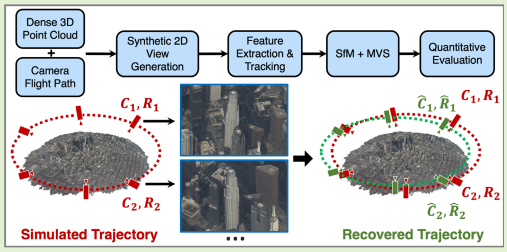
Local Feature Performance Evaluation for Structure-From-Motion and Multi-View Stereo Using Simulated City-Scale Aerial Imagery

Ke Gao, *Graduate Student Member, IEEE*, Hadi AliAkbarpour, *Member, IEEE*, Joshua Fraser, Koundinya Nouduri, Filiz Bunyak, Ricky Massaro, Guna Seetharaman, *Fellow, IEEE*, and Kannappan Palaniappan, *Senior Member, IEEE*

***Abstract*—Ubiquitous low cost multi-rotor and fixed wing drones or unmanned aerial vehicles (UAVs) have accelerated the need for reliable, robust, and scalable Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipelines suitable for a variety of flightpath trajectories especially in degraded environments. Feature tracking being a core part of SfM and MVS, is essential for multiview scene modeling and perception, but difficult to evaluate in large scale datasets due to the lack of sufficient ground-truth. For large-scale aerial imagery, accurate camera orientation and dense 3D point cloud accuracy can be used to assess the impact of accurate feature localization and track length. We propose a novel view simulation (or synthesis) framework which generates visually realistic new unseen camera views for feature detection using known high fidelity camera poses for modeling. Seven state-of-the-art local handcrafted and learning-based features are quantitatively evaluated for robustness and matchability within the SfM and MVS pipelines using the open source COLMAP software. Our experimental results provide performance rankings of each feature, using twelve different evaluation metrics across three synthetic city-wide aerial image sequences. We show that recent learned features, SuperPoint and LF-Net, have not only reached the quality of the best handcrafted features like SIFT and SURF, but now outperform them in terms of more accurate 3D camera pose estimates and longer feature tracks. SuperPoint produces 1.51 meter average position error and 0.03° average angular error, while SIFT remains competitive (second best for pose and overall) with 1.78 meter and 0.11° errors respectively.**



***Index Terms*—Local feature detector, SIFT, SURF, ORB, AKAZE binary feature, MSER, LF-Net, SuperPoint, image matching, photogrammetry, camera pose, bundle adjustment, simulation environments, virtual reality, augmented reality.**

I. Introduction

EXTRACTING features and establishing tracks of feature correspondences along a video sequence are critical steps in Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipelines [1]. Many feature extraction methods are exclusively designed for a particular computer vision task [2], so their performance can be substantially influenced by different types of input data. As a result, it is important to find feature extraction approaches that provide robust results across different scenarios. SfM is widely used in many applications including 3D reconstruction, autonomous driving, robotics, augmented reality, 3D scanning, *etc*. Inputs to a SfM system are raw images and the final outputs are a sparse 3D point cloud and estimated camera 3D poses. The first main process performed on the input images is to apply local feature detectors to identify potentially distinguishable image points. Once the feature points are detected on each image, a feature descriptor is used to assign a unique identity to each feature. The feature descriptors from different views are compared against each other so that feature correspondences or matches are established. Such comparison takes place between multiple views and those features that are matched together create a group which is known as feature track. From the geometry point of view, feature points within a track represent

Manuscript received July 31, 2020; revised November 12, 2020; accepted November 15, 2020. Date of publication December 7, 2020; date of current version April 16, 2021. This work was supported in part by the U.S. Army Research Laboratory under Grant W911NF-1820285 and in part by the Army Research Office DURIP under Grant W911NF-1910181. The associate editor coordinating the review of this article and approving it for publication was Dr. Julio C. Rodriguez-Quiñonez. *(Corresponding author: Ke Gao.)*

Ke Gao, Hadi AliAkbarpour, Joshua Fraser, Koundinya Nouduri, Filiz Bunyak, and Kannappan Palaniappan are with the Computational Imaging and VisAnalysis Lab, Electrical Engineering and Computer Science Department, University of Missouri, Columbia, MO 65211 USA (e-mail: kegao@mail.missouri.edu; akbarpour@missouri.edu; fraserj@mail.missouri.edu; knfdt@mail.missouri.edu; bunyak@missouri.edu; pal@missouri.edu).

Ricky Massaro is with the U.S. Army Engineer Research and Development Center Geospatial Research Laboratory, Alexandria, VA 22315 USA (e-mail: ricky.d.massaro@erdc.dren.mil).

Guna Seetharaman is with Advanced Computing Concepts, U.S. Naval Research Laboratory, Washington, DC 20375 USA (e-mail: guna.seetharaman@nrl.navy.mil).

Digital Object Identifier 10.1109/JSEN.2020.3042810

the images (projections) of an ideally unique 3D point from the scene. At this stage, most SfM approaches use the 2D point correspondences to estimate initial poses corresponding to each image view. There are multiple approaches that are used in the literature to estimate camera poses. Nister's *5-point* [3] and Longuet-Higgins' *8-point* [4] are the two most common algorithms for this purpose that estimate the essential and fundamental matrices, respectively. Once the feature tracks are built, and assuming that the camera poses are available, the feature points within each track are triangulated (cast) into 3D as rays. Ideally, these rays must converge into a single 3D point. However, in real scenarios this is not the case due to existence of error in both feature localization and inaccuracy of camera poses. There are both linear and non-linear methods for triangulation so that an optimal 3D point is estimated for the observations (feature points) within a track.

The triangulated tracks constitute a sparse 3D point cloud. At this stage of the MVS pipeline, both the camera poses and sparse 3D points are initialized. In real scenarios, however, these initial values are always prohibitively noisy and too inaccurate to be used in a downstream processes such as dense 3D reconstruction. Therefore one has to refine them by applying an optimization technique. Bundle Adjustment (BA) is a well known approach in computer vision literature which takes the initial values of the camera poses along with the sparse 3D points obtained from the triangulation stage and iteratively performs a non-linear least squares minimization to produce a refined solution that satisfies the reprojection error cost function [5], [8]. At this point, the geometry of cameras are known and a set of sparse 3D points representing the underlying scene are available. Henceforth, this information can be potentially exploited by many downstream applications such as dense 3D reconstruction [6], moving object tracking [9], image registration [10], [11], video analytics [12], *etc*. As also stressed in [13], the outputs of such downstream applications are only as good as the quality of the refined camera poses and the input images. The quality of input images mostly depends on the hardware technology. Recently, we have been witnessing rapid advances in the availability of high-resolution aerial imagery due to improvements in data acquisition hardware and the abundant availability of low-cost unmanned aerial vehicles (UAVs). This has increased the demands for reliable, robust and scalable SfM methods which can thoroughly exploit the available information in the high-resolution imagery.

Therefore it is crucial to evaluate the performance of different local features in the context of SfM and MVS, particularly on city-scale wide area motion imagery (WAMI) datasets [14], [15]. Performing a reliable evaluation of the estimated camera poses within a SfM pipeline has been impeded by a lack of accurate ground-truth on city-scale WAMI datasets. Instead, most of existing approaches have focused their evaluation either on the components of a SfM pipeline which do not require ground-truth such as reprojection error and population of sparse and dense 3D points, or on some toy-scale and non-realistic datasets [16]. To resolve this problem, we propose a visually realistic synthetic WAMI dataset generation framework that provides high fidelity ground-truth for the camera geometry. The developed large-scale synthetic datasets enable us to systematically study the effects of local features on the precision of the recovered camera 3D poses in addition to the common indirect error metrics. We evaluate a group of popular handcrafted features (both floating-point and binary) and recently developed deep learning-based features in a SfM and MVS pipeline, *i.e*. COLMAP [5]–[7], on our proposed MU synthetic WAMI dataset and the public Digital Imaging and Remote Sensing Image Generation (DIRSIG) dataset [17] from Rochester Institute of Technology (RIT). The proposed pipeline for local feature evaluation is illustrated in Figure 1. Our experimental results indicate interesting observations on the reliability and suitability of various handcrafted and learned features for different purposes. We provide individual ranking of features for different categories of error metrics including reprojection error, number of sparse 3D points, number of inliers, and camera pose (position, rotation) errors.

