An Unsupervised Augmentation Framework for Deep Learning based Geospatial Object Detection: A Summary of Results

Yiqun Xie, Rahul Bhojwani, Shashi Shekhar
Dept. of Computer Science & Eng.
University of Minnesota - Twin Cities
xiexx347,bhojw005,shekhar@umn.edu

Joseph Knight
Dept. of Forest Res.; Remote Sensing & Geo. Analysis Lab
University of Minnesota-Twin Cities
jknight@umn.edu

ABSTRACT

Given remote sensing datasets in a spatial domain, we aim to detect geospatial objects with minimum bounding rectangles (i.e., angle-aware) leveraging deep learning frameworks. Geospatial objects (e.g., buildings, vehicles, farms) provide meaningful information for a variety of societal applications, including urban planning, census, sustainable development, security surveillance, agricultural management, etc. The detection of these objects are challenging because their directions are often heavily mixed and not parallel to the orthogonal directions of an image frame due to topography, planning, etc. In addition, there is very limited training data with angle information for most types of objects. In related work, state-of-the-art deep learning frameworks detect objects using orthogonal bounding rectangles (i.e., sides are parallel to the sides of an input image), so they cannot identify the directions of objects and generate loose rectangular bounds on objects. We propose an Unsupervised Augmentation (UA) framework to detect geospatial objects with general minimum bounding rectangles (i.e., with angles). The UA framework contains two schemes, namely a ROtation-Vector (ROV) based scheme and a context-based scheme. The schemes completely avoid the need for: (1) additional ground-truth data with annotated angles; (2) restructuring of existing network architectures; and (3) re-training. Experimental results show that the UA framework can well approximate the angles of objects and generate much tighter bounding boxes on objects.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → **Neural networks**; **Machine learning algorithms**;

KEYWORDS

Rotations; rectangles; geospatial objects; deep learning; remote sensing

ACM Reference Format:
Yiqun Xie, Rahul Bhojwani, Shashi Shekhar and Joseph Knight. 2018. An Unsupervised Augmentation Framework for Deep Learning based Geospatial Object Detection: A Summary of Results. In *26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18), November 6–9, 2018, Seattle, WA, USA.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3274895.3274901

SIGSPATIAL '18, November 6–9, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5889-7/18/11…$15.00
https://doi.org/10.1145/3274895.3274901

1 INTRODUCTION

Given a remote sensing dataset, we aim to automatically generate catalogs of geospatial objects (e.g., buildings, vehicles, farm fields) by leveraging deep learning frameworks for object detection. The detection of an object O is modeled by the **Minimum Bounding Rectangle (MBR)** of O, which is the smallest rectangle (i.e., angle-aware) that entirely covers O. Fig. 1 shows an example of an input image and the MBR output.

(a) Input image (b) MBR output

Figure 1: Example input and MBR output.

Catalogs of geospatial objects are valuable assets for many sectors and application domains [2, 4, 5, 15, 17]. For example, building footprints provide meaningful information for urban planning, solar suitability analysis, census, agricultural management, etc. The tracking of vehicles or ships is important for security surveillance, law enforcements and resource allocation. Fine-scaled farm field data facilitates the monitoring of agricultural land-covers, yield estimation, conservation planning, etc. With the increased availability of high-resolution remote sensing data (e.g., satellite and UAV imagery, LiDAR), there are great opportunities to automatically identify, record and track a variety of geospatial objects.

The problem is challenging because: (1) the directions of geospatial objects are often not parallel to the orthogonal axes of an image frame; (2) the availability of training dataset with annotated angles is very limited for general types of objects, and manual generation of such dataset is tedious and time-consuming, which also does not take advantage of existing rich training data without angles; and (3) Geospatial objects often have a dense distribution due to the limited usable or allocatable space, and loose area estimations (e.g., bounding rectangles that are not aware of the directions of objects) can cause heavy overlaps among detections.

In recent years, end-to-end deep learning frameworks have shown promising results in computer vision by outperforming traditional multi-stage methods, which are often combinations of

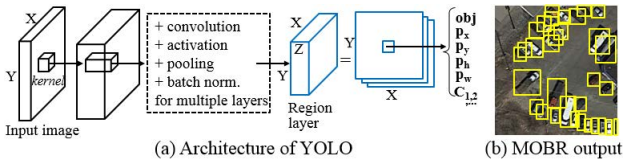


Figure 2: The YOLO framework and MOBR output.

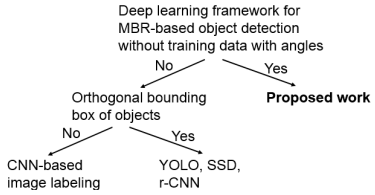


Figure 3: Novelty of the proposed approach.

manually-constructed features from image processing techniques (e.g., Histogram of Gradients) and classic machine learning algorithms (e.g., Support Vector Machines). In particular, Convolutional Neural Networks (CNN) have become the most popular architecture and set new benchmark performances in image classification and labeling [3, 16]. Compared to flat neural networks, CNN uses local-connections between layers as well as weight-sharing schemes to model the spatial adjacency of pixels and reduce the difficulty of learning. In remote sensing, CNN has also shown enhancements in learning and creating features for image classification tasks [11]. It has also achieved high precision in land-use and land-cover classification using satellite images [6, 14]. However, the original architectures of CNN only focus on image-theme (e.g., with or without certain objects) classification and cannot predict the locations and sizes of objects. You-Only-Look-Once (YOLO) and its variation Single Shot MultiBox Detector (SSD) are the recent state-of-the-art frameworks that extend CNN for object detection using additional regression modeling [8, 12]. The general regression ideas of these frameworks are very similar. In the following we provide a brief overview of YOLO's general architecture as an example.

YOLO is built on top of convolutional and pooling layers. Different from the original CNNs, it has a region layer at the end to estimate the **Minimum Orthogonal Bounding Rectangles (MOBR)** of objects using regression. An MOBR of an object O is the smallest rectangle, that has its sides parallel to the sides of an input image and entirely covers O. Fig. 2(b) shows an example of MOBR outputs. Compared to the MBRs in Fig. 1(b), the MOBRs are not flexible in directions.

Fig. 2(a) shows the general structure of YOLO and its region layer, where the X,Y dimensions represent the same two-dimensional space covered by the input image. The Z dimension of each cell contains a list of parameters used to estimate the existence of an object obj, location (p_x, p_y), size (p_h, p_w) and class C of a potential object. In the region layer, the cell that contains the center of an object is responsible for detecting it. Eq. (1) shows the regression modeling used to convert the raw parameters to the final MOBR.

$$(c_x,\ c_y) = \Big(x+\sigma(p_x),\ y+\sigma(p_y)\Big), \quad (l_w, l_h) = \Big(w\cdot e^{p_w},\ h\cdot e^{p_h}\Big) \quad (1)$$

where (c_x, c_y) is the center location of an MOBR estimated by offsets to the top-left corner (x, y) of the corresponding cell in the region layer; $\sigma()$ is the sigmoid function (output in the range $(0, 1)$); w, h are the prior (anchoring) width and height of objects; and l_w, l_h are the width and height of the MOBR.

