathbf{f}}$) and ground truth displacement field map ($\mathbf{f}_{\mathrm{gt}}$). However, the displacement cannot be predicted equally well on every pixel in the image. For example, building corners can be precisely matched, while building boundary pixels have ambiguous displacement along one dimension (the boundary), which is classically known as the aperture problem, and whereas pixels $\mathbf{x}$ inside homogeneous cadaster zones (i.e. far inside or outside buildings) suffer from ambiguity in both spatial dimensions and thus cannot hope for a precise displacement vector $\mathbf{f}(\mathbf{x})$. To take this into account, we distinguish 4 different classes of pixels on the input cadaster image A_2 (which can be seen as a misaligned polygon raster image): background, polygon interior, edge, and vertex, in decreasing difficulty order. We denote by $c(\mathbf{x}) \in \{1, 2, 3, 4\}$ the class of a pixel $\mathbf{x} \in [1, H] \times [1, W]$. We apply a different loss coefficient w_c on each pixel class c, making the loss a weighted average of square errors. The loss coefficients are of increasing values from background pixels to vertex pixels ($w_1 \leq w_2 \leq w_3 \leq w_4$). As pixel classes are unbalanced, the loss of a pixel $\mathbf{x}$ is normalized by the pixel count of its corresponding class, denoted by $n_{c(\mathbf{x})}$. The displacement field cost is thus defined as:

$$L^{\mathrm{disp}}(\hat{\mathbf{f}}) = \sum_{\mathbf{x} \in [1,H] \times [1,W]} \frac{w_{c(\mathbf{x})}}{n_{c(\mathbf{x})}} \left\| \hat{\mathbf{f}}(\mathbf{x}) - \mathbf{f}_{\mathrm{gt}}(\mathbf{x}) \right\|_2^2 \tag{1}$$

Segmentation cost function. As the task of aligning buildings requires to be able to detect where buildings are, we consider an additional segmentation task, to help the training. For each pixel $\mathbf{x}$, and for each class $c \in \{\text{background, polygon interior, edge, vertex}\}$ independently, we predict the probability $\hat{p}^c(\mathbf{x})$ that a pixel $\mathbf{x}$ belongs to class c. The associated loss function is the sum of class-specific cross-entropies $KL\left(\mathcal{D}(p^c_{\text{gt}}) \,\|\, \mathcal{D}(\hat{p}^c)\right)$, where $p^c_{\text{gt}}(\mathbf{x})$ is the binary ground truth (whether pixel $\mathbf{x}$ is of class c), and where $\mathcal{D}(p)$ stands for the distribution $(p, 1-p)$ over the two possibilities (of class c or not). We also apply different coefficients w'_c for each class, to put more emphasis on vertices than edges, and than interior and background ($w'_1 \leq w'_2 \leq w'_3 \leq w'_4$). The segmentation cost function is thus:

$$L^{\text{seg}}(\hat{\mathbf{p}}) \;=\; \frac{1}{HW} \sum_{\mathbf{x}\in[1,H]\times[1,W]} \sum_{c=1}^{4} w'_c \, KL\left(\mathcal{D}(p^c_{\text{gt}}) \,\|\, \mathcal{D}(\hat{p}^c)\right). \tag{2}$$