II. Related Work

The requirement for robust local features has led to many quantitative evaluation works on feature extraction methods. Moreels and Perona [18] evaluated various combinations of feature detectors and descriptors. Hessian-affine detector combined with SIFT descriptor performs the best under perspective distortions and illumination variances. Mikolajczyk and Schmid [19] evaluated a group of feature descriptors on the Oxford Affine Covariant Regions Dataset (ACRD) [20] which presents photometric and geometric transformations in 8 image sets. The performance of descriptors varies using different feature detectors and the most widely-used SIFT descriptor outperforms the other methods. Heinly *et al*. [21] compared three binary feature descriptors, BRIEF [22], BRISK [23], and ORB [24] with floating-point descriptors SIFT and SURF on an extended Oxford Dataset. The experiments indicated that binary descriptors achieve significant speed improvement but their performances vary according to image transformation types. SIFT descriptor yields the best results particularly for geometric transformations. Fan *et al*. [16] provided comparative analysis of handcrafted features and learning based features for image-based 3D reconstruction application on the DTU MVS dataset [25] that contains different scenes including various small objects and surface materials, and a large scale Structure-from-Motion dataset [26] that contains many unordered Internet images of several landmarks collected by ground cameras. Their evaluation results indicate that binary features produce acceptable 3D reconstruction results on the datasets with no distracting images. However, traditional floating-point handcrafted features like SIFT still provide better results than their binary counterparts for large scale datasets with distracting images. The learned descriptors in recent years outperform the handcrafted methods but the pioneering deep learning-based descriptor DeepDesc [27] provides less competitive performance. Schönberger *et al*. [7] presented extensive experimental evaluation of both handcrafted and learned local feature descriptors for applications of SfM and MVS on a group of small and large scale benchmark datasets. Their evaluation verified that the learned descriptors outperform

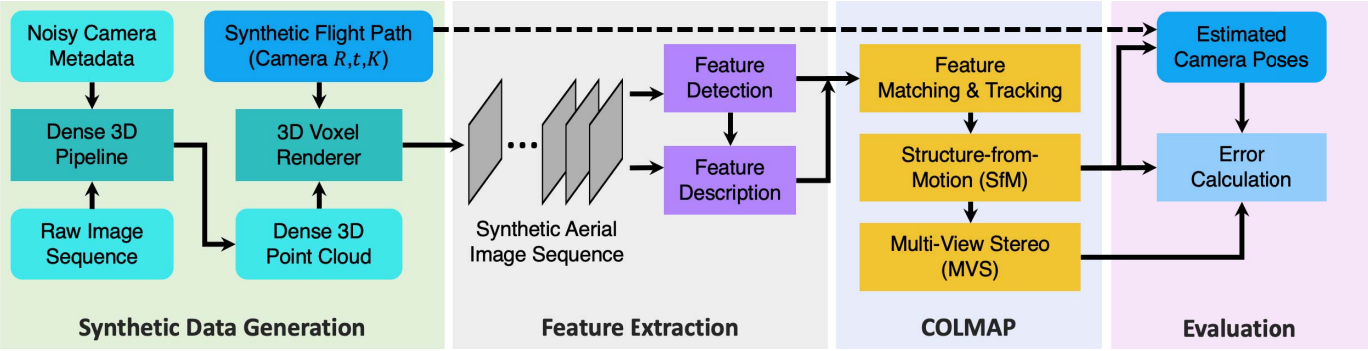


Fig. 1. Our feature evaluation pipeline consists of four components: (i) Dense 3D point cloud generation using AgiSoft commercial software with our own 3D voxel rendering method for synthetic aerial imagery generation, (ii) feature extraction using different methods, (iii) feature association and image-based 3D reconstruction (SfM and MVS) using COLMAP [5], [6], and (iv) quantitative evaluation. We use Matlab scripts provided by COLMAP to interface between feature extraction and 3D reconstruction, and Python scripts for evaluation [7].

SIFT in general, but advanced handcrafted descriptors [28], [29] yield comparable or superior results compared to the state-of-the-art learned features particularly in complex SfM scenarios. To address the lack of reliable training data for deep learning-based methods, Mueller *et al*. [30] proposed an image rendering and translation framework to generate high-quality synthetic street-view images for both 2D and 3D computer vision tasks. Nilosek *et al*. [17] developed a synthetic aerial image dataset of a suburban scene for the purpose of evaluating a variety of 3D reconstruction approaches in this scenario, but the synthetic data was generated from a hand-made 3D model which is not visually realistic. Özdemir *et al*. [31] introduced 3DOMCity, a scale model photogrammetric contest benchmark for evaluating image-based tasks such as 3D reconstruction and point cloud classification. The dataset is derived from a physically constructed 3D scale model of a city simulating an urban scene at fine detail using calibrated and precisely positioned cameras. Stathopoulou *et al*. [32] evaluated three 3D reconstruction pipelines (COLMAP, OpenMVG, and AliceVision) and their combinations on different large scale datasets. Their experiments showed that SIFT outperforms AKAZE for feature extraction in these pipelines, and incremental bundle adjustment and patch-based MVS produce more accurate results.

III. Feature Detection and Description

In this section, we introduce the feature detectors and descriptors that are evaluated in this article. Both handcrafted and deep learning-based approaches are included.

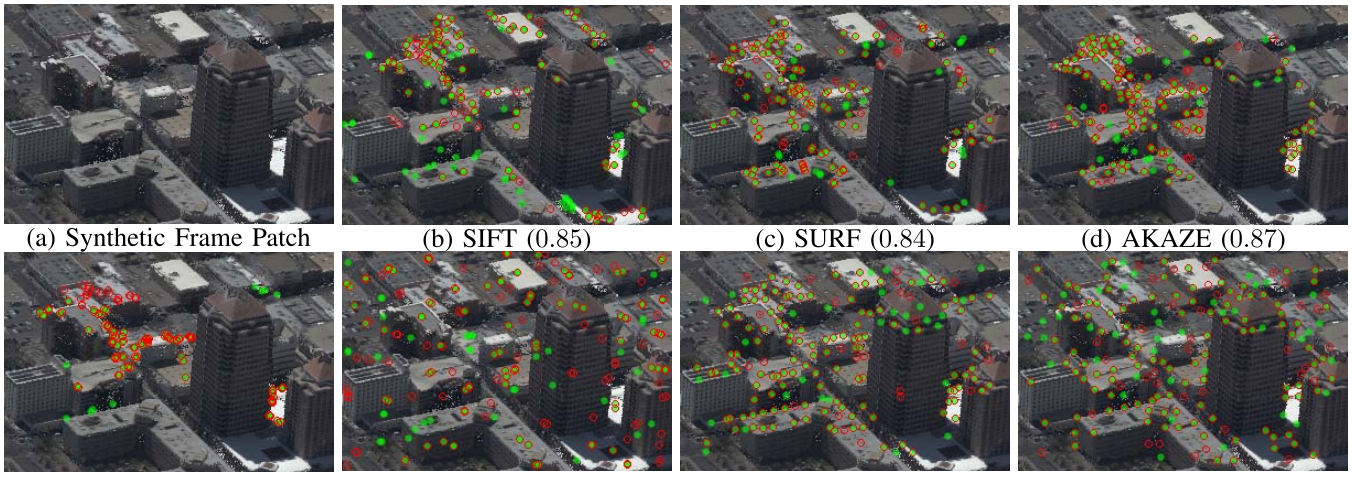
A. Feature Detectors

A feature detector extracts features with distinctive patterns from its local neighborhood in the image [33]. A reliable feature point detector is expected to consistently and accurately detect keypoints over long image sequences. Feature points detected on a sample image patch cropped from ABQ-215 synthetic aerial image by the feature detectors discussed below are shown in Figure 2.

- *SIFT*: Scale-Invariant Feature Transform (SIFT) [34] uses Difference of Gaussians DoG) for keypoints detection. Keypoints are extracted from local maxima/minima in the DoG at different scales. An orientation is then assigned to each keypoint using orientation histogram from a local region around the keypoint.
- *SURF*: Speeded Up Robust Features (SURF) [35] feature detector is developed based on SIFT but achieves higher computation efficiency. SURF approximates Laplacian of Gaussian using a box filter. SURF extracts keypoints from the determinant of the Hessian matrix.
- *AKAZE*: Accelerated KAZE [36] also uses the determinant of the Hessian matrix to detect keypoints. It uses nonlinear scale space based on the Fast Explicit Diffusion (FED) operator. Rotation invariance is achieved using Scharr filters.
- *ORB*: Oriented FAST and Rotated BRIEF (ORB) [24] utilizes the FAST corner detector [40] to extract keypoints from a scale pyramid. The intensity centroid of the local neighborhood around a keypoint estimates orientation.
- *MSER*: MSER [37] is a grayscale blob detector. It extracts maximally stable extremal regions (MSER) that are stable connected components. Pixels inside an extremal region should have approximately the same intensity through a broad range of thresholds. Deep learning feature descriptor DeepCompare uses MSER keypoints.
- *LF-Net*: Local Feature Network (LF-Net) [38] uses a fully convolutional network to detect feature keypoints. The scale-invariant feature points are detected from a final feature score map by performing softmax operation on N resized feature maps. The feature map for each scale is produced by a simple ResNet [41] architecture.
- *SuperPoint*: SuperPoint [39] uses a VGG-style [42] encoder to convert an input image to reduced dimensionality tensors. The tensor is upsampled by an interest point decoder to original size as a feature map. Pixel values in the map represent the likelihood of keypoint detections.

B. Feature Descriptors

To create a feature descriptor, a local region around a feature point is extracted and converted into a 1D array. A robust descriptor is expected to be invariant to a wide range of image transformations, such as translation, rotation, scale, illumination variance, image blur, perspective changes, *etc*. The studied feature descriptors are summarized in Table II.



(a) Synthetic Frame Patch (b) SIFT (0.85) (c) SURF (0.84) (d) AKAZE (0.87)

(e) ORB (0.80) (f) MSER/DeepCompare (0.65) (g) LF-Net (0.83) (h) SuperPoint (0.74)

Fig. 2. Feature point detection results for a sample image patch (300 × 200 pixels) cropped from ABQ-215 first frame (1650 × 1100) of synthetic aerial image sequence and the corresponding region of interest subsampled and cropped from the real aerial image with the same camera pose. Red hollow circles (synthetic) and green filled circles (real image) show the detected feature points at the same resolution with detectors applied to the patch and 150 keypoints detected using each method; red circles filled green are matches with a tolerance of 1 pixel in keypoint position. Dice score for just the patch is shown in parenthesis.