Besides accuracy, another major advantage of deep learning frameworks over traditional image processing techniques is their demonstrated generality, which means one network architecture can be used to detect a large variety of general objects using different training datasets [12]. In comparison, traditional methods often require separate multi-phase designs and calibrations for each type.

On the other hand, the major limitation of these deep learning based object detection frameworks (e.g., YOLO [12], SSD [8], r-CNN [13]) is that they assume orthogonal directions for objects and use Minimum Orthogonal Bounding Rectangles (MOBR) to represent them. Thus, these frameworks do not consider the angles of objects, which often result in inaccurate size estimations and heavy overlaps among detections.

The main reason for this MORB design is that in general there is very limited publicly-available training data with angle attributes, and it is very time-consuming to generate such data for each specific problem [7]. Although we can train a sample deep learning model by re-creating a smaller amount training data with angle information, it is difficult to generalize it to large-scales or new applications, which still require large amounts of new training samples with angle information. This significantly limits the practical use of learning schemes that require angle information during training. It also cannot leverage the existing rich training data of MOBRs. Fig. 3 shows a classification tree of the related and proposed work.

To overcome these limitations, we propose an Unsupervised Augmentation (UA) framework to detect the general MBRs (i.e., not necessarily orthogonal) of geospatial objects (Fig. 1). The "unsupervised augmentation" aims to address the general challenge of the unavailability of training data with angles. Thus, it does not require any additional data with angle information to accomplish the task, as compared to supervised learning schemes [7, 9]. Within this framework, we present two schemes of augmentation, namely a ROtation-Vector (ROV) based scheme and a context-based scheme. Overall, the UA framework avoids the need for: (1) additional ground truth datasets with annotated angles of objects; (2) additional design of network architectures; and (3) re-training.

Through experiments, we confirmed that both ROV-Reflection and ROV-Spatial methods are able to reduce the effects of empty areas in rotated images, and that ROV-Spatial performs better than ROV-Reflection when source test imagery is available. In addition, both the ROV and the context based schemes can approximate the angles of objects well. The context based approach is more accurate when estimating angles while ROV is better at tightening the area estimations of objects.

Scope and outline: The scope of this problem is to leverage existing MOBR-based deep learning frameworks to detect general MBRs (i.e., with rotation angles and tighter bounding rectangles). The target objects are assumed to have a general rectangular shape from a top-down satellite view. The rest of the paper is outlined as follows: Sec. 2 formally defines the problem, Sec. 3 discusses the

proposed approaches, Sec. 4 presents analytical validations, Sec. 5 shows the experimental evaluations and Sec. 6 concludes the paper.

2 PROBLEM DEFINITION

The problem is formally defined as follows:

Inputs:

- Remote sensing images used for training;
- Ground truth of geospatial objects for each training image;
- Remote sensing images used for testing;

Outputs:

- A deep-learning based object detection framework, whose outputs are MBRs of geospatial objects;

Objectives:

- Detection accuracy of geospatial objects (i.e., angle, area);

Constraints:

- The spatial resolution of the remote sensing imagery is sufficiently high to distinguish the objects of interest;
- Not requiring ground truth data with angle information;

Fig. 1 (a) and (b) shows the example input and output of this MBR-based problem formulation. The second constraint addresses the challenge of the general unavailability of ground truth data with MBRs. In addition, we also aim to take advantage of existing MOBR-based deep learning framework and their training dataset of MOBRs. Thus, ideally the solutions could avoid the requirement for extra redesigning and retraining of separate deep learning architectures. This guarantees that existing MOBR-based frameworks (e.g., YOLO [12], SSD [8], r-CNN [13]) can be directly used for MBR detection.

3 THE PROPOSED UNSUPERVISED AUGMENTATION FRAMEWORK

The goal of "Unsupervised Augmentation" (UA) is to detect Minimum Bounding Rectangles (MBR) without the need for additional training data with MBRs. It uses a MOBR-based deep learning method as a sub-routine. Within this framework, we propose two unsupervised augmentation schemes, namely a ROtation-Vector (ROV) based scheme and a context based scheme.

3.1 Proposed Scheme 1: A Rotation-Vector based Approach

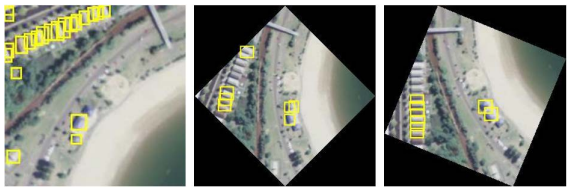
The ROtation-Vector (ROV) based approach augments each test image by rotating it at different angles using a rotation vector:

Definition 1. ***Rotation Vector.*** *A rotation vector V is a vector of m distinct rotation angles (counter-clock direction), where $V_i \in [0, 2\pi)$ and π is the radian corresponding to* 180°.

Note that ROV is not applicable to training because rotation will change the correct size of an MOBR and that change is undecidable with only one MOBR known at a fixed angle.

The idea of ROV is to get rotated representations of each object in the augmented test data, and then use detected MOBRs at these different angles to derive the best angle and size of its MBR. The key question is then how to rotate the image.

Choice of pivot and angles in ROV: Note that a rotation angle α is not the only parameter in a rotation operation; we also need to determine which pixel to use as the pivot. Intuitively, if we rotate the image by α at the center of an object, we can guarantee that the object in the rotated image is also rotated by α. However, in this object detection problem, since the object centers are unknown, this approach may lead to rotated images at all pixels in a test image, which will introduce a significant amount of computational and space cost. Besides, the second decision to make is the set of rotation angles to choose. In Def. 1, each angle is limited to range (radian) $[0, 2\pi)$ by default because of the general periodicity of angles. For example, with a fixed pivot, the rotation by 0 is exactly the same as the rotation by 2π. Since the goal here is to identify the angles of objects using the rotations, the periodicity can potentially be different. If a smaller periodicity exists, we can leverage it to reduce the size of the rotation vector (i.e., eliminating unnecessary angles).



(a) No empty area (b) Empty areas at $\pi/4$ (c) Empty areas at $5\pi/16$

Figure 4: Detections with and without empty areas.

To avoid lengthy analysis of the choices, here we directly present the key findings and decisions, which will be validated through theorems and proofs in Sec. 4 on analytical validation:

- We only need to rotate an image at its center regardless of the locations and angles of objects as well as the rotation angle α (Thm. 1, Sec. 4);
- The rotation vector only needs to consider angles in range $[0, \pi/2)$ without losing completeness (Thm. 2, Sec. 4).