2.2 Neural network with double inputs and outputs

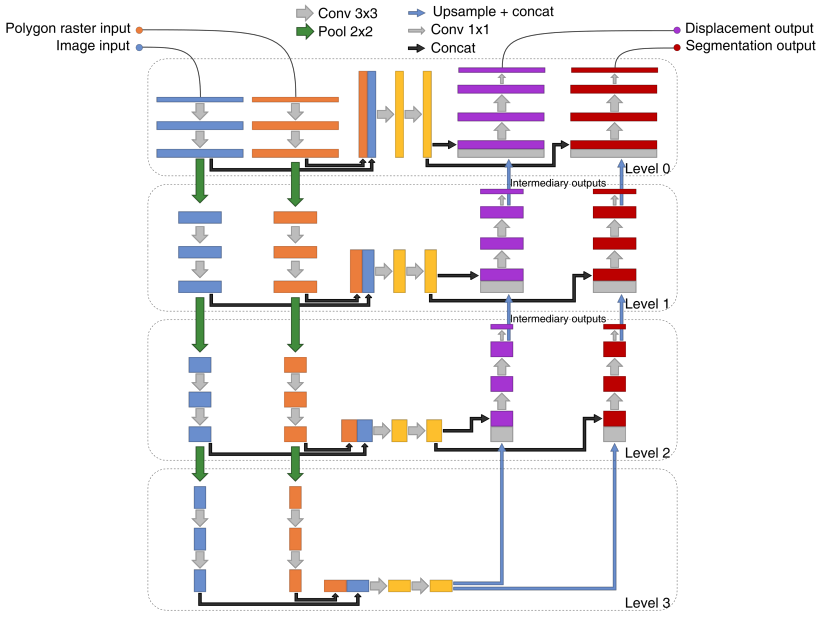
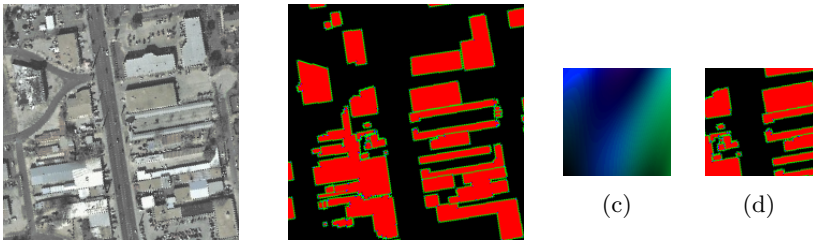


Fig. 2: Model architecture for one resolution. There are 4 different levels inside the network: one after each pooling operation in addition to the first level.

To address both tasks (alignment and segmentation), the main building block of the method is designed as a neural network with 2 image inputs and 2 image outputs (see Fig. 2). It uses skip-connections at multiple levels in the network like U-Net [12]. Model inputs and outputs examples are shown in Fig. 3. The input image A_1 has 3 channels, with real values in $[-1, 1]$, standing for RGB. The input

misaligned polygon raster A_2 has also 3 channels, with Boolean values in $\{0, 1\}$, corresponding to polygon interior, edge and vertices. The output displacement field map has 2 channels with real values in $[-1, 1]$, standing for the x and y components of the displacement vector. The output segmentation (or pixel-wise classification) has 4 channels: one for the background class, and three as for the input polygon raster.



(a) (b) (c) (d)

Fig. 3: (a) input image, (b) input misaligned polygon raster, (c) output displacement field map, (d) output segmentation.

The network is fully convolutional and uses only 3×3 convolutional kernels without padding, which reduces the image size slightly after every layer but avoids border artifacts. A 220×220 px input image thus leads to a 100×100 px output. The first layer of the A_1 input image branch has 32 convolutional filters. Then the number of filters doubles after each pooling operation. For the misaligned polygon raster input branch (A_2), the first layer requires 16 filters only, as polygon rasters are less complex than standard images. In total, the network has about 9 million trainable parameters. To train this network block, the ground truth displacement map has values in [-4 px, 4 px] which are normalized to [-1, 1] to match the output value range.

2.3 Multi-resolution

From coarse to fine. The multi-resolution approach iteratively applies a neural network at increasing resolutions (see Fig. 4 for a diagram of the multi-resolution pipeline). By solving the alignment problem from coarse to fine resolutions, the difficulty at each resolution step becomes drastically lower than for the whole problem. At the first step, the network is applied to the inputs downscaled by a factor of 8. Assuming the initial displacements to predict are in the range [-32px, 32px], the new low-resolution ones are within [-4px, 4px] only, reducing significantly the search space. Then each next resolution step multiplies the image resolution by a factor 2, and supposes that the remaining, finer deformation to be found is within [-4px, 4px] at that scale (the larger displacements having been found at the coarser scales). Note that we could multiply the resolution by a factor 4 at each step, but that, to allow for a scale overlap and increase stability, we keep a factor 2.