TABLE I
SUMMARY OF THE EVALUATED FEATURE DETECTORS. PARAMETERS OF EACH FEATURE DETECTOR FOR ABQ-215 DATASET INCLUDED

Feature Detector	Feature Type	Scale Invariant	Rotation Invariant	Subpixel Accuracy	Learned	Parameters
SIFT [34]	Blob	✓	✓	✓	✗	cv::xfeatures2d::SIFT::create(int nfeatures = 5000)
SURF [35]	Blob	✓	✓	✓	✗	cv::xfeatures2d::SURF::create(double hessianThreshold = 1290.00)
AKAZE [36]	Blob	✓	✓	✓	✗	cv::AKAZE::create(float threshold = 0.0015)
ORB [24]	Corner	✗	✓	✗	✗	cv::ORB::create(int nfeatures = 5000)
MSER [37]	Region	✓	✓	✓	✗	cv::MSER::create(int _min_area = 60, int _max_area = 14400)
LF-Net [38]	Corner & Blob	✓	✓	✓	✓	--top_k (number of keypoints) = 5000
SuperPoint [39]	Corner	✓	✓	✓	✓	--conf_thresh (detector confidence threshold) = 0.125

TABLE II
SUMMARY OF THE EVALUATED FEATURE DESCRIPTORS

Feature Descriptor	Dimension	Data Type	Learned	Training Set
SIFT [34]	128	float	✗	-
SURF [35]	64	float	✗	-
AKAZE [36]	61	byte	✗	-
ORB [24]	32	byte	✗	-
DeepCompare [43]	256	float	✓	Yosemite [44]
LF-Net [38]	256	float	✓	Outdoors [45], [46]
SuperPoint [39]	256	float	✓	MS-COCO [47]

- *SIFT*: The local neighborhood around a keypoint is divided into 16 sub-regions. An 8-bin histogram of each sub-region is computed using the orientation and gradient magnitude. Concatenating the histograms from all sub-regions creates the descriptor.
- *SURF*: 4 × 4 sub-regions are cropped from the local area around a keypoint. Wavelet responses are computed within each sub-region to form the descriptor.
- *AKAZE*: AKAZE is a binary descriptor. It proposes a Modified-Local Difference Binary (M-LDB) to perform binary tests using intensity and gradient of local region around a keypoint. The binary test results form the descriptor.
- *ORB*: ORB is also a binary descriptor. The binary tests are operated on the sampling pairs that have low correlation and high variance.
- *DeepCompare*: DeepCompare [43] employs a Siamese network to learn a similarity function for patches around keypoints extracted by MSER detector.
- *LF-Net*: Descriptor extraction is performed by a descriptor network given a cropped image patch around each keypoint. The network consists of three convolutional filters followed by fully-connected layers.
- *SuperPoint*: A descriptor encoder is used to upsample the intermediate tensor to the full size. A UCN-like [48] is utilized to generate a descriptor for every 8 pixels. Then bicubic interpolation is performed to create the complete set of descriptors.

IV. SYNTHETIC DATASET GENERATION

We use a pinhole camera model in which the homogeneous 2D point $\mathbf{x} = [x\ y\ 1]^\intercal$ represents the image of a homogeneous 3D point $\mathbf{X} = [X\ Y\ Z\ 1]^\intercal$ on the 2D camera focal plane with the projection from 3D to 2D defined as:

$$\mathbf{x} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{X} \tag{1}$$

where $\mathbf{R}_{3\times3}$ and $\mathbf{t}_{3\times1}$ are the rotation matrix and translation vector from the world coordinate system to local camera coordinates, respectively, and $\mathbf{K}_{3\times3}$ defines the camera calibration

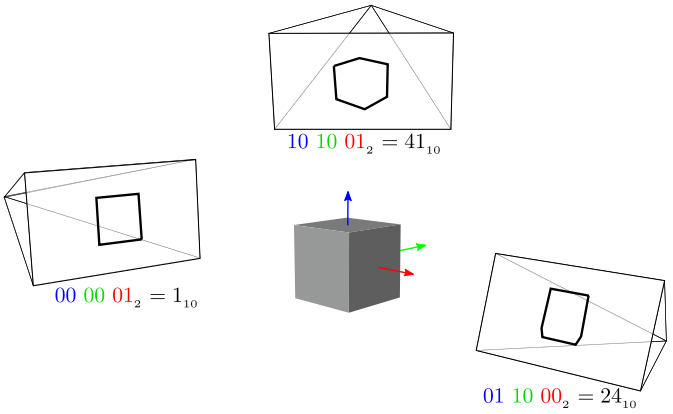


Fig. 3. Fast screen space area calculation for voxels. The convex hull for the gray-shaded 3D voxel cube is determined using a lookup table based on the hull code calculated from voxel extrema and camera position. The projected image space convex voxel hulls and associated hull codes are shown for three sample camera views.

(intrinsic or interior orientation) matrix:

$$\mathbf{K} = \begin{bmatrix} f & 0 & u \\ 0 & f & v \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where f is the camera focal length in pixels, (u, v) is the principal point and the lens distortion is treated as being negligible or corrected in a separate step. The position of the i^{th} camera, $\mathbf{C}_i$ in 3D is given by:

$$\mathbf{C}_i = -\mathbf{R}_i^{\mathsf{T}} \mathbf{t}_i \tag{3}$$

We propose a 3D voxel renderer to generate synthetic aerial image sequences. The synthetic aerial imagery generation process utilizes a dense 3D point cloud of an urban area provided by 3D reconstruction tools like Pix4D [49] and AgiSoft [50]. Each point is rendered as a voxel represented by a cube with position and extents. Our voxel software renderer is designed to produce repeatable results independent of the precision of the graphics hardware and to be efficient for views where the rendered voxels will be approximately the size of a single pixel.

Algorithm 1 GETHULLCODE: Generates Hull Code Using Camera Position Compared to Voxel Minimum and Maximum Ranges

Input : $\mathbf{C}$, *voxel* /* `camera position Eq. 3, voxel extrema` */

Output : *hullcode*

1: $hullcode \leftarrow (\mathbf{C}_x < voxel_{min_x})$
$+ (\mathbf{C}_x > voxel_{max_x}) \ll 1$
$+ (\mathbf{C}_y < voxel_{min_y}) \ll 2$
$+ (\mathbf{C}_y > voxel_{max_y}) \ll 3$
$+ (\mathbf{C}_z < voxel_{min_z}) \ll 4$
$+ (\mathbf{C}_z > voxel_{max_z}) \ll 5$

2: **return** (*hullcode*)

The projected image or screen space area of each voxel is determined efficiently using the method by Schmalstieg and Tobler [51]. A hull code is constructed by setting bits to indicate the camera's position with respect to minimum and maximum extents of the voxel with two bits for each x, y, and z direction as shown by GETHULLCODE in Algorithm 1. This hull code maps into a lookup table of pre-computed vertex indices for the convex hull of a cube. Figure 3 shows three examples of camera positions, hull codes, and the resulting convex hulls. Each voxel vertex in the convex hull is projected to the image plane and the area in pixels A is calculated using,

$$A = \frac{1}{2} \sum_{i=1}^{n-1} x_i (y_{i+1} - y_{i-1}), \quad \text{with } i \bmod n \tag{4}$$

where x and y are the hull coordinates projected to the image plane using Eq. (1) and n is the number of vertices in the hull. While a projected voxel is larger than a single pixel, the voxel is subdivided and the renderer recurses. Only when a sub-voxel is at least as small as a single pixel is the fragment tested against the depth buffer and drawn to the color buffer if the depth test passes. Pseudocode for the voxel renderer is given in Algorithm 2. GETSUBVOXEL performs the voxel subdivision returning a new voxel that is an octant of the original. WRITEPIXEL projects the voxel center to the image plane, performs a depth test, and updates pixel depth and color. Figure 4 shows two synthesized views generated using our

(a) (b)

Fig. 4. Two synthetic views for ABQ point cloud rendered using the recursive voxel renderer described in Section IV. Both the orthographic (a) and low altitude (b) synthesized views are dissimilar from any camera poses used during 3D reconstruction.

renderer that are dissimilar from any of the camera poses used for 3D reconstruction.

Algorithm 2 RENDERVOXEL: Recursive Voxel-Based Subdivision Rendering

Input : *camera*, *buffer*, *voxel*
Output : *buffer* /* Updated depth and color pixel values */
area ← IMAGESPACEAREA (*camera*, *voxel*) /* Eq. 4 */
if *area* ≤ 1.0 **then**
  WRITEPIXEL (*camera*, *buffer*, *voxel*)
else
  for $i \leftarrow 1$ to 8 **do**
    subvoxel ← GETSUBVOXEL (*voxel*, *i*)
    RENDERVOXEL (*camera*, *buffer*, *subvoxel*)
  end for
end if
return (*buffer*)

V. FEATURE TRACKING, SFM, AND MVS

In this section, we discuss the methodology for establishing feature tracks or associations, triangulation, and bundle adjustment used in Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipelines like COLMAP.