Empty areas: Rotating an image generates empty areas around the boundary (Fig. 4). These empty areas (i.e., black triangles) may cause errors in detection. For example, the zero values of those "null" pixels could potentially reduce the activation values in the network layers, leading to lower confidence for objects near the boundary. Such effects may vary case by case and are generally hard to quantify and neutralize. Thus, the output tends to miss objects that are close to the empty areas (Fig. 4 (b) and (c)).

To address this issue, we propose two completion algorithms, namely ROV-Reflection and ROV-Spatial, to mitigate the effects of empty areas.

The input of a deep learning framework (e.g., YOLO, SSD) often has the same number of rows and columns (i.e., a square image). If it does not, it will be resized to a square shape. Thus, to avoid potential shape distortions of geospatial objects, we assume square-shaped input images in the remainder of the paper.

3.1.1 ROV-Reflection. ROV-Reflection completes the empty areas by reflecting the scenes in the image itself using the image borders as the mirror lines. As shown in Fig. 5, the black space on the top-left corner is completed using reflections from the inside of the image with the light-blue line as the mirror line. As we can see, the scenes in the red dashed triangles are symmetric along the mirror line. The other black spaces are filled in similarly.

Figure 5: ROV-Reflection: Filling the black space by mirroring the scenes in the test image. (best in color)

Figure 6: ROV-Spatial: Filling the black space by source.

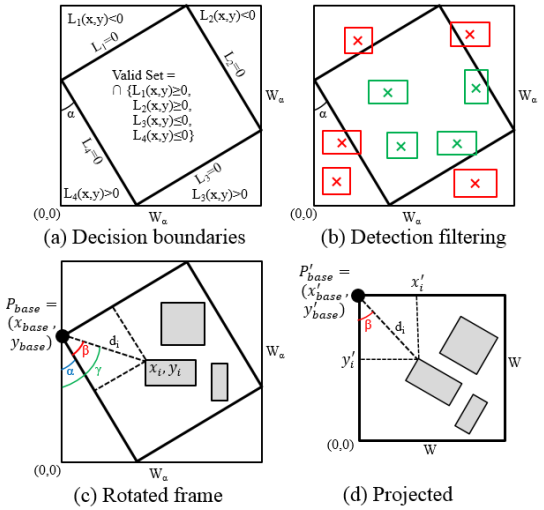


Figure 7: Filtering and projection.

There are four borders of a square-shaped test image (i.e., the lines between the image and the empty areas), and each of them can be used as a mirror line in reflection. For each pixel in the empty areas, we use its nearest border as the mirror line. Consider the bottom-left pixel of a rotated image (including empty areas) as the origin (0,0) of the coordinate system S that we use for reflection, and the width of a pixel as the unit length in S. For a pixel at row u and column v in a $W \times W$ image, we have its coordinates in S as (p_x, p_y), where $p_x = v-1$ and $p_y = W-u$. Denote the set of four mirror lines as L, where L_i is represented by $A_i x + B_i y + C_i = 0$, $\forall i = 1, ..., 4$. The reflected coordinate (p'_x, p'_y) is:

$$p'_x = -2A^* \frac{A^* p_x + B^* p_y + C^*}{(A^*)^2 + (B^*)^2} + p_x$$

$$p'_y = -2B^* \frac{A^* p_x + B^* p_y + C^*}{(A^*)^2 + (B^*)^2} + p_y$$

$$A^*, B^*, C^* = \underset{A_i, B_i, C_i \in L}{\operatorname{argmin}} \frac{|A_i \cdot p_x + B_i \cdot p_y + C_i|}{\sqrt{A_i^2 + B_i^2}}$$

The corresponding row u' and column v' of (p'_x, p'_y) are then $u' = [W - p'_y]$ and $v' = [p'_x + 1]$. The line parameters A_i, B_i, C_i for each line in L can be derived based on the rotation angle α (counter-clock direction) being used. For example, the parameters for L_1 (top-left corner) are: $A_1 = \tan\alpha, B_1 = -1, C_1 = \frac{\sqrt{2}W}{2} \cdot \frac{\cos\alpha}{\sin(\pi/4+\alpha)} - 1$.

3.1.2 ROV-Spatial. While ROV-Reflection completes the empty areas and makes the resulting image visually contiguous, the mirroring may still generate non-natural things (e.g., sharp angles caused by reflection) that could potentially affect the results. To address this, ROV-Spatial fills in the empty areas using original scenes from their corresponding spatial extents. This requires access to the large source imagery that was used to generate the test images, or necessary spatial references (e.g., projection, coordinates) of the test images which can be used to combine them into a single mosaic.

Once the source test imagery is located, ROV-Spatial rotates it at the geographic center of the test image by the same angle α and clip it using the bounding box of the rotated test image (red box in Fig. 6). This guarantees that each black space is filled with its actual missing data. As shown in Fig. 6, ROV-Spatial's augmentation introduces fewer artificial effects to the test image compared to ROV-Reflection. Note that the detections outside the rotated image (defined by the yellow dashed box) will be removed from the results.

3.1.3 Filtering, projection and grouping. Since the objects in a test image can have different rotation angles, it is not appropriate to use detections from a single rotation to represent the MBRs of all objects in the image. To address this issue, we propose a three-phase algorithm to identify the MBR of each individual object. For each image, the first phase filters out the detections that are outside the original image frame. The second phase projects the remaining detections from the rotated images (i.e., augmentations) to the original image, which creates multiple detections at different angles for each object. Finally, the last phase clusters the detections into local groups, each of which contains detections of the same object. Once the three phases are completed, we can identify the MBRs by selecting the detection with the minimum area in each group. For simplicity, we will use the bottom-left corner of each image as the origin (0,0) of its coordinate system, the bottom-border as the direction of the X-axis, and the left-border as the Y-axis.

Filtering: The filtering phase removes detections whose centers are not within the extent of the original image (i.e., the rotated square in Fig. 7(a)). Similar to Sec. 3.1.1, the borders of the original image in Fig. 7(a) are represented by the four lines $L_1(x, y) = 0$ to $L_4(x, y) = 0$, where $L_i(x, y) = A_i x + B_i y + C_i$. For example, the parameters for L_1 are $A_1 = \tan\alpha$, $B_1 = -1$, and $C_1 = (\frac{\sqrt{2}w_\alpha}{2}) \cdot (\frac{\cos\alpha}{\sin(\pi/4+\alpha)}) - 1$.