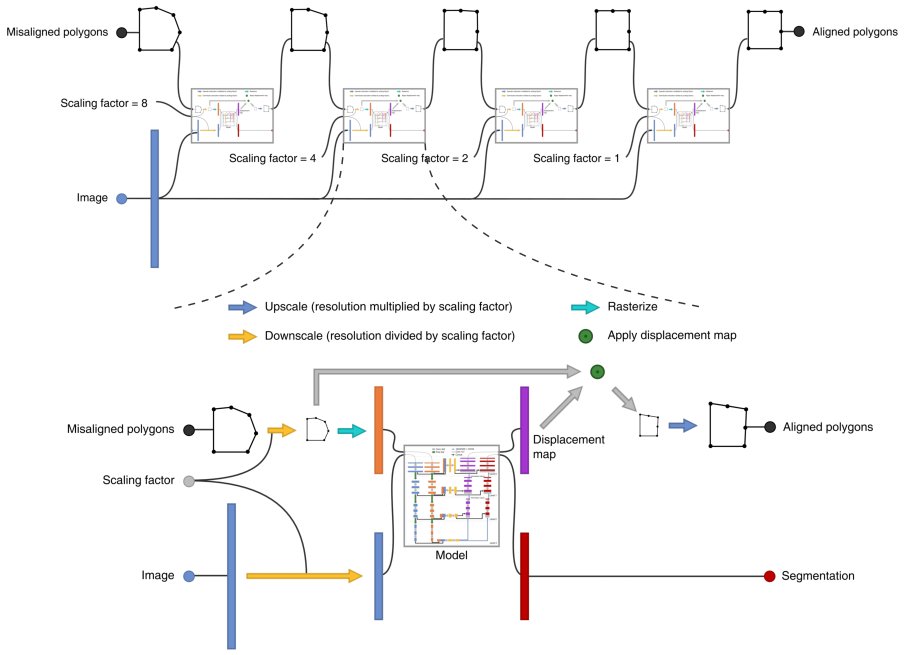


Fig. 4: Multi-resolution pipeline.

Intermediate cost functions. When training, the network quickly learns to output a null displacement map, as it is the average of the ground truth displacement maps, and is the best possible constant output. To help the gradients flow and avoid the network being stuck in a local minimum, we added intermediate outputs in the displacement map and segmentation branches at levels $l = 0, 1, 2$ of each resolution-specific block (see Fig. 2). The size of these intermediate outputs increases from inside the network block towards the final block outputs. The corresponding loss functions are applied to these intermediary outputs with different coefficients α_l, and denoted by L_l^{disp} and L_l^{seg}. As the training advances, the intermediary losses' coefficients α_l are pushed to zero so that only the final output is optimized. This helps the optimization process well:

$$L_{\text{total}}^{\text{disp}} = \sum_{l=0}^{2} \alpha_l L_l^{\text{disp}} \qquad (3) \qquad\qquad L_{\text{total}}^{\text{seg}} = \sum_{l=0}^{2} \alpha_l L_l^{\text{seg}} \qquad (4)$$

Final objective function. The final objective function is a linear combination of the displacement and segmentation cost functions: $L = \lambda_1 L_{\text{total}}^{\text{disp}} + \lambda_2 L_{\text{total}}^{\text{seg}}$.

3 Experimental setup

3.1 Datasets used

We perform experiments on a dataset made of the two available following ones:

1. Inria Aerial Image Labeling Dataset [8],
2. Aerial imagery object identification dataset for building and road detection, and building height estimation [1].

The first dataset has 360 images of 5000×5000 px for 9 cities from Europe and the U.S. Each image has in average a few thousand buildings. The second dataset has 24 images of about 5000×5000 px and 1 image of about 10000×10000 px for 9 cities from the U.S. The building footprints were pulled from OSM for both datasets. We removed images whose OSM data is too misaligned, keeping images where it is relatively good. We split the second dataset into 3 sets: 8 images for training, 3 for validation and 3 for testing. To develop our method to generalize to new cities and capture conditions, each set does not contain images from the same cities under the same capture conditions as any of the other 2 sets. From the first dataset, 332 images were picked for training and 26 for validation. These datasets may seem small compared to other deep learning datasets which can contain millions of images, but because each image is very big, they provide enough data. For example, our testing dataset contains 13614 buildings. More information on the splitting of these datasets into train, validation and test sets can be found in Section 3 of the supplementary materials.