A. Feature Tracking

Algorithm 3 DISTANCERATIOMATCHING: Matching Strategy to Establish Feature Tracks Using L_2 Distances in Feature Space

Input : Reference image feature descriptor for r^{th} feature point $v_r \in \mathcal{R}$; candidate feature descriptors in matching image $\{v_m^i \in \mathcal{M} \mid i = 1, 2, \ldots, N\}$; ratio threshold $\tau = 0.8$ as in [1], [7]
Output : Best match feature index i for v_r in $\mathcal{M}$
$v_m^{n_1} \leftarrow$ Find nearest neighbor, n_1, of v_r in $\mathcal{M}$
$v_m^{n_2} \leftarrow$ Find second nearest neighbor, n_2, of v_r in $\mathcal{M}$
$d_{n_1} \leftarrow$ DISTANCE $(v_m^{n_1}, v_r)$
$d_{n_2} \leftarrow$ DISTANCE $(v_m^{n_2}, v_r)$
if $d_{n_1}/d_{n_2} \geq \tau$ **then**
  /* No suitable matching feature found for v_r */
  $n_1 \leftarrow$ NULL
end if
return (n_1)

The performance of bundle adjustment (BA) and Structure-from-Motion (SfM) is greatly influenced by feature tracking accuracy over a long image sequence, since feature tracks may contain outliers due to feature association mismatches. The first step of feature tracking is to extract keypoints by applying a feature detector on a pair of images, *i.e.* reference image and matching image. A feature descriptor is then computed for each keypoint by mapping the local neighborhood structure around the keypoint in the image to a 1D array. After that, the feature matching module (see Algorithm 3), calculates the L_2 Euclidean feature distance between a descriptor in the reference image and each descriptor in the matching image, and then selects the best matching pair using (brute-force) search and distance ratio threshold with $\tau = 0.8$ as in the original SIFT implementation [34], COLMAP [7] and our DCTF feature descriptor (with $\tau = 0.7$) [1]. Feature correspondences are exhaustively computed between all pairs of images in the image sequence as discussed in COLMAP [5]–[7]. Feature matching in the COLMAP internal pipeline and for external feature evaluation use the same distance ratio criterion (Algorithm 3). Feature tracks are established by associating two-view feature correspondences. Note that the same $O(n^2)$ image matching strategy between n views is used for all features evaluated in this work. In our related paper we used faster $O(n)$ image matching between sequential (temporally) adjacent views [1], [10].

B. Triangulation

Ideally, all 2D points $\mathbf{x}_{j,i}$ within a track of matched features define the image coordinates of an identical 3D point $\mathbf{X}_j$ in the scene. In other words, if the 3D coordinates of a point $\mathbf{X}_j$ is known, then all its corresponding 2D image points can be computed by projecting $\mathbf{X}_j$ onto all views (camera) using (1). In MVS applications the coordinates of 3D point $\mathbf{X}_j$ is not available, however, one can estimate it by casting ray passing through the camera center and the image point in each camera. In a perfect model, all the cast rays must intersect at an identical point in 3D which would be equivalent to $\mathbf{X}$. However, in real scenarios this is not the case and therefore an optimal solution for $\mathbf{X}$ must be estimated. One such method is known as *triangulation*. The output of the triangulation stage is a sparse 3D point cloud.

C. LM-Based Optimization for Bundle Adjustment

As mentioned previously, the 3D points associated with the feature tracks are all estimated from the measurements (feature points) using the geometry of the respective cameras (views). The camera poses used to cast the rays and estimate the 3D points in the triangulation process are often highly imprecise. There is a common optimization method called *Bundle Adjustment (BA)* to improve the estimation. BA refers to the problem of jointly refining the estimated camera poses and 3D points in an optimal manner using reprojection error as the quality metric. Given a set of n cameras with initial poses (translations and orientations) and, m 3D points, BA optimization is defined as a least squares minimization using the L_2-norm or sum-of-squared reprojection errors:

$$E = \min_{\mathbf{R}_i, \mathbf{t}_i, \mathbf{K}_i, \mathbf{X}_j} \sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{x}_{ji} - g(\mathbf{X}_j, \mathbf{R}_i, \mathbf{t}_i, \mathbf{K}_i)\|^2 \qquad (5)$$

where $\mathbf{R}_i$, $\mathbf{t}_i$, $\mathbf{K}_i$ are respectively the rotation matrix, translation vector and calibration (intrinsic) matrix of the i-th camera, $\mathbf{X}_j$ is the j-th 3D point in the scene and observation $\mathbf{x}_{ji}$ is the 2D image coordinates of feature $\mathbf{X}_j$ in camera i. The mapping $g(\mathbf{X}_j, \mathbf{R}_i, \mathbf{t}_i, \mathbf{K}_i)$ is the reprojection model defined in (1). The reprojection error basically measures the Euclidean

distances between the projection of an estimated 3D point on its corresponding views using the estimated camera poses (rotation and translation) and the observation 2D point in the same view. In an ideal case (noise free), E must be zero. However, this is not the case due to presence of noise both in the feature correspondences (*e.g.* low localization precision) and also in the estimated camera poses. In order to optimize these estimations and mitigate the noise level, Levenberg Marquardt (LM) [8], [52] is the most widely used solver for BA optimization process. It is worth mentioning that the intrinsic parameters of the cameras used in our experiments are all considered to be available. When this is not the case, camera calibration methods for estimating camera intrinsic parameters can be utilized [53]–[55].

D. Dense Reconstruction

After generating the sparse point cloud and camera poses by SfM, COLMAP recovers a dense model of the scene using a MVS pipeline [6]. The depth and normal estimates for each registered image are first produced with pixel-wise view selection using photometric and geometric constraints. Then the depth and normal estimates are fused into a dense point cloud. Finally, a dense surface is created from the fused point cloud.

VI. Experimental Results

We evaluated the local features discussed in Section III for image-based 3D reconstruction using SfM and MVS on three synthetic aerial image sequences. Implementation of handcrafted feature methods SIFT, SURF, AKAZE, and ORB used the OpenCV 4.2 C++ (CPU) library. The open source code and pre-trained models for the deep learning-based feature methods DeepCompare, LF-Net, and SuperPoint were by the original authors. Experiments for all methods used the same computer with an 8-core Intel Core i7-7700HQ 2.80GHz CPU and an NVIDIA GeForce GTX 1060 GPU.

A. Evaluation Protocols

Algorithm 4 Local Feature Evaluation Procedure for SfM and MVS Pipelines Using COLMAP Drivers [7]

1: Feature point detection and description
2: Feature matching ← DistanceRatioMatching
3: Feature track creation
4: Structure-from-Motion (SfM) using COLMAP
5: Multi-View Stereo (MVS) using COLMAP
6: Quantitative evaluation

1) SfM and MVS Using COLMAP: The feature detection and matching for establishing feature tracks discussed in Section V-A are based only on 2-D image appearance information. We use COLMAP [5]–[7] to evaluate the features in the context of Structure-from-Motion and Multi-View Stereo since COLMAP is widely used for image-based 3D reconstruction tasks [32]. The evaluation procedure is shown in Algorithm 4. The image-based 3D reconstruction pipeline generally performs SfM first on input image sequences to produce the optimized camera poses and a sparse representation of the scene. After that, MVS is performed on the image sequence and the output of SfM to reconstruct a dense point cloud of the scene. As a result, the accuracy of a feature matching approach can have a significant impact on the quality of both SfM and MVS and it is beneficial to have a quantitative evaluation of different features in this scenario. We select the following metrics to quantify the feature performances.

- *# Sparse 3D Points*: number of 3D points reconstructed in the sparse model generated by SfM.
- *Average Track Length*: average number of consecutively matched feature keypoints (verified observations) along a feature track within an image sequence.
- *Average 2D Observations per Image*: average number of verified matched keypoints in one image within a sequence.
- *Reprojection Error*: distance between the projected image point from the triangulated 3D point and the observed image point.
- *# Inlier Pairs*: total number of inlier pairs within a sequence. An inlier image pair is defined as a pair of images in the input sequence that contains a minimum of 15 inlier feature matches.
- *# Inlier Matches*: total number of verified inlier feature matches (correspondences) within a sequence. Two-view geometric verification protocols [5] are used to differentiate inlier matches from outliers.
- *# Dense 3D Points*: number of 3D points reconstructed in the dense model generated by MVS.

2) Camera Pose Error: To evaluate features in terms of recovered camera pose accuracy, we compute position error and angular error for each camera recovered by COLMAP using feature tracks from each method. The camera poses used in 3D voxel renderer for synthetic data generation (shown in Figure 1) are used as ground-truth. The coordinate system of the recovered cameras is aligned to that of the ground-truth cameras [56]. The positional L_2 error for camera i is,

$$e_c(i) = \left\| \widehat{\mathbf{C}}_i - \mathbf{C}_i \right\| \tag{6}$$

where $\widehat{\mathbf{C}}_i$ is the estimated position of camera i using COLMAP, $\mathbf{C}_i$ as the ground-truth location of camera i, with,

$$RMSE_c = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_c^2(i)} \ . \tag{7}$$

The maximum position error $e_{c(i)}$ can be useful to identify extreme outliers in the estimated camera positions,

$$E_c = \max_{i=1,\ldots,n} (e_c(i)) \tag{8}$$

The camera angular error can be measured in several ways including Euler angles, quaternions, and the polar or rotation angle, based on an axis-angle representation [57]. Using the axis-angle representation, a rotation matrix is defined as a rotation by angle α around an axis, with $\text{trace}(\mathbf{R}) = 1 + 2\cos(\alpha)$. The angular error for camera i is then [57], [58],

$$e_\alpha(i) = \arccos\left[(\text{trace}(\mathbf{R}_i \widehat{\mathbf{R}}_i^{\mathsf{T}}) - 1)/2\right] \tag{9}$$