According to Fig. 7(a), we have the regions inside the original image as $S_{valid} = \cap\{L_1(x, y) \geq 0, L_2(x, y) \geq 0, L_3(x, y) \leq 0, L_4(x, y) \leq 0\}$. Then, if the center of a detection $(x_c, y_c) \notin S_{valid}$,

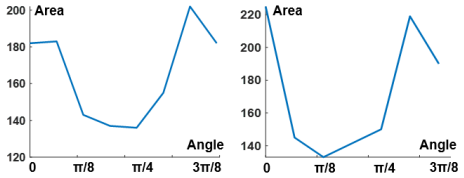


Figure 8: Area distribution by angles.

we remove it from the results. Fig. 7(b) shows the remaining and removed detections in green and red, respectively.

Projection: This phase maps all the detections from the rotated image back to the original image frame. Denote $\mathbf{P}_{base} = [x_{base}, y_{base}]$ and $\mathbf{P'}_{base} = [x'_{base}, y'_{base}]$ as the top-left corner of the original image in the rotated and the original image, respectively (Fig. 7 (c) and (d)). Denote $\mathbf{P} = [\mathbf{x}, \mathbf{y}] \in \mathbb{R}^{4\times 2}$ as a matrix of four corner points belonging to a detection in the rotated image frame, and $\mathbf{P'} = [\mathbf{x'}, \mathbf{y'}]$ as the projected matrix in the original image frame. As shown in Fig. 7(c) and (d), the projection of each point can be computed using its distance to $\mathbf{P}_{base}$ and the angle β, which is the difference between angle γ and the rotation angle α. The derivation is as follows:

$$\boldsymbol{\gamma} = \tan^{-1}((\mathbf{x} - x_{base}) \oslash (\mathbf{y} - y_{base}))$$

$$\boldsymbol{\beta}_i = \begin{cases} \pi + \boldsymbol{\gamma}_i - \alpha, & \text{if } \boldsymbol{\gamma}_i < 0 \\ \boldsymbol{\gamma}_i - \alpha, & \text{otherwise.} \end{cases}$$

$$\mathbf{x'} = D\Big([\mathbf{x}, \mathbf{y}], [x_{base}, y_{base}]\Big) \odot \sin(\boldsymbol{\beta})$$

$$\mathbf{y'} = w - D\Big([\mathbf{x}, \mathbf{y}], [x_{base}, y_{base}]\Big) \odot \cos(\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are vectors containing angles of each corner point of a detection; the subscript i denotes the i^{th} element of a vector; $\oslash$ and $\odot$ denote element-wise division and multiplication operations (i.e., Hadamard division and product), respectively; $D()$ is a distance function of two input matrices of points, whose output is a vector $\mathbf{d}$ with $\mathbf{d}_i = \|[\mathbf{x}_i, \mathbf{y}_i] - [x_{base}, y_{base}]\|_2$; and w is the side length of the original image.

Grouping: After all valid detections are projected back to the original image frame, we have multiple detections at different angles for each object in the scene. In order to group the detections by their corresponding objects, we keep the detections generated by different rotation angles in separate layers. Then, starting with a single detection d generated by a specific rotated angle α_1, we identify its nearest detection d' in each of the other layers and insert d' into the group if the distance between the centers of d and d' is smaller than $\min(d.diag, d'.diag)/2$, where $diag$ denotes the diagonal length of a detection. If there does not exist a valid d' in a layer, then we do not add anything into the group from that layer. Fig. 8 shows the distributions of detection's areas in two example groups. As we can see, the areas gradually converge to the minimum area from its two sides. By the definition of MBR (Sec. 2), we select the detection with the minimum area in each group as the best approximation of the object's MBR.

Alg. 1 shows the overall structure of the ROV method. In line 5, the "completeEmptySpace()" method takes an optional input img_S, which is the source test imagery. If img_S is provided, ROV-Spatial will be used, and ROV-Reflection otherwise. Fig. 9 shows the high-level execution trace step by step from line 9 to line 18 in Alg. 1. The example uses only three angles for simplicity. The tables in the figure shows the projection results for one rectangle (red).

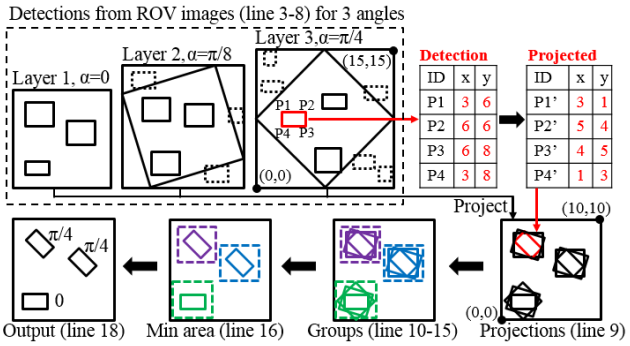


Figure 9: Execution steps of Algorithm 1 (best in color).

Algorithm 1: Rotation-vector based method

Require: (1) A test image img; (2) A rotation vector $V = [\alpha_1, ..., \alpha_n]$; (3) (optional) Source imagery img_S;

1: L_{temp} = []
2: L_{out} = []
3: **for** α in V **do**
4: img_α = rotate(img, α)
5: img'_α = completeEmptySpace(img_α, α, optional: img_S)
6: $mobrList_\alpha$ = deepCNN(img'_α)
7: L_{temp}.addLayer($mobrList_\alpha$)
8: **end for**
9: L_{temp} = filter_and_project(L_{temp}, V)
10: **for** $rect$ in L_{temp}.getLayer(1) **do**
11: $group$ = [$rect$]
12: **for** $layer$ in L_{temp}.getLayer([2,len(V)]) **do**
13: $rect'$ = getNearest($rect$, $layer$)
14: $group$.addMember($rect'$)
15: **end for**
16: L_{out}.addObject($group$.getMBR())
17: **end for**
18: **return** L_{out}

3.2 Proposed Scheme 2: A Context-based Approach

Context data contains spatial context information of certain types of objects, which can be used to infer the rotation angles of the objects. For example, roads and topographic models can serve as the context of vehicles, buildings and farm fields/plots. The context-based approach uses available context data to augment detections from MOBR-based deep learning frameworks. It assumes that there exists certain contexts that be used to determine the direction of objects, and is not applicable otherwise.

Since appropriate contexts can differ across different types of objects, here we use buildings and roads as an example to illustrate the use of this context-based method.

Figure 10: Roads as a context for buildings.

As shown in Fig. 10, buildings are often built in the context of nearby roads and their directions are often parallel to the roads, even in scenarios where the roads are curved. While this is a well established pattern, there is a few cases where buildings may not follow it. Fig. 10 shows two anomalies in the orange dashed-boxes: (1) buildings that are far away from roads; (2) buildings around the end of roads. In those cases, we do not rotate the detections.