3.2 Data preprocessing

Displacement map generation. The model needs varied ground truth displacement maps in order to learn, while the dataset is made of perfectly aligned image pairs only ($\mathbf{f} = 0$). We generate these randomly in the same way as described in [19], by generating normalized 2D Gaussian random fields added together for each coordinate (see Fig. 3(c) for an example). The displacements are then scaled so that the maximum absolute displacement is 32 px. The ground truth polygons are then inversely displaced by the generated displacements to compute the misaligned polygons which are then rasterized ($A_2 \circ (\mathrm{Id} + \mathbf{f})$).

Scaling and splitting into patches. All the images of the datasets do not have the same ground sample distance (pixel width measured on the ground in meters), which is a problem for a multi-resolution approach which learns a specific model per resolution. As more than 90% of the images of our dataset have a ground sample distance of 0.3 meters, we rescale the other images to that ground sample distance so that they all match. Then every data sample (image, misaligned polygon raster, and displacement map), which usually is of size 5000×5000 px, is rescaled to 4 different resolutions defined by the scaling factors. For example a scaling factor of 2 rescales the data sample to a size of 2500×2500 px, resulting in a ground sample distance of 0.6 m. Successive rescaling steps are performed as one, to limit interpolation errors of the scaling operation. Finally, data samples are split into patches of size $220 * \sqrt{2} = 312$ px with a stride of $100/2 = 50$ px, to account for rotations in the data augmentation step.

Classical data augmentations. To augment the dataset, classical augmentation techniques were used: perturb image brightness, contrast, saturation randomly, rotate by a random real angle $\theta \in [0, 2\pi]$, random horizontal flip and crop to the final input size of 220×220 px.

Random dropping of input polygons. With displacements of only up to 4px, it could be easy for the network to keep a small error by outputting, as a segmentation, just a copy of the input polygon raster A_2. This behavior does not allow the network to learn buildings from the input image A_1, and it cannot learn to detect new buildings either. To avoid this, we randomly drop (remove) polygons from the polygon raster input before feeding it to the network. This introduces a new hyper-parameter $keep_poly_prob \in [0, 1]$ which is the probability each input polygon is kept and actually fed to the network while training. This operation is also data augmentation in that it generates multiple possible inputs from one data sample.

3.3 Training

As shown in Fig. 4, 4 different models are used with scaling factors 8, 4, 2 and 1. We trained 4 models independently for each scaling factor. We used the Adam optimizer with batch size 32 on a GTX 1080 Ti. We used a learning rate of $1e^{-4}$ until iteration 25000 and then $0.5e^{-4}$ until the end (100000 iterations). We also used weight L_2 regularization with a factor of $1e^{-4}$. Tab. 1 summarizes the loss coefficients for the intermediate losses (Eq. 3,4) as well as for the different pixel classes (Eq. 1,2). We set $keep_poly_prob = 0.1$ to only keep 10% of input polygons, in order to learn new buildings on 90% of the buildings.

Table 1: Intermediate loss coefficients α_l and class loss coefficients w_c, w'_c.

Up to iter.	2500	5000	7500	100000
α_0	0.50	0.75	0.9	1.0
α_1	0.35	0.20	0.1	0
α_2	0.15	0.05	0	0

	w_c	w'_c
Background ($c = 1$)	0	0.05
Interior ($c = 2$)	0.1	0.1
Edge ($c = 3$)	1	1
Vertex ($c = 4$)	10	10

4 Cadastral map alignment

4.1 Results

To evaluate the method and its variations for the alignment task, we applied the full pipeline on the 3 images of the city of San Francisco from the second dataset. For each image, we generated 10 different displacement maps for a more precise evaluation. For visual results of the alignment, see Fig. 5.