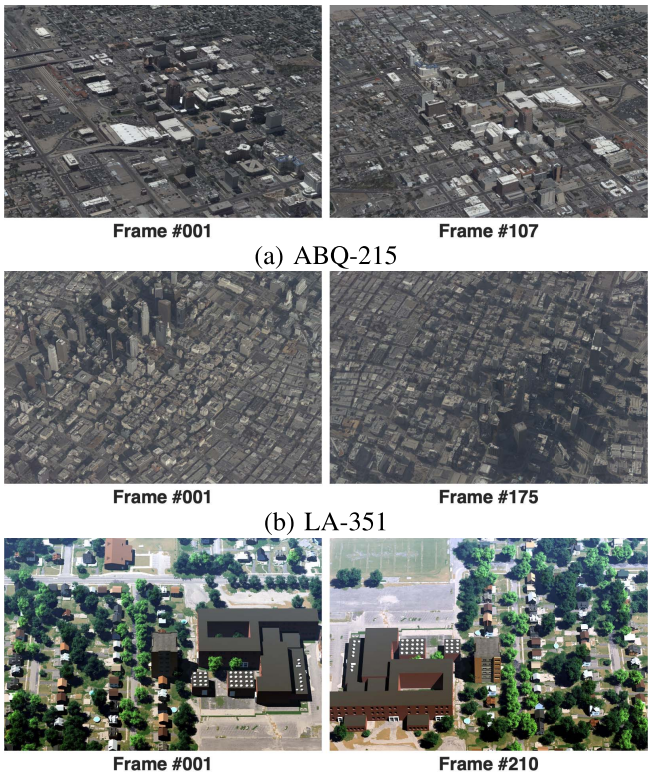


Fig. 5. First and middle frames from three *synthetic* aerial image sequences. ABQ-215 and LA-351 sequences were generated by the proposed 3D voxel renderer, consisting of 215 and 351 images respectively. Image size for both sequences are 1650 × 1100. RIT-DIRSIG is a public synthetic geospatial dataset consisting of 420 aerial images. Image size for RIT-DIRSIG is 1200 × 800.

where $\widehat{\mathbf{R}}_i$ is the COLMAP estimated rotation matrix for camera i, $\mathbf{R}_i$ is the ground-truth rotation matrix for camera i. Similar to position error, Eq. 7 and Eq. 8, RMSE and maximum of angular error are calculated as,

$$RMSE_\alpha = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_\alpha^2(i)} \tag{10}$$

$$E_\alpha = \max_{i=1,\ldots,n}(e_\alpha(i)). \tag{11}$$

B. Aerial Imagery Datasets

We evaluate the local features discussed in Section III on two sequences from the proposed MU synthetic WAMI dataset and a public synthetic geospatial dataset (RIT-DIRSIG). The detailed data information is shown in Table III. Sample frames from each image sequence are shown in Figure 5.

1) MU Synthetic WAMI Dataset: We generated two synthetic WAMI aerial image sequences of urban scenes, *i.e.* Albuquerque (ABQ) and Los Angeles (LA), using the 3D voxel renderer described in Section IV. The dense 3D point cloud reconstructions were estimated using AgiSoft [50]. The input to AgiSoft included the real aerial image sequences that were captured by an airborne camera mounted on an aircraft circling the downtown areas of ABQ and LA. The flight trajectory for the real aerial data collection was a full orbit and the camera constantly tracked the center of the scene. We used

TABLE III
CAMERA AND FLIGHTPATH PARAMETERS TO GENERATE SYNTHETIC AERIAL IMAGE SEQUENCES INCLUDING FRAME SIZE (QUARTER RESOLUTION COMPARED TO ORIGINAL WAMI [10], [15] BY SHORTENING FOCAL LENGTH, SAME FOV), HEIGHT ABOVE GROUND LEVEL (AGL), CIRCULAR ORBIT RADIUS (KM), AVERAGE GROUND SAMPLING DISTANCE (CM) AND LENS FOCAL LENGTH (PIXEL UNITS)

Dataset	# Images	Image Size	AGL (km)	Orbit Radius (km)	GSD (cm)	Focal Length (px)
ABQ-215	215	1650×1100	1.50	2.50	100	4413.1
LA-351	351	1650×1100	4.50	4.30	200	4443.7
RIT-DIRSIG	420	1200×800	0.90	1.40	30	3909.0

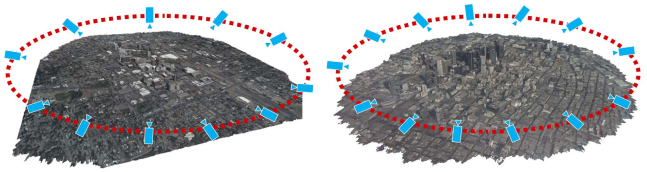


Fig. 6. Flightpaths used for generating synthetic WAMI sequences. ABQ-215 consists of 215 cameras (views). Altitude above ground level (AGL) and orbit radius for ABQ-215 are 1.5km and 2.5km respectively. LA-351 consists of 351 cameras. AGL and orbit radius for LA-351 are 4.5km and 4.3km respectively.

the same camera flightpaths for rendering synthetic views, with exact camera poses, for both ABQ-215 and LA-351 sequence. The camera flight paths are shown in Figure 6. Note that we used wider lens, shorter focal length to generate MU Synthetic WAMI dataset in order to accommodate deep learning algorithms such as LF-Net and SuperPoint that have difficulty scaling up to large images.

2) RIT-DIRSIG Synthetic Geospatial Dataset: Rochester Institute of Technology (RIT) Digital Imaging and Remote Sensing Image Generation (DIRSIG) dataset [17] provides synthetic aerial imagery and camera pose information. The synthetic frame was generated using a hand-built 3D scene of a suburban area. The camera flightpath was a simulated full circular orbit around the scene.

C. Feature Detection Evaluation

We use the Dice coefficient to assess the quality of the keypoint feature detections in the synthetic WAMI frames compared to the original ABQ-215 and LA-351 images [10], [15] (see Figure 7). The per frame (or per camera) Dice coefficient for each feature is,

$$\text{Dice}(i, f) = \frac{2 \times |\widehat{\mathbf{M}}_{i,f} \cap \mathbf{M}_{i,f}|}{|\widehat{\mathbf{M}}_{i,f}| + |\mathbf{M}_{i,f}|} \tag{12}$$

where $\widehat{\mathbf{M}}_{i,f}$ is the keypoints mask for the i^{th} synthetic frame (or camera) and feature detector, f, in which each keypoint is one pixel. $\mathbf{M}_{i,f}$ is the ground-truth keypoints (single pixel detection) mask extracted using the corresponding real image. $|\widehat{\mathbf{M}}_{i,f}|$ and $|\mathbf{M}_{i,f}|$ represent the number of non-zero elements in each mask. The Dice coefficient compares the baseline (ground-truth) keypoints extracted by each feature detector applied to the original WAMI images with corresponding synthetic frames. Point correspondences are identified within a tolerance of 1 pixel in position. The larger the Dice coefficient,

TABLE IV
TIMING ON THREE SYNTHETIC AERIAL IMAGE SEQUENCES. AVERAGE PER FRAME TIME IS THE TIME TAKEN TO COMPLETE FEATURE DETECTION AND DESCRIPTION INCLUDING I/O. SPEED IS THE TIME TO PROCESS 10^6 PIXELS. FIRST, SECOND, AND THIRD BEST RESULTS FOR EACH SEQUENCE ARE HIGHLIGHTED IN RED, GREEN, AND BLUE, RESPECTIVELY

Method	Platform	Average Per Frame Time (s)			Speed
		ABQ-215	LA-351	RIT-DIRSIG	(s/Mpixels)
SIFT	CPU	1.92	1.99	1.00	1.07
SURF	CPU	1.32	1.51	**0.50**	0.69
AKAZE	CPU	**0.39**	**0.40**	**0.16**	**0.20**
ORB	CPU	**0.61**	**0.61**	**0.20**	**0.29**
DeepCompare	GPU	4.76	4.82	5.38	3.63
LF-Net	GPU	1.42	1.39	0.51	0.69
SuperPoint	GPU	**0.72**	**0.65**	0.51	**0.43**

the more similar the detected keypoints are compared to the ground-truth. As shown in Figure 7, the majority of keypoints across detectors and synthetic frames are in agreement with their corresponding ground-truth keypoint locations with the highest for SIFT and ORB and lowest for MSER. Both handcrafted and learned feature detectors generated consistent detection results across frames in ABQ-215 and LA-351 sequences. MSER detector produced the lowest Dice coefficient but still more than half of the keypoints MSER detected on synthetic frames are consistent with those it detected on the real aerial images. The feature detection performances of different methods demonstrate the high image quality of the proposed synthetic WAMI sequences with respect to the real WAMI dataset.

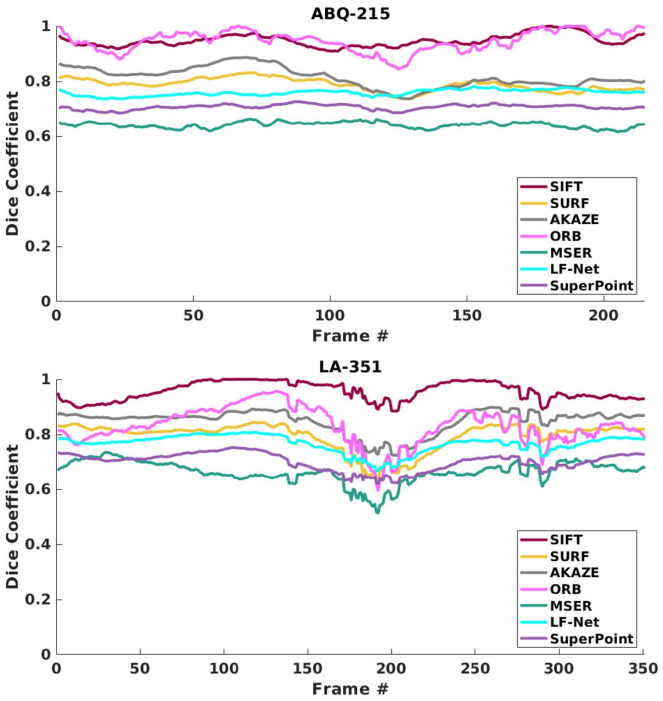


Fig. 7. Localization agreement of 2D feature detectors, between synthetically generated WAMI frames and original corresponding views, downsampled to match a synthetic camera. Thresholds adjusted to get 5000 keypoints in both synthetic and downsampled original images. Dice coefficient equivalent to percent of point correspondences with a tolerance of 1 pixel in position.