Alg. 2 shows the general process of the context-based approach. The third input β is the radian of the angle between the direction of the context and the object. For example, $\beta = \pi/4$ or $\pi/3$ for vehicle-parking lot (one way) by design standards, and $\beta = 0$ for road-building. In Line 4 of Alg. 2, the function "getContextItem()" needs to be defined based on specific context-object relationship. For example, for road-building, the function returns the nearest road segment of a building. For road-vehicle, it is the intersecting road segment. Note that the context based approach only adjusts the angles of the detections but does not re-estimate their areas (only have detections from the original non-rotated image), which is a difference to ROV. Fig. 11 shows the high-level execution steps of Alg. 2 (lines 3 to 10) through a building-road example.

Algorithm 2: Context-based method

Require:
(1) A test image img;
(2) A list of context objects L_{ctxt};
(3) A context-object angle β;
1: L_{result} = **new** List()
2: L_{mobr} = deepCNN(img)
3: **for** $MOBR$ in L_{mobr} **do**
4: c = getContextItem($MOBR$, L_{mobr})
5: α = getContextAngle(c)
6: α' = getRotationAngle(α, β)
7: $MOBR'$ = rotate(obj: $MOBR$, pivot: $MOBR.center$, angle: α')
8: L_{result}.append($MOBR'$)
9: **end for**
10: **return** L_{result}

Summary of data requirements: Within the Unsupervised Augmentation (UA) framework, there are different data requirements of solutions that may affect their usability in different application scenarios. Table 1 shows the data requirements for the ROV and the context based methods.

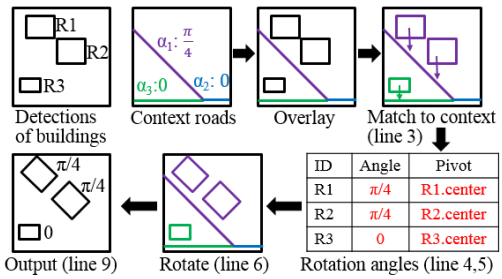


Figure 11: Execution steps of Algorithm 2 (best in color).

Table 1: Data requirements for the proposed methods

Requirements	Context	ROV-Reflection	ROV-Spatial
Access to context data	✓		
Access to source test data			✓
Access to plain test data	✓	✓	✓

4 ANALYTICAL VALIDATION

In this section, we formally validate the design decisions in the ROV scheme (Sec. 3.1) through theorems and mathematical proofs.

First, Thm. 1 shows that we only need to rotate an image at a single pixel (e.g., center for convenience) regardless of the locations and angles of objects as well as the rotation angle α.

Theorem 1. *An object in an image is rotated by α no matter which pixel is used as the pivot to rotate the image by α. In addition, the size of the resulting image is independent of the choice of the pivot.*

Proof. First, in Fig. 12, suppose A and B are two points on the boundary of an object where edge $\overline{AB}$ is parallel to (i.e., the same as) the direction of the object, and suppose the intersection P of the axes is the pivot of the rotation (the rotation angle is α in the counter-clock direction). L is a line passing through P and $\overline{AP} \perp L$. Edge $\overline{A'B'}$ is the rotated version of $\overline{AB}$, and L' is the rotated version of L. The dashed lines in Fig. 12(a) show the rotation from A to A' by α. Since P is the pivot point, the rotation from A to A' follows an arc on the circle with center P and radius r. In addition, since L' (passing P) is also rotated by α, we have $\overline{A'P} \perp L'$. Similarly, Fig. 12(b) shows that $\triangle PBC \cong PB'C'$ because $PB = PB'$, $\angle BPC = \angle B'PC'$ and $\angle BCP = \angle B'C'P$. Thus, we have $BC = B'C' = u$ and $CP = C'P' = v$. The angle β between $\overline{AB}$ and L is then equal to the angle β' between $\overline{A'B'}$ and L' because $\beta = \tan^{-1}\frac{u-r}{v} = \beta'$. Thus, the angle between $\overline{AB}$ and $\overline{A'B'}$ is equal to the angle between L and L', which is α. Then, we show that the size of the rotated image does not depend on the choice of pivot pixel. As an array, an image always has a rectangular shape in a two-dimensional space (i.e., height×width). Here we ignore the dimension of its depth since it is not relevant to the rotations. If we consider the input image as a rectangle R_{img}, the size of the rotated image is defined by the MOBR of the rotated R_{img} to minimize the empty areas. Denote the side length of R_{img} as W (square input). No matter which pixel is used as the pivot, the resulting side length is always $W \cdot (|\sin\alpha| + |\cos\alpha|)$. □

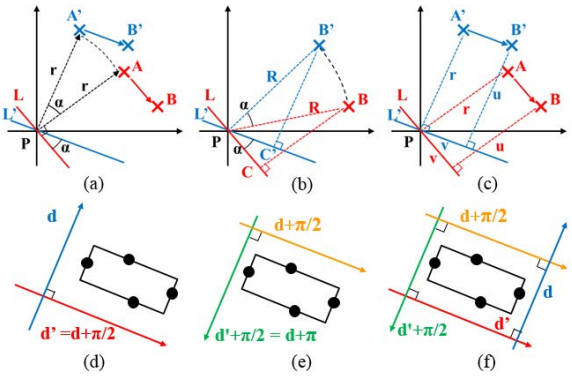


Figure 12: Rotations of objects.

Through Lemma 1 and Thm. 2, we further show that it is sufficient to only use a set of $\alpha \in [0, \pi/2)$ instead of $\alpha \in [0, 2\pi)$. The radian π is equivalent to 180°. In the following we denote MBR_α as the MBR of an object at a specific angle α. For example, MBR_0 ($\alpha = 0$) is the MOBR.

Lemma 1. *For an object, its MBR_α is the same as its $MBR_{\alpha+\frac{\pi}{2}}$.*

Proof. Fig. 12(d) shows the MBR_α of an object, where the axes are also shown in α directions. The black points denote the vertices of the bounded object that lie on the boundary of MBR_α. First, if we reset d and d' as the orthogonal directions in this two dimensional space, then MBR_α becomes the MOBR. Similarly, if we reset $d + \pi/2$ and $d' + \pi/2$ as the orthogonal directions, then $MBR_{\alpha+\pi/2}$ becomes the MOBR (Fig. 12(e)). By definition, any two orthogonal directions differ by $\pi/2$ radian in a two-dimensional space. Thus, an MOBR must remain orthogonal (and minimum by definition) if the orthogonal directions are rotated by $\pi/2$ (Fig. 12(f)). Thus, the two MOBRs must be the same $\Rightarrow MBR_\alpha = MBR_{\alpha+\frac{\pi}{2}}$. □

Theorem 2. *For two rotation vectors $\boldsymbol{u} = \{\boldsymbol{u}_i \mid \boldsymbol{u}_i \in [0, \pi/2), \forall i \in \{1, 2, ..., m\}\}$ and $\boldsymbol{v} = \{\boldsymbol{v}_i \mid \boldsymbol{v}_{(i+z\cdot m)} = \boldsymbol{u}_i + z \cdot \frac{\pi}{2}, \forall z \in \{0, 1, 2, 3\}, i \in \{1, 2, ..., m\}\}$, the set of distinct rotation angles covered by them ($\boldsymbol{u}$ and $\boldsymbol{v}$) is the same.*