To measure the accuracy of the alignment, for any threshold τ we compute the fraction of vertices whose ground truth point distance is less than τ. In other

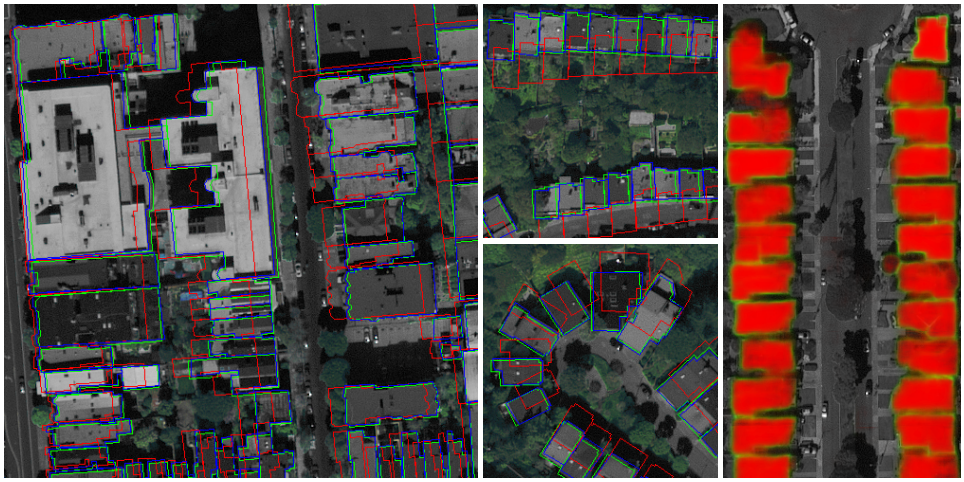


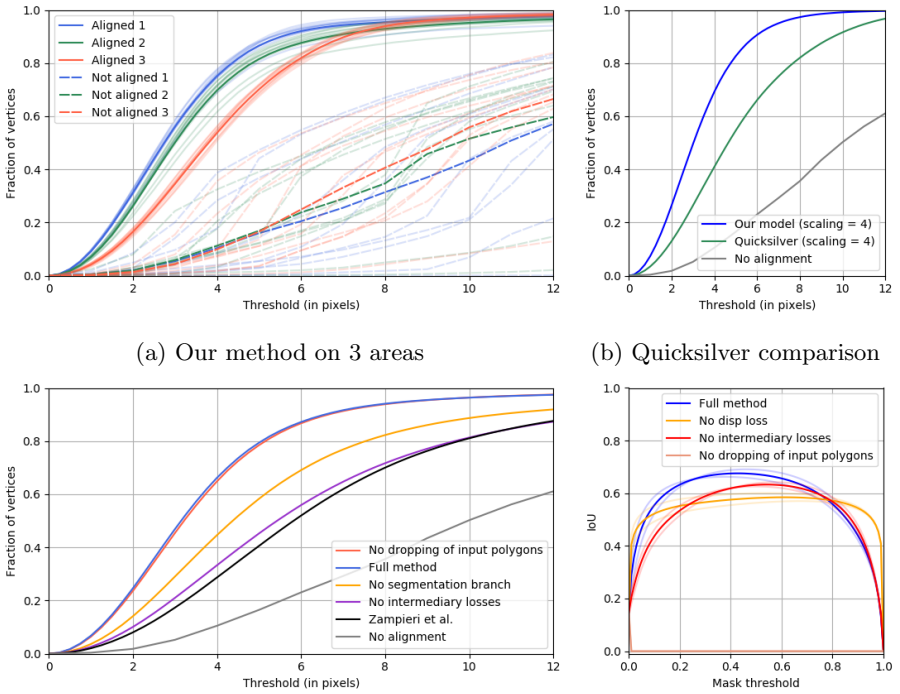
Fig. 5: Several crops of test images. Green buildings: ground truth; red: misaligned [input]; blue: aligned [our output]. The right crop is an example of the segmentation output.

words, we compute the Euclidean distance in pixels between ground truth vertices and aligned vertices, and plot the cumulative distribution of those distances in Fig. 6a (higher is better). The no alignment curves are for comparison and show the accuracy obtained if the output displacement map is zero everywhere.