D. SfM and MVS Evaluation

The quantitative evaluation results of handcrafted features and learned features for SfM and MVS tasks on synthetic aerial image sequences are shown in Table V. To minimize the bias within an image sequence, we use the same number of feature points for each method, *i.e.* 5000 keypoints are detected by each method for ABQ-215 and LA-351 and 3000 keypoints for RIT-DIRSIG. Average per frame run-time for each method to perform feature extraction (detection and description) in three synthetic aerial image sequences is shown in Table IV. Feature methods are ranked using each evaluation metric,

$$\text{score}(f \mid m) = \left(\frac{1}{3} \sum_{s=1}^{3} \text{rank}(f \mid m, s) \right)^{-1} \tag{13}$$

where f denotes a feature method, m an evaluation metric in Table V, and s one of the synthetic test sequences. We sort the scores and present the ranking of methods for each metric in Table VI. Additionally, we compute the overall score of each method over all evaluation metrics including speed (Table IV) using the Equation below and show the overall ranking in the last column of Table VI.

$$\text{score}(f) = \frac{1}{12} \sum_{m=1}^{12} \text{score}(f \mid m) \tag{14}$$

As shown in Table V and Table VI, the handcrafted features produce larger numbers of sparse points as SfM output for all three synthetic aerial imagery sequences. SURF performs the best on both ABQ-215 and RIT-DIRSIG sequences and the third best on the LA-351 sequence. Two binary features, AKAZE and ORB, provide similar number of sparse points as SIFT. Deep learning-based features generate less sparse points in comparison. Number of inlier pairs exhibits a similar pattern that handcrafted features outperform the learned features in general. SIFT and SURF provide consistent performance on three sequences, producing the most inlier image matching pairs. ORB which is widely used in simultaneous localization and mapping (SLAM) also yields comparable results. This metric indicates that SIFT, SURF, and ORB are capable of matching features between images with large perspective variations. Additionally, AKAZE and ORB perform well in terms of number of dense points as MVS output. Surprisingly, handcrafted features including SIFT, SURF, and AKAZE provide more accurate localization and smaller reprojection errors compared to learned features. However, the average track length using these handcrafted features are much shorter than tracks produced by recent deep learning-based features, like LF-Net and SuperPoint. Longer feature tracks tend to result in greater average reprojection error according to (5) because longer tracks consist of more cameras which on the other hand tend to generate more precise camera pose refinement (see Section VI-E for detailed discussion). For number of observations per image and number of inlier matches, LF-Net and SuperPoint also generated superior results by producing a larger number of accurate feature matches between image pairs. By comparison, DeepCompare which is an early approach for deep learning based feature

TABLE V

FEATURE EVALUATION FOR SFM AND MVS ON THREE SYNTHETIC AERIAL IMAGE SEQUENCES. RMSE AND MAXIMUM OF POSITION ERROR AND ANGULAR ERROR ARE THE EVALUATION METRICS FOR CAMERA POSE ACCURACY. UNITS FOR POSITION ERROR AND ANGULAR ERROR ARE METERS AND DEGREES, RESPECTIVELY. *# Dense 3D Points* IS USED FOR MVS EVALUATION AND THE OTHER METRICS ARE FOR SFM EVALUATION. FIRST, SECOND, AND THIRD BEST RESULTS FOR EACH COLUMN IN EACH IMAGE SEQUENCE ARE HIGHLIGHTED IN RED, GREEN, AND BLUE, RESPECTIVELY

ABQ-215 (# frames: 215)

Method	# Sparse 3D Points	Avg Track Length	Avg 2D Obs Per Image	# Inlier Pairs	# Inlier Matches	Reproj E (Eq. 5)	# Dense 3D Points	$RMSE_c$ (Position)	Max E_c	$RMSE_\alpha$ (Angular)	Max E_α
SIFT	62,956	**15.94**	**4669**	**17,792**	**6,830,488**	**0.43**	2,087,275	**1.27**	**2.63**	0.0263	**0.0390**
SURF	**71,793**	13.45	4489	**22,449**	4,905,780	**0.56**	2,080,177	1.63	3.87	**0.0249**	0.0468
AKAZE	**70,819**	14.11	4648	7,000	3,680,955	**0.41**	2,083,609	2.35	4.15	0.0438	0.0631
ORB	**68,125**	8.51	2696	**22,710**	1,550,521	0.60	**2,131,788**	5.35	6.94	0.1011	0.1371
DeepCompare	39,883	14.09	2613	4,601	1,706,791	0.80	**2,111,696**	3.50	7.90	0.0427	0.0629
LF-Net	50,602	**20.06**	**4722**	8,742	**6,408,293**	0.62	2,083,745	**1.08**	**2.58**	**0.0120**	**0.0196**
SuperPoint	46,261	**23.60**	**5078**	7,149	**8,794,655**	0.75	**2,087,964**	**1.21**	**2.83**	**0.0147**	**0.0271**

LA-351 (# frames: 351)

Method	# Sparse 3D Points	Avg Track Length	Avg 2D Obs Per Image	# Inlier Pairs	# Inlier Matches	Reproj E (Eq. 5)	# Dense 3D Points	$RMSE_c$ (Position)	Max E_c	$RMSE_\alpha$ (Angular)	Max E_α
SIFT	**62,153**	**27.06**	**4792**	**61,289**	**27,600,727**	**0.49**	3,797,807	**2.39**	**5.73**	0.0403	**0.0597**
SURF	**59,932**	24.64	4207	**60,404**	16,227,900	0.62	3,826,895	4.06	9.74	0.0540	0.0742
AKAZE	55,130	25.60	4021	21,160	11,760,395	**0.40**	**3,866,452**	6.18	13.61	0.0983	0.1201
ORB	**63,611**	17.47	3166	**57,190**	4,772,146	**0.61**	**4,070,464**	8.94	22.45	0.0940	0.1174
DeepCompare	47,929	22.97	3137	13,752	6,013,048	0.89	**3,944,201**	5.48	15.38	**0.0382**	0.0742
LF-Net	48,270	**34.67**	**4768**	39,915	**24,464,602**	0.69	3,806,072	**3.26**	**8.72**	**0.0278**	**0.0478**
SuperPoint	42,905	**39.41**	**4817**	17,155	**26,223,759**	0.71	3,798,926	**1.94**	**5.28**	**0.0223**	**0.0376**

RIT-DIRSIG (# frames: 420)

Method	# Sparse 3D Points	Avg Track Length	Avg 2D Obs Per Image	# Inlier Pairs	# Inlier Matches	Reproj E (Eq. 5)	# Dense 3D Points	$RMSE_c$ (Position)	Max E_c	$RMSE_\alpha$ (Angular)	Max E_α
SIFT	**45,572**	25.38	2754	**87,990**	**23,125,789**	**0.51**	6,892,999	**1.69**	**5.19**	0.2509	**0.2811**
SURF	**50,714**	23.63	**2854**	**87,989**	15,308,022	**0.65**	6,923,138	**2.40**	9.70	**0.1641**	0.8531
AKAZE	**44,486**	27.76	**2941**	50,204	12,866,713	**0.49**	**6,955,397**	2.91	6.31	0.3799	0.4811
ORB	35,031	**28.43**	2371	87,571	5,451,781	0.80	**7,395,128**	7.35	17.13	0.5556	0.8162
DeepCompare	19,881	16.33	773	12,186	1,433,961	1.01	**7,463,046**	9.77	25.11	0.6638	1.0050
LF-Net	39,054	**30.72**	**2857**	**87,581**	**15,845,534**	0.71	6,892,489	2.41	**5.44**	**0.1959**	**0.2325**
SuperPoint	32,401	**35.19**	2714	24,842	**19,113,996**	0.97	6,915,165	**1.39**	**3.58**	**0.0617**	**0.0824**

TABLE VI

AVERAGE RANK (EQ. 13) OF FEATURES FOR EACH EVALUATION METRIC (COLUMN) OVER THREE SYNTHETIC AERIAL IMAGE SEQUENCES IN TABLE IV AND TABLE V. LAST COLUMN SHOWS OVERALL RANK FOR EACH FEATURE USING AVERAGE OF RANKS FOR ALL METRICS (EQ. 14)

Rank	**# Sparse 3D Points**	**Avg Track Length**	**Avg 2D Obs Per Image**	**# Inlier Pairs**	**# Inlier Matches**	**Reproj E (Eq. 5)**	**# Dense 3D Points**	***RMSE*$_c$ (Position)**	**Max E_c**	***RMSE*$_\alpha$ (Angular)**	**Max E_α**	**Speed**	**Overall Rank**
1	SURF	Super-Point	SuperPoint/ LF-Net	SIFT	SIFT	AKAZE	ORB	Super-Point	Super-Point	Super-Point	Super-Point	AKAZE	Super-Point
2	SIFT	LF-Net	SIFT	SURF	Super-Point	SIFT	Deep-Compare	SIFT	SIFT	LF-Net	LF-Net	ORB	SIFT
3	AKAZE	SIFT	AKAZE	ORB	LF-Net	SURF	AKAZE	LF-Net	LF-Net	SURF	SIFT	Super-Point	AKAZE
4	ORB	AKAZE	SURF	LF-Net	SURF	ORB	Super-Point	SURF	SURF	SIFT	SURF	SURF	LF-Net
5	LF-Net	SURF	ORB	AKAZE	AKAZE	LF-Net	SURF	AKAZE	AKAZE	Deep-Compare	Deep-Compare	LF-Net	SURF
6	Super-Point	ORB	Deep-Compare	Super-Point	Deep-Compare	Super-Point	SIFT	Deep-Compare	ORB	AKAZE	AKAZE	SIFT	ORB
7	Deep-Compare	Deep-Compare	-	Deep-Compare	ORB	Deep-Compare	LF-Net	ORB	Deep-Compare	ORB	ORB	Deep-Compare	Deep-Compare

extraction provides weak feature matching performance, resulting in the smallest number of observations per frame, fewest number of inlier pairs and the largest reprojection error.