Proof. Based on Lemma 1, we know that the MBR_α of an object has the same direction as $MBR_{\alpha+\frac{\pi}{2}}$. Thus, having an angle $\mathbf{v}_i = \mathbf{u}_i + \pi/2$ does not generate an MBR of a distinct direction (i.e., rotation angle) compared to the original $\mathbf{u}_i$. With this, it is straight forward to show the same result for other integer multipliers z (i.e., $\mathbf{v}_i = \mathbf{u}_i + z \cdot \pi/2, \forall z \in 0, 1, 2, 3$) using induction. □

Thm. 2 shows that the rotation angles in a rotation vector should be listed in the range $[0, \pi/2)$ to avoid redundancy in angle enumeration.

5 EXPERIMENTAL EVALUATIONS

Fig. 13 shows the general design of the experiments, where the data and methods are discussed in detail in Sec. 5.1. The design aims to answer the following five evaluation questions (First three: sensitivity analysis; Last two: comparative analysis):

- Do empty areas in rotated images affect solution quality?
- Does ROV-Tuning reduce the effect of empty areas?
- Do the completion algorithms (i.e., ROV-Reflection, ROV-Spatial) reduce the effect of empty areas?
- Do the proposed ROV and context based approaches improve accuracy on angle estimation?
- Do the proposed ROV and context based approaches improve accuracy on area estimation (i.e., tightening the rectangular bounds on objects)?

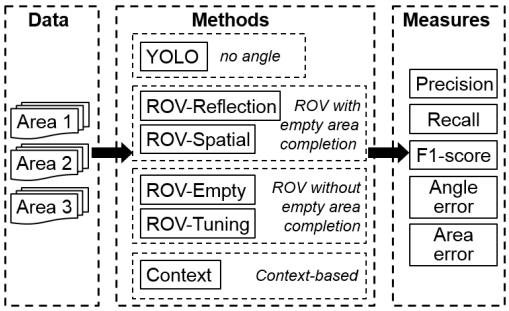


Figure 13: Experiment design.

5.1 Experiment Setup

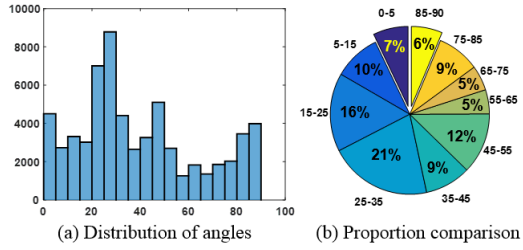
5.1.1 Candidate Approaches. Since the focus of this paper is on unsupervised augmentation (i.e., no need for ground-truth MBRs) of deep learning frameworks, which has not been well-studied, our candidate approaches mainly considered the two ROV methods (i.e., ROV-Reflection to ROV-Spatial) and the context-based approach. We also added the original YOLO framework as a baseline to show the improvements on object angle and area estimation compared to pure MOBR detections. In addition, to show the effects of empty areas in rotated images (Sec. 3.1), we also added the following two methods:

ROV-Empty: This method directly work with rotated test images without any completion of empty areas;

ROV-Tuning: We fine-tune a trained model using images with artificially-inserted empty areas, which cover the boundary regions of the training images. The ground-truth building footprints in the inserted empty areas are removed for correctness. This method requires fine-tuning of an existing model as well as access to its training dataset. The goal is to evaluate if this fine-tuning helps mitigate the empty-area issue or causes more confusion in training.

In summary, the candidate approaches in the evaluation are: (1) YOLO (orthogonal); (2) ROV-Empty; (3) ROV-Tuning; (4) ROV-Reflection; (5) ROV-Spatial; and (5) the context-based approach (Context). Note that all candidate methods (except ROV-Tuning) used exactly the **same set of trained weights** for fair comparisons.

5.1.2 Dataset. In order to quantitatively evaluate the accuracies on object angle and area estimation of different candidate approaches, we need to select a geospatial object for which there exists such a dataset that we can use to generate MBRs for evaluation purposes (still trained with MOBR). Datasets that meet this criterion for general objects are very difficult to find in our exploration. Thus, we chose building as an example in the experiments, and used the free



(a) Distribution of angles (b) Proportion comparison

Figure 14: Distribution of building angles.

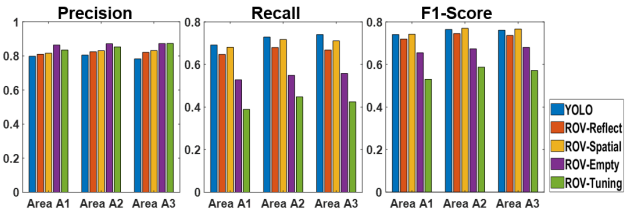


Figure 15: Sensitivity analysis in test areas. (best in color)

and publicly available Massachusetts Buildings Dataset (digitized Esri shapefile) in [1, 10]. The imagery data we used were standard county mosaics from the National Agricultural Imagery Program (NAIP), which is also freely available. The resolution of the NAIP imagery is one-meter. We used about 127,282 building footprints in the dataset for training the MOBR-based YOLO framework, and 51,326 for testing. The NAIP imagery was split into 1022 test images of size 208×208 to feed into the YOLO framework. The total area of the test region is about 45 km^2. In order to show possible variations of result quality in different sub-areas of the test region, we split the test region into three contiguous areas (i.e., Area A1-A3) to evaluate and present the statistics. The number of test images in the three areas are 340, 341 and 341, and the number of buildings are 17376, 18954 and 14996, respectively. We also adjusted the layer architecture of YOLO to improve the detection accuracy for buildings. Since layer design is not the focus of this paper, we do not describe it in detail. All the candidate methods (except ROV-tuning) share the same trained weights. To facilitate the training process, we used transfer learning with pre-trained weights from the PASCAL VOC dataset [12]. The training and prediction related operations with YOLO were performed using a NVIDIA Tesla K40 GPU, and the YOLO framework was implemented using TensorFlow.

5.2 Experimental Results

Fig. 14 shows the distribution of building angles in the entire test region. As we can see, the angles of buildings are heavily mixed and only about 13% of buildings align well with the orthogonal direction (i.e., $0^\circ - 5^\circ$: 7% and $85^\circ - 90^\circ$: 6%). This indicates that the orthogonal direction is not a good representation for the building footprints in this region. The angle distribution can also lead to overestimation of object areas in most of the test region if pure MOBRs are used. For example, the overestimation of areas could reach above 100% for buildings with a 45° angle.