We compared our method to Zampieri et al. [19], which we trained on the same dataset; its test accuracy is summarized in Fig. 6c. We also compare to Quicksilver [17]; however it could not handle the 32px displacements we tested on (it gave worse results than no alignment), so we trained Quicksilver with images downscaled by a factor of 4 and displacements of 4 px maximum at the downscaled resolution. We compare this version of Quicksilver with the model of our method trained with the same downscaling factor for a fair comparison. Fig. 6b shows the result of this comparison.

To test the segmentation of new buildings, we apply the model trained at the highest resolution (with a scaling factor of 1) with just the image as input. The polygon raster input is left blank as would be the case with an empty map. In this extreme case, all buildings in the image are new (with respect to the empty polygon raster input). See the right image of Fig. 5 for an example of segmentation. We measure the IoU (Intersection over Union) between the ground truth polygons and the output polygon raster (which combines the polygon interior, edge and vertex channels of the model's output). The polygon raster has values between 0 and 1, we threshold it at various values to obtain a polygon mask. The IoU (Intersection of Union) is then computed for the 3 test images (see Fig. 6d for the mean IoUs).

We proceeded to ablation studies to measure the performance gains of the 3 main contributions of this work. For the alignment objective, we removed the segmentation branch in a first experiment, removed all intermediary losses in a second experiment and set the probability of dropping input polygons to 1 in



(a) Our method on 3 areas

(b) Quicksilver comparison

(c) Ablation studies

(d) Segmentation task

Fig. 6: Accuracy cumulative distributions and mean IoU values. Faded curves each correspond to one of the 10 polygon maps to align per image. Solid curves are the average.

a third experiment. See Fig. 6c for the mean accuracy cumulative distributions of these experiments. For the segmentation objective, we set the probability of dropping input polygons to 1 in a first experiment, removed all intermediary losses in a second experiment and removed the displacement branch in a third experiment. See Fig. 6d for the mean IoUs of these experiments.

We also tested the generalization across datasets by using a third dataset. The networks trained on this new dataset plus the dataset by Bradbury et al. [1] generalize well to the Inria dataset. This experiment can be found in the supplementary materials.

4.2 Discussion

The ground truth from OSM we used is still not perfect, we can see some misalignment of a few pixels in some places. But our method still can learn with this ground truth.

We observe that the final alignment accuracy does not depend much on the initial misalignment. As can be seen in Fig. 6a for each area, accuracy curves corresponding to “not aligned” have a lot of variability (because of the randomly-

generated displacements), whereas accuracy curves corresponding to "aligned" have a very low variability.

We show in Fig. 6c that we improve upon the method of Zampieri et al. and in Fig. 6b that our model performs better than Quicksilver in the one-resolution setting with small displacements.

Impact of the segmentation branch. When removing the segmentation branch altogether, we globally lose accuracy on the alignment task. Fig. 6c shows a drop of 22% for a threshold value of 4px. The segmentation branch was initially added to stabilize the beginning of the training, it turns out it also improves the final performance of the network as well as providing a way to detect new buildings that do not appear in the input misaligned building map. We chose to classify polygon edges and vertices in addition to the very common polygon interior classification, in order to control how much the network has to focus on learning edges and vertices to optimize its cost function. It also gives more information compared to a classical building classification map. Classifying building edges allows for the separation of individual buildings that touch each other as can be seen in the right image of Fig. 5. The final goal of using the segmentation output is to vectorize the bitmap to obtain building polygons that can be added to an existing map. Detecting vertices, edges and interior of buildings should help this vectorization step which we hope to explore in a future work.

Impact of the displacement branch. When removing the displacement branch, we also lose accuracy on the segmentation task. Fig. 6d shows a relative gain of the full method in terms of area under the curve of 4.6% compared to not using multi-task learning.

Impact of intermediary cost functions. We lose more alignment accuracy when not using intermediary cost functions by setting $\alpha_0 = 1, \alpha_1 = 0$ and $\alpha_2 = 0$. Fig. 6c shows a drop of 32% for a threshold value of 4px. It also affects the segmentation task. Fig. 6d shows a relative gain of the full method in terms of area under the curve of 8.3% compared to not using intermediary cost functions. This technique proves that using intermediary losses could improve semantic segmentation methods.