Similar to the observation by Fan *et al.* in [16], our evaluation results demonstrate that a binary feature has an advantage of speed and is competent for a SfM pipeline by producing reasonable results compared to the floating-point features such as SIFT and SURF which are computationally expensive. However, the binary feature ORB generates one of the shortest average track length among all the features. Moreover, ORB produces significantly fewer inlier matches than SIFT, SURF, and AKAZE using the same amount of feature keypoints. Another observation in [16] on the learned features underperforming the handcrafted features (SIFT and SURF) is consistent with ours when only the usual indirect metrics such as reprojection error and number of generated sparse points *etc*, are taken into account.

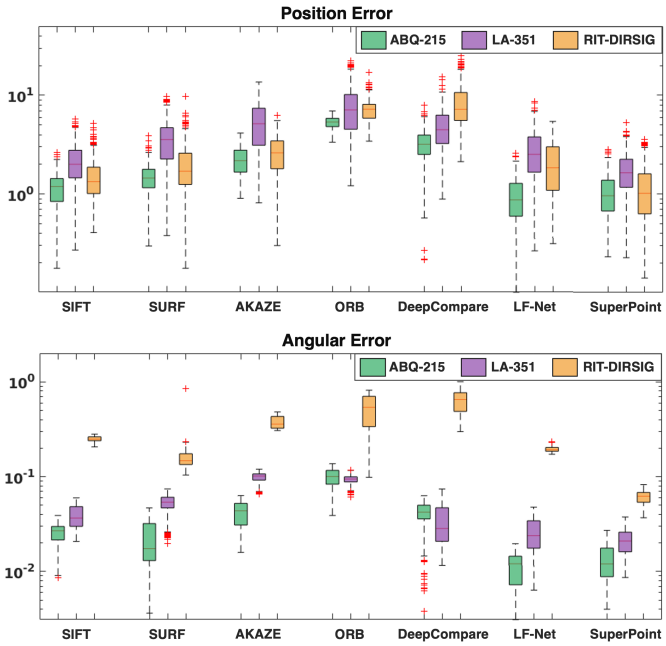


Fig. 8. Visualization of position error (top) and angular error (bottom) for each method on three synthetic aerial image sequences using box plots. The bottom and top edges of each box represent the 25-th and 75-th percentiles respectively. The bar inside each box marks the median value. The whiskers extend to the extreme inlier data points. The outlier data points are marked with red plus symbols. Units for position error and angular error are meters and degrees, respectively.

E. Camera Pose Estimation Accuracy

The camera pose estimation accuracy as an output of SfM is evaluated by computing the position error and angular error compared to the ground-truth camera poses for each view in the sequence. Our results show that the learned features (LF-Net and SuperPoint) in recent years produce superior results in terms of precision of the estimated camera poses (rotation and translation). We believe the reason behind this is the ability of these learned features to persistently detect and accurately match keypoints over the image sequences. This is verified in the *Avg Track Length* column in Table V despite the fact that the reprojection error for these advanced learned features are higher than that of the handcrafted features like SIFT and SURF. As shown in Table VI, SIFT and SURF also generate low camera pose errors as well as relatively long feature tracks. This is aligned with the observation in [10] that a longer feature track leads to camera pose refinements of higher quality. In other words, the longer tracks tie more cameras to each other within the BA optimization and lead to a lower drift on the camera 3D poses by keeping many cameras together. The summary statistics of position error and angular error of each method for three image sequences are illustrated using box plots in Figure 8. The average position errors and angular errors of each method for the three sequences are shown in Table VII. Additionally, we evaluated the camera pose estimation accuracy of COLMAP baseline using three SIFT implementations provided (SIFT-GPU, RootSIFT [59], and DSP-SIFT [29]). COLMAP using RootSIFT and SIFT-GPU features generate better or comparable results with SuperPoint, slightly outperforming COLMAP using DSP-SIFT features.

TABLE VII

AVERAGE REPROJECTION ERROR (PIXELS), CAMERA POSE ERRORS RMSE AND MAXIMUM POSITION ERROR (METERS), AND ANGULAR ERROR (DEGREES) ACROSS THREE SYNTHETIC AERIAL SEQUENCES FOR THE SEVEN FEATURE METHODS EVALUATED. FEATURES ARE SORTED BY THEIR AVERAGE RANK (COLUMNS 3 TO 6) ACROSS DATASETS IN TABLE V; SIFT AND LF-NET ARE TIED

Method	Reproj E (Eq. 5)	$RMSE_c$ (Position)	Max E_c	$RMSE_\alpha$ (Angular)	Max E_α
SuperPoint	0.81	1.51	3.90	0.03	0.05
SIFT	0.48	1.78	4.52	0.11	0.13
LF-Net	0.67	2.25	5.58	0.08	0.10
SURF	0.61	2.70	7.77	0.08	0.32
AKAZE	0.43	3.81	8.02	0.17	0.22
ORB	0.67	7.21	15.51	0.25	0.36
DeepCompare	0.90	6.25	16.13	0.25	0.38

In summary, all methods demonstrate consistent camera pose refinement performances across different image sequences. SuperPoint as the state-of-the-art learned features outperform the rest of the feature methods and both provide the lowest position and angular errors across frames with small variances. Handcrafted floating-point SIFT and SURF yield comparable results as LF-Net. Handcrafted binary features AKAZE has remarkably higher speed performance but its camera pose refinement accuracy is inferior to SIFT and SURF. Another binary feature ORB and the early deep learning-based method DeepCompare exhibit the worst results in terms of both position error and angular error. Both ORB and DeepCompare show large error variances across frames in each sequence.

VII. CONCLUSION

We evaluated the performance of some of the most popular handcrafted and deep learning-based local features, within the widely used COLMAP open source SfM and MVS pipelines. We used a pixel accurate voxel renderer to generate simulated city-scale WAMI datasets from an AgiSoft 3D point cloud model reconstructed from real aerial imagery, as well as a fully synthetic open source campus model RIT-DIRSIG with synthetic aerial imagery. Unlike most existing work in this domain, our evaluations includes metrics for measuring the feature performance on the camera pose estimation, as a result of the proposed realistic-looking synthetic view generation framework. Our categorized ranking of the features shows the suitability and reliability of each type of feature depending on the application. Handcrafted features produced more reconstructed sparse and dense points with smaller reprojection errors compared to the deep learning-based features. Surprisingly, smaller reprojection error did not result in better camera pose estimates. However, recently developed learning-based features, *i.e.* SuperPoint and LF-Net, provided longer feature tracks and greater number of inlier matches. More importantly, advanced learned features outperformed the handcrafted features in terms of recovered camera pose accuracy. We observe that the earlier DeepCompare learned feature generated lower quality results in general. Handcrafted binary features, AKAZE and ORB, had a significant speed advantage but their performance was inferior to other state-of-the-art features. Future work will focus on further improving image quality of the synthetic aerial imagery, using calibrated

scale models and evaluating features using additional SfM and MVS pipelines.

ACKNOWLEDGMENT

The raw wide area motion imagery (WAMI) aerial data sets for ABQ and LA were collected by Steve Suddarth at TransparentSky under contract. The dense 3D point clouds generated using AgiSoft were provided by the U.S. Army Engineer Research and Development Center Geospatial Research Laboratory. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the U. S. Government or agency thereof.

REFERENCES

[1] K. Gao, H. Aliakbarpour, G. Seetharaman, and K. Palaniappan, "DCT-based local descriptor for robust matching and feature tracking in wide area motion imagery," *IEEE Geosci. Remote Sens. Lett.*, early access, Jun. 26, 2020, doi: 10.1109/LGRS.2020.3000762.

[2] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 66–89, Jan. 2016.

[3] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4104–4113.

[4] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 501–518.

[5] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1482–1491. [Online]. Available: https://github.com/ahojnnes/local-feature-evaluation/

[6] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–770, Jun. 2004.

[7] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, Sep. 1981.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[9] N. Al-Shakarji, F. Bunyak, H. Aliakbarpour, G. Seetharaman, and K. Palaniappan, "Multi-cue vehicle detection for semantic video compression in georegistered aerial videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2019, pp. 56–65.