Table 2: Precision, recall and F1 scores in Areas A1 to A3

Area ID-Metric	YOLO	ROV-reflect.	ROV-spatial	ROV-empty	ROV-tuning
A1-Precision	79.8%	81.0%	81.6%	86.4%	83.4%
A1-Recall	69.1%	64.7%	68.1%	52.8%	38.9%
A1-F1 score	74.1%	72.0%	74.2%	65.5%	53.0%
A2-Precision	80.5%	82.4%	83.0%	87.1%	85.3%
A2-Recall	72.8%	68.0%	71.7%	54.9%	44.8%
A2-F1 score	76.5%	74.5%	77.0%	67.4%	58.7%
A3-Precision	78.3%	82.1%	83.2%	87.3%	87.3%
A3-Recall	74.1%	66.8%	71.1%	55.8%	42.5%
A3-F1 score	76.1%	73.6%	76.6%	68.0%	57.1%

5.2.1 Sensitivity analysis. First we conducted sensitivity analysis to assess the effect of empty spaces in rotated images on the detection accuracy of ROV methods. Here we included the results of YOLO (MOBR version) as a baseline to measure the effect since it does use any rotation and is not affected by the empty areas. For the context based approach, its precision, recall and F1-scores are mostly the same as those of YOLO (e.g., less than 1% difference).

The effect of empty areas in rotated images on solution quality: Table 2 shows the precision, recall and F1 scores of the candidate approaches in the three test areas. The statistics are also visualized in Fig. 15. As we can see, the recall (i.e., $\frac{|detections \cap true|}{|true|}$) of ROV-Empty is only 50% to 55% in the Areas A1 to A3, which is 10%-20% lower than that of YOLO (i.e., no empty areas). This result is consistent with our analysis in Sec. 3.1, that says empty/zero-valued pixels could lead to low activation values (e.g., ReLU) in deep network layers, and reduce the probability score on objects. This effect is particularly large around the borders between the image and the empty areas (Fig. 4). Other than the effects on recall, the empty areas did not have much impact on precision (i.e., $\frac{|detections \cap true|}{|detections|}$). Its F1-scores (i.e., $\frac{2}{(precision^{-1}+recall^{-1})}$ are lower mainly due to the lower recalls.

The performance of artificial fine-tuning (ROV-Tuning) on reducing the effect of empty areas: Table 2 shows that ROV-Tuning has consistently lower precision, recall and F1 scores than those of ROV-Empty in the three test areas. This means that fine-tuning not only did not improve the performance but made it even worse (i.e., about a 10% drop for both the recall and F1 scores). The reason might be that: (1) it is difficult to offset the effects of large chunks of zero pixels; (2) the kernels which were learned to reduce such effects along the border may have hurt the predictions at places which have no empty areas around (i.e., introducing confusion into the training). This shows that artificial fine-tuning could not help improve the accuracies based on our experiments.

The performance of the proposed completion algorithms (i.e., ROV-Reflection, ROV-Spatial) on reducing the effect of empty areas: Compared to ROV-Empty, ROV-Reflection and ROV-Spatial do have on average 10% to 15% increases in both recall and F1-scores. The F1 scores are also at the same level as or higher than those of YOLO. This indicates that the completion algorithms are able to mitigate the effects of empty areas and improve the

Table 3: Errors of angles and areas in Areas A1 to A3

Area ID-Metric	YOLO	ROV-reflect.	ROV-spatial	ROV-empty	ROV-tuning	Context
A1-E_r	26.5°	9.9°	9.1°	11.0°	7.3°	3.5°
A1-E_a	57.2%	29.8%	29.0%	33.2%	32.0%	57.2%
A2-E_r	22.8°	9.2°	8.6°	10.7°	7.3°	3.6°
A2-E_a	48.2%	26.3%	25.5%	28.9%	28.2%	47.8%
A3-E_r	19.0°	10.2°	9.8°	10.9°	9.3°	4.7°
A3-E_a	44.6%	27.2%	26.8%	29.2%	30.0%	44.4%

*Notation: E_a = Error of area (%), E_r = Error of rotation angle (°)

detection rate of objects around the borders. ROV-Spatial performed consistently better than ROV-Reflection with small margins, which shows that the high fidelity of true spatial fillings do help improve the solution quality. On the other hand, ROV-Spatial requires access to source test imagery in order to be usable, whereas ROV-Reflection can work directly on plain test images. Thus, there is a trade-off between the solution quality and generality.

5.2.2 Comparative analysis. We evaluated the improvements on angle and area estimation achieved by the proposed UA framework by comparing to the baseline YOLO.

The performance of the proposed ROV and context based methods on improving the accuracy of angle estimation: Table 3 shows the errors for angles E_r (i.e., difference from the true angles) in the three test areas. The statistics are also visualized in Fig. 16. Since the default YOLO framework models objects with MOBRs, it cannot estimate the angles of objects well when they are not aligned with the orthogonal directions of a test image. Thus, YOLO had high errors of angles (i.e., about 20° to 25°) in the test areas. ROV-Reflection and ROV-Spatial were able to reduce the errors to about 9°. Here the ROV methods used a rotation vector with 8 angles, so the distance between nearest angles was 11.25°. If a true angle is right at the middle of two nearby angles in the rotation vector, then it will incur an error of at least $11.25°/2 = 5.625°$. This may partially contribute to the 5° error gap between ROV and the context based method. In the three test areas, the context-based method was able to reduce the errors to 3° to 4°, achieving the best performance. Fig. 17 (top) shows the distribution of angle errors across 100 test images, where the X and Y axes represent image IDs and angle errors, respectively. To avoid too much overlap in the visualization, we plotted the errors of three representative methods: YOLO, ROV-Spatial and the context based approach. As we can see, the error reductions from YOLO to both ROV and Context were significant, and Context was slightly better than ROV.