Impact of randomly dropping input polygons. If we set the probability of dropping input polygons to 0 (equivalent to not using this technique), the network almost does not output anything for the segmentation output, and consequently the IoU gets very low (see Fig. 6d). The reason is that it learns to output a building only when there is a building in the input. We checked that this technique does not decrease the performance of the alignment task: Figure 6c shows that the 2 curves corresponding to using this technique (full method) and not using it (no dropping of input polygons) are equivalent.

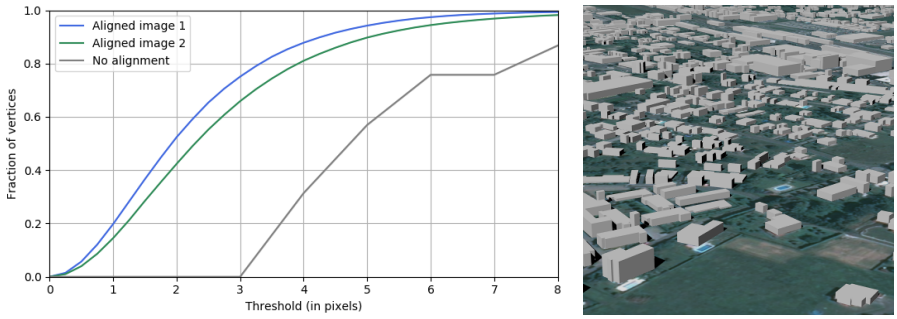
5 2.5D reconstruction of buildings

5.1 Method

The alignment of building polygons can be used to solve the building height estimation problem to reconstruct 2.5D buildings from a stereo pair of images. The inputs to the method are 2 orthorectified satellite images captured from different views and a cadaster map of buildings which does not need to be precise. The first step consists in using our alignment method to align the building cadaster to the first image and to the second image. We then measure for each building the distance between the corresponding aligned buildings in both images. From this distance it is possible to compute the height of the building with trigonometry (assuming the ground is flat). For this we need to know the elevation (e_i) and azimuth (a_i) angles of the satellite when it captured each image i. We first used Eq. 19 from [6] which uses only elevation angles because they assume both images are captured from the same side of nadir and also that the satellite ground path goes through the captured area (meaning the azimuth angles are the same). We generalize their formula by including azimuth angles, thus linking the building height H to the distance D between the 2 aligned building polygons as:

$$H = D \frac{\tan(e_1)\tan(e_2)}{\sqrt{\tan^2(e_1) + \tan^2(e_2) - 2\tan(e_1)\tan(e_2)\cos(a_1 - a_2)}} \tag{5}$$

5.2 Results and discussions



(a) Accuracy cumulative distributions (b) 2.5D buildings

Fig. 7: Height estimation by alignment of polygons of one view to the image of the other view and vice versa.

We applied this method to a pair of Pléiades satellite images of the city of Leibnitz. So that the accuracy can be quantified, ground truth buildings with a height attribute were given to us by the company Luxcarta along with the satellite images. Image 1 was captured with an elevation and azimuth angles of 76.7°and 212.9°respectively. For Image 2 those angles are 69.6°and 3.6°. We aligned the building polygons to the 2 images with our method; the accuracy of the alignment is displayed in Fig. 7a. The models of our method were trained

on aerial images only and were applied as-is to these 2 satellite images. We use the same elevation and azimuth angles for every building of an image, as our data comprises the mean angles of the images only. In reality, the angles are not constant across the image and Eq. 5 can be used with different angles per building. See Fig. 7b for the 2.5D buildings obtained by this method. For each building we measure the difference between the ground truth height and the predicted height. We obtain a mean error of 2.2 meters. We identified 2 sources of error:

1. Our model did not train on images far from nadir (lack of ground truth data at those angles). This is why Image 2 has lower alignment accuracy than Image 1: being farther from nadir, the network has more trouble with it.
2. Alignment errors get multiplied by a greater factor when the elevation angle is near nadir (closer to 90°) than when it is farther from it (closer to 0°).