[10] H. AliAkbarpour, K. Palaniappan, and G. Seetharaman, "Parallax-tolerant aerial image georegistration and efficient camera pose refinement–without piecewise homographies," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 8, pp. 4618–4637, Aug. 2017.

[11] H. Aliakbarpour, K. Palaniappan, and G. Seetharaman, "Stabilization of airborne video using sensor exterior orientation with analytical homography modeling," in *Machine Vision and Navigation*. New York, NY, USA: Springer, 2020, pp. 579–595.

[12] R. Aktar, H. AliAkbarpour, F. Bunyak, T. Kazic, G. Seetharaman, and K. Palaniappan, "Geospatial content summarization of UAV aerial imagery using mosaicking," *Proc. SPIE*, vol. 10645, Apr. 2018, Art. no. 106450I.

[13] Y. Furukawa and C. Hernández, "Multi-view stereo: A tutorial," *Found. Trends Comput. Graph. Vis.*, vol. 9, nos. 1–2, pp. 1–148, 2015.

[14] R. Porter, A. Fraser, and D. Hush, "Wide-area motion imagery," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 56–65, Sep. 2010.

[15] K. Palaniappan, R. M. Rao, and G. Seetharaman, "Wide-area persistent airborne video: Architecture and challenges," in *Distributed Video Sensor Networks*. London, U.K.: Springer, 2011, pp. 349–371.

[16] B. Fan *et al.*, "A performance evaluation of local features for image-based 3D reconstruction," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4774–4789, Oct. 2019.

[17] D. Nilosek, D. J. Walvoord, and C. Salvaggio, "Assessing geoaccuracy of structure from motion point clouds from long-range image collections," *Opt. Eng.*, vol. 53, no. 11, Nov. 2014, Art. no. 113112.

[18] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3D objects," *Int. J. Comput. Vis.*, vol. 73, no. 3, pp. 263–284, Jul. 2007.

[19] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[20] K. Mikolajczyk *et al.*, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, nos. 1–2, pp. 43–72, Nov. 2005.

[21] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Proc. Eur. Conf. Comput. Vis.* Florence, Italy: Springer, 2012, pp. 759–773.

[22] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2010, pp. 778–792.

[23] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2548–2555.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[25] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanaes, "Large scale multi-view stereopsis evaluation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 406–413.

[26] K. Wilson and N. Snavely, "Robust global translations with 1DSfM," in *Proc. Eur. Conf. Comput. Vis.* Zürich, Switzerland: Springer, 2014, pp. 61–75.

[27] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 118–126.

[28] A. Bursuc, G. Tolias, and H. Jégou, "Kernel local descriptors with implicit rotation matching," in *Proc. 5th ACM Int. Conf. Multimedia Retr.*, Jun. 2015, pp. 595–598.

[29] J. Dong and S. Soatto, "Domain-size pooling in local descriptors: DSP-SIFT," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5097–5106.

[30] M. S. Mueller, T. Sattler, M. Pollefeys, and B. Jutzi, "Image-to-image translation for enhanced feature matching, image retrieval and visual localization," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 42, pp. 111–119, Sep. 2019.

[31] E. Özdemir, I. Toschi, and F. Remondino, "A multi-purpose benchmark for photogrammetric urban 3D reconstruction in a controlled environment," *ISPRS-Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 4212, pp. 53–60, Sep. 2019.

[32] E.-K. Stathopoulou, M. Welponer, and F. Remondino, "Open-source image-based 3D reconstruction pipelines: Review, comparison and evaluation," *ISPRS-Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 42, pp. 331–338, Nov. 2019.

[33] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends Comput. Graph. Vis.*, vol. 3, no. 3, pp. 177–280, 2007.

[34] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[35] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.* Graz, Austria: Springer, 2006, pp. 404–417.

[36] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *Proc. Brit. Mach. Vis. Conf.*, 2013, p. 1.

[37] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image Vis. Comput.*, vol. 22, no. 10, pp. 761–767, Sep. 2004.

[38] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "LF-Net: Learning local features from images," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2018, pp. 6234–6244.

[39] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 224–236.

[40] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis.* Graz, Austria: Springer, 2006, pp. 430–443.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[43] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4353–4361.
[44] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 43–57, Jan. 2011.
[45] J. Heinly, J. L. Schönberger, E. Dunn, and J.-M. Frahm, "Reconstructing the world* in six days *(as captured by the yahoo 100 million image dataset)," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3287–3295.
[46] B. Thomee *et al.*, "YFCC100M: The new data in multimedia research," *Commun. ACM*, vol. 59, no. 2, pp. 64–73, 2016.
[47] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Zürich, Switzerland: Springer, 2014, pp. 740–755.
[48] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker, "Universal correspondence network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2414–2422.
[49] Pix4D. *Pix4D–A Unique Suite of Photogrammetry Software for Drone Mapping*. Accessed: Jul. 24, 2020. [Online]. Available: https://www.pix4d.com/
[50] AgiSoft. *AgiSoft Metashape–Photogrammetric Processing of Digital Images and 3D Spatial Data Generation*. Accessed: Jul. 24, 2020. [Online]. Available: https://www.agisoft.com/
[51] D. Schmalstieg and R. F. Tobler, "Fast projected area computation for three-dimensional bounding boxes," *J. Graph. Tools*, vol. 4, no. 2, pp. 37–43, Jan. 1999.
[52] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—A modern synthesis," in *Proc. Int. Workshop Vis. Algorithms*. Berlin, Germany: Springer, 1999, pp. 298–372.
[53] S. Liu, Y. Peng, Z. Sun, and X. Wang, "Self-calibration of projective camera based on trajectory basis," *J. Comput. Sci.*, vol. 31, pp. 45–53, Feb. 2019.
[54] H. Zhu *et al.*, "Camera calibration from very few images based on soft constraint optimization," *J. Franklin Inst.*, vol. 357, no. 4, pp. 2561–2584, Mar. 2020.
[55] L. R. Ramírez-Hernández *et al.*, "Improve three-dimensional point localization accuracy in stereo vision systems using a novel camera calibration method," *Int. J. Adv. Robotic Syst.*, vol. 17, no. 1, Jan. 2020, Art. no. 172988141989671.
[56] B. K. Horn, H. M. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 5, no. 7, pp. 1127–1135, 1988.
[57] P. R. Evans, "Rotations and rotation matrices," *Acta Crystallographica D, Biol. Crystallogr.*, vol. 57, no. 10, pp. 1355–1359, 2001.
[58] A. R. Widya, A. Torii, and M. Okutomi, "Structure from motion using dense CNN features with keypoint relocalization," *IPSJ Trans. Comput. Vis. Appl.*, vol. 10, no. 1, p. 6, Dec. 2018.
[59] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2911–2918.

Ke Gao (Graduate Student Member, IEEE) received the M.S. degree in electrical engineering from the University of Missouri, Columbia, MO, USA, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science. His research interests include visual object tracking, feature detection and matching for wide area motion imagery, and digital image processing for plant morphometry.

Hadi AliAkbarpour (Member, IEEE) received the Ph.D. degree in automation and robotics, under the supervision of Prof. Jorge Dias, from the Department of Electrical and Computer Engineering, University of Coimbra, Portugal. From September 2007 to March 2013, he was a Researcher with the Institute of Systems and Robotic (ISR). He is currently an Assistant Research Professor with the Computer Science Department, University of Missouri, Columbia, USA. His research interests include computer vision, wide area motion imagery (WAMI), and robotics and computational human behavior analysis.

Joshua Fraser received the Ph.D. degree in computer science from the University of Missouri, Columbia, MO, USA. He is currently a Research Scientist with the University of Missouri. His research interests include large multidimensional data visualization, human–computer interaction, and interactive rendering.

Koundinya Nouduri received the M.S. degree in computer science from the University at Buffalo (SUNY). He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, University of Missouri, Columbia. His research interests include deep learning-based aerial image matching and localization, image denoising, and text generation using deep learning.

Filiz Bunyak received the Ph.D. from the University of Missouri at Rolla. She is currently an Assistant Research Professor with the Computer Science Department, University of Missouri. Her research interests include image processing, computer vision, and pattern recognition with emphasis on biomedical image analysis, aerial and wide-area surveillance, visual tracking, data fusion, segmentation, and level set methods.

Ricky Massaro received the Ph.D. degree in computational sciences from George Mason University, in 2011. He has been with ERDC for more than 16 years and specializes in photogrammetric reconstruction from military sensors and platforms. He is currently a Physical Scientist with the USACE-ERDC-Geospatial Research Lab, Alexandria, VA, USA.

Guna Seetharaman (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Miami, FL, USA. He was a Principal Engineer of Computing Architectures with the Information Directorate, Air Force Research Laboratory (AFRL), until 2014. He is currently a Senior Scientist for Advanced Computing Concepts, Information Directorate, U.S. Naval Research Laboratory (NRL), Washington, DC, USA. His current research interests include high performance computing, computer vision, and information exploitation. He is the Chair of the IEEE Mohawk Valley Section.

Kannappan Palaniappan (Senior Member, IEEE) received the Ph.D. degree from the University of Illinois at Urbana–Champaign, Champaign, IL, USA. He is currently a Professor with the Electrical Engineering and Computer Science Department, University of Missouri. His research interests include the synergistic intersection of image and video processing, computer vision, high-performance computing, and machine learning to understand, quantify, and model physical processes with applications to biomedical, space, and defense imaging. His recent multidisciplinary contributions range across orders of scale from subcellular microscopy at the molecular level to aerial and satellite remote sensing imaging at the macro level.