The performance of the proposed ROV and context based approaches on area estimation: In Table 3, the area error E_a for each detection was evaluated as $\frac{|a-a_{true}|}{a_{true}}$, where a and a_{true} are the areas of the detected object and the true object, respectively. As we can see, the ROV methods were able to reduce the area errors from on average 45%-55% to on average 25%-30%. To better see the error, for a residential house of size $10\,m \times 10\,m$, a 25% difference corresponds to a size of about $11.2\,m \times 11.2\,m$ or $8.7\,m \times 8.7\,m$. However, since the context based approach only focuses on

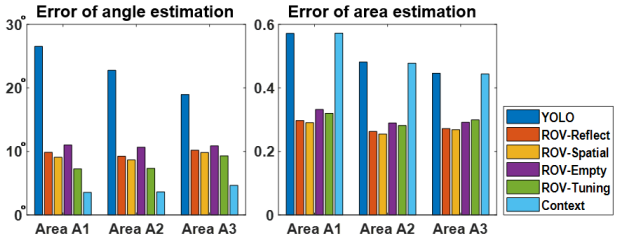


Figure 16: Comparative analysis in test areas. (best in color)

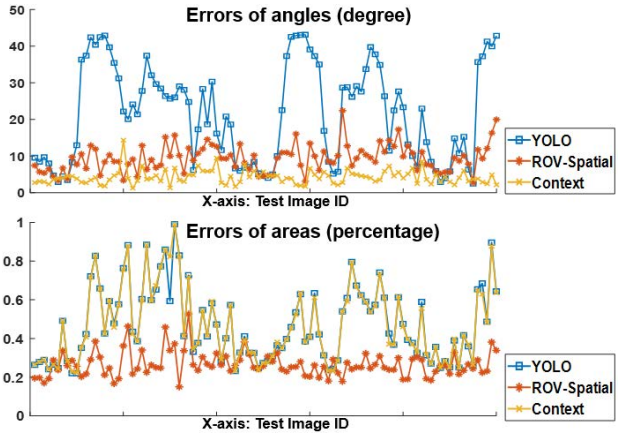


Figure 17: Errors in 100 test image samples. (best in color)

the angles, it was not able to reduce the errors on areas in the experiments. Fig. 17 (bottom) shows the distribution of area errors across 100 test images. ROV-Spatial showed better performance over both YOLO and the context-based approach.

Visualization: Besides statistics, we also compared the solution quality of the six candidate approaches through map visualizations in Fig. 18. The two rows in the figure correspond to two test images, and the columns represent different methods. As we can see, the results of YOLO only contained MOBRs of buildings, which also created loose bounds on building areas and led to overlaps among detections (Fig. 18(a)). Note that such overlaps also caused missing detections in places where buildings are very dense, because YOLO employs overlap removal to avoid duplicated detections. In contrast, ROV-Reflection and ROV-Spatial (Fig. 18(b) and (c)) were able to capture the angles of building footprints and generate tighter rectangular bounds on the buildings. The results of ROV-Spatial were more complete compared to ROV-Reflection. In Fig. 18(d) and (e), we can see that ROV-Empty was affected by the empty areas and had fewer detections especially along the boundaries of the images. ROV-Tuning did not improve the solution quality and missed more buildings. Finally, the context based approach performed the best in terms of angle estimation. While it did not tighten the area bounds, the overlaps were reduced because of the corrected angles.

6 CONCLUSIONS AND FUTURE WORK

We proposed two unsupervised augmentation schemes, namely a ROtation-Vector (ROV) based scheme and a context based scheme,

Figure 18: Visualization of detection results in two test images. (best in color)

to detect MBRs of geospatial objects without the need for: (1) additional training dataset with angle information; (2) redesign of network architecture; and (3) re-training. Within the ROV scheme, we also developed two completion algorithms, namely ROV-Reflection and ROV-Spatial, to mitigate the issue of missing detections caused by empty areas in rotated images. Through experiments, we showed that both the ROV and the context based schemes can estimate the rotation angles with high accuracy. While the context based scheme performed better on angle prediction, it could not tighten the estimations of objects' areas. The ROV scheme was able to significantly improve the accuracy of area estimation.

Future work: One limitation of the current ROV scheme is that it cannot cover all possible angles due to the discrete angle representation in the rotation vector. In future work, we aim to explore new techniques to further refine the angles without introducing much computational overhead (e.g., a very high-resolution rotation vector). While this work mainly considers the solution qualities, we will investigate computational refinements to improve the efficiency. In addition, since ROV performed better on area estimation and the context based method was better on angle estimation, we will explore an integrated approach of the two to further improve the solution quality. We will also investigate integrations with other unsupervised techniques (e.g., Hough transform) and experiment with more deep networks.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants No. 1737633, 1541876, 1029711, IIS-1320580, 0940818 and IIS-1218168, the USDOD under Grants No. HM1582-08-1-0017 and HM0210-13-1-0005, ARPA-E under Grant No. DE-AR0000795, USDA under Grant No. 2017-51181-27222, NIH under Grant No. UL1 TR002494, KL2 TR002492 and TL1 TR002493 and the OVPR Infrastructure Investment Initiative and Minnesota Supercomputing Institute (MSI) at the University of Minnesota. We also would like to thank Kim Koffolt and the spatial computing research group for their helpful comments and refinement.

REFERENCES

[1] 2013. Massachusetts Buildings Dataset. https://www.cs.toronto.edu/ vmnih/data/.
[2] Raphaël Compagnon. 2004. Solar and daylight availability in the urban fabric. *Energy and buildings* 36, 4 (2004), 321–328.
[3] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.
[4] Kirti Kusum Joshi and Tatsuhito Kono. 2009. Optimization of floor area ratio regulation in a growing city. *Regional Science and Urban Economics* 39, 4 (2009), 502–511.
[5] Tetsu Kubota, Masao Miura, Yoshihide Tominaga, and Akashi Mochida. 2008. Wind tunnel tests on the relationship between building density and pedestrian-level wind velocity: Development of guidelines for realizing acceptable wind environment in residential neighborhoods. *Building and Environment* 43, 10 (2008), 1699–1708.
[6] Nataliia Kussul, Mykola Lavreniuk, Sergii Skakun, and Andrii Shelestov. 2017. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters* 14, 5 (2017), 778–782.
[7] Lei Liu, Zongxu Pan, and Bin Lei. 2017. Learning a Rotation Invariant Detector with Rotatable Bounding Box. *arXiv preprint arXiv:1711.09405* (2017).
[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *European conference on computer vision*. Springer, 21–37.
[9] Zikun Liu, Jingao Hu, Lubin Weng, and Yiping Yang. 2017. Rotated region based CNN for ship detection. In *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 900–904.
[10] Volodymyr Mnih. 2013. *Machine Learning for Aerial Image Labeling*. Ph.D. Dissertation. University of Toronto.
[11] Keiller Nogueira, Otávio AB Penatti, and Jefersson A dos Santos. 2017. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition* 61 (2017), 539–556.
[12] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 6517–6525.
[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.
[14] Adriana Romero, Carlo Gatta, and Gustau Camps-Valls. 2016. Unsupervised deep feature extraction for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing* 54, 3 (2016), 1349–1362.
[15] M Rylatt, S Gadsden, and K Lomas. 2001. GIS-based decision support for solar energy planning in urban environments. *Computers, Environment and Urban Systems* 25, 6 (2001), 579–603.
[16] Hoo-Chang Shin, Holger R Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M Summers. 2016. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging* 35, 5 (2016), 1285–1298.
[17] Shuo-sheng Wu, Le Wang, and Xiaomin Qiu. 2008. Incorporating GIS building data and census housing statistics for sub-block-level population estimation. *The Professional Geographer* 60, 1 (2008), 121–135.