These 2 sources of errors should be solved by the same solution: training the networks on images farther from nadir with good ground-truth.

6 Conclusions

The multi-task, multi-resolution method presented in this paper can be used to effectively solve the common problem of aligning existing maps over a new satellite image while also detecting new buildings in the form of a segmentation map. The use of multi-task learning by adding the extra segmentation task not only helps the network to train better, but also detects new buildings when coupled with a data augmentation technique of randomly dropping input polygons when training. Adding intermediate losses at different resolution levels inside the network also helps the training by providing a better gradient flow. It improves the performance on both alignment and segmentation tasks and could be used in other deep learning methods that have an image-like output which can be interpreted at different scales. Interestingly, multi-task learning also helps the segmentation task, as adding the displacement loss when training increases IoU.

We also tried our method on the task of building height estimation, generating simple but clean 3D models of buildings. We hope that our method will be a step towards automatically updating maps and also estimating 3D models of buildings. In the future we plan to work on the segmentation branch by using a better-suited loss for each output channel and a coherence loss between channels, to further improve map updating. We also plan to improve displacement learning by adding a regularization loss enforcing piece-wise smooth outputs, such as the BV (Bounded Variation) norm or the Mumford-Shah functional.

7 Acknowledgments

This work benefited from the support of the project EPITOME ANR-17-CE23-0009 of the French National Research Agency (ANR). We also thank Luxcarta for providing satellite images with corresponding ground truth data and Alain Giros for fruitful discussions.

References

1. Bradbury, K., Brigman, B., Collins, L., Johnson, T., Lin, S., Newell, R., Park, S., Suresh, S., Wiesner, H., Xi, Y.: Aerial imagery object identification dataset for building and road detection, and building height estimation (Jul 2016)
2. Charpiat, G., Faugeras, O., Keriven, R.: Image statistics based on diffeomorphic matching. In: ICCV (Oct 2005)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (June 2005)
4. Fischer, P., Dosovitskiy, A., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. CoRR (2015)
5. Hermosillo, G., Chefd'Hotel, C., Faugeras, O.: Variational methods for multimodal image matching. International Journal of Computer Vision (Dec 2002)
6. Licciardi, G.A., Villa, A., Mura, M.D., Bruzzone, L., Chanussot, J., Benediktsson, J.A.: Retrieval of the height of buildings from worldview-2 multi-angular imagery using attribute filters and geometric invariant moments. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (Feb 2012)
7. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision (Nov 2004)
8. Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P.: Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In: IGARSS (2017)
9. Manfr, L.A., Hirata, E., Silva, J.B., Shinohara, E.J., Giannotti, M.A., Larocca, A.P.C., Quintanilha, J.A.: An analysis of geospatial technologies for risk and natural disaster management. ISPRS International Journal of Geo-Information (2012)
10. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org (2017)
11. Ptucha, R., Azary, S., Savakis, A.: Keypoint matching and image registration using sparse representations. In: IEEE International Conference on Image Processing (Sept 2013)
12. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. CoRR (2015)
13. Ruder, S.: An overview of multi-task learning in deep neural networks. CoRR (2017)
14. Thompson, J.A., Bell, J.C., Butler, C.A.: Digital elevation model resolution: effects on terrain attribute calculation and quantitative soil-landscape modeling. Geoderma (2001)
15. Wang, A.: Correction of atmospheric effects on earth imaging (extended inverse method). Mathematical Modelling (1987)
16. Wright, J.K.: Map makers are human: Comments on the subjective in maps. Geographical Review (1942)
17. Yang, X., Kwitt, R., Niethammer, M.: Quicksilver: Fast predictive image registration - a deep learning approach. CoRR (2017)
18. Ye, Y., Shan, J., Bruzzone, L., Shen, L.: Robust registration of multimodal remote sensing images based on structural similarity. IEEE Transactions on Geoscience and Remote Sensing (May 2017)
19. Zampieri, A., Charpiat, G., Tarabalka, Y.: Coarse to fine non-rigid registration: a chain of scale-specific neural networks for multimodal image alignment with application to remote sensing. CoRR (2018)