LSVL: Large-scale season-invariant visual localization for UAVs

Jouko Kinnari[1], Riccardo Renzulli[2], Francesco Verdoja[3] and Ville Kyrki[3]

Abstract— Localization of autonomous unmanned aerial vehicles (UAVs) relies heavily on Global Navigation Satellite Systems (GNSS), which are susceptible to interference. Especially in security applications, robust localization algorithms independent of GNSS are needed to provide dependable operations of autonomous UAVs also in interfered conditions. Typical non-GNSS visual localization approaches rely on known starting pose, work only on a small-sized map, or require known flight paths before a mission starts. We consider the problem of localization with no information on initial pose or planned flight path. We propose a solution for global visual localization on a map at scale up to 100 km^2, based on matching orthoprojected UAV images to satellite imagery using learned season-invariant descriptors. We show that the method is able to determine heading, latitude and longitude of the UAV at 12.6–18.7 m lateral translation error in as few as 23.2–44.4 updates from an uninformed initialization, also in situations of significant seasonal appearance difference (winter-summer) between the UAV image and the map. We evaluate the characteristics of multiple neural network architectures for generating the descriptors, and likelihood estimation methods that are able to provide fast convergence and low localization error. We also evaluate the operation of the algorithm using real UAV data and evaluate running time on a real-time embedded platform. We believe this is the first work that is able to recover the pose of an UAV at this scale and rate of convergence, while allowing significant seasonal difference between camera observations and map.

I. INTRODUCTION

The ability of a Unmanned Aerial Vehicle (UAV) to robustly estimate its position is one of the basic requirements of autonomous flight. In missions in which the UAV needs to collaborate with other agents operating in the same environment, information of position in a shared, global frame of reference is needed.

There are several possible ways to estimate position. A localization solution can rely on infrastructure built for this purpose. In outdoor UAV operations, by far the most common is to rely on Global Navigation Satellite Systems (GNSS). In ideal conditions, GNSS receivers provide measurements of position. However, GNSS, as well as other radio beacon systems, are susceptible to blockages in radio signal path, inaccuracies due to multipath propagation, and spoofing and jamming attacks [1], [2], which may be used for denying operation of UAVs in an area.

[1]J. Kinnari is with Saab Finland Oy, Salomonkatu 17B, 00100 Helsinki, Finland {firstname.lastname}@saabgroup.com

[2]R. Renzulli is with Department of Computer Science, University of Turin, Italy {firstname.lastname}@unito.it

[3]F. Verdoja and V. Kyrki are with School of Electrical Engineering, Aalto University, Finland {firstname.lastname}@aalto.fi
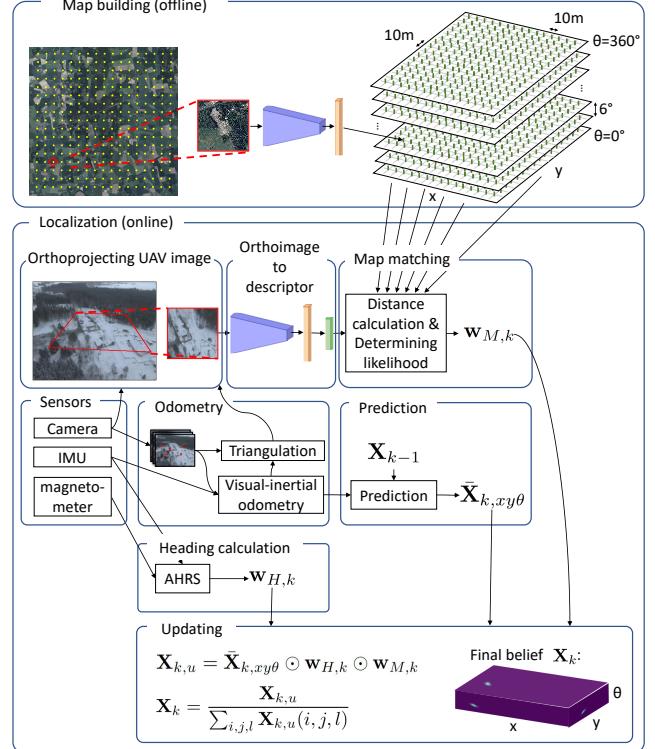

Fig. 1: Block diagram of proposed localization solution.

A sensor system commonly carried by UAVs is the combination of a camera and an Inertial Measurement Unit (IMU). Several works have focused on using this sensor combination for resolving the position of the UAV by tracking the difference to a known starting position by methods of visual-inertial odometry (VIO) [3]. As VIO implementations integrate noisy signals, they suffer from drift over long flights [4]. Simultaneous Localization and Mapping (SLAM) methods [5] can help in partially compensating for this drift by detecting loop closures, but only if the mission contains reentry to a previously visited area. In addition, VIO and SLAM approaches are not robust to random failures in position tracking; temporary failures may lead to loss of position information in the global frame with only a small chance of recovery.

If a map is available, matching camera images acquired by a UAV to the map allows compensating for drift induced by odometry methods and provides an estimate of position with respect to the map. This approach is called visual localization [6], [7]. Using a georeferenced map allows, in principle, finding the position of a UAV with respect to the map, even

in the case of no prior information on position at start of mission or after random failures in positioning.

There are, however, multiple challenges in visual localization. Appearance difference between the UAV image and the map may be significant due to changes in season — an image acquired by a UAV in winter looks very different from a satellite image acquired in summer. Moreover, the environment in which the UAV is operating may be naturally ambiguous, *e.g.*, when flying over vast areas of forests. In order for the localization solution to support recovery from random localization failures, the size of the map must correspond to the operating area of the UAV. This means that the localization solution must be computationally efficient enough to run on an onboard computer even on large maps, and it must tolerate the natural ambiguity.

In this work we propose LSVL, which addresses all of the above mentioned challenges of robust visual localization at large scale. The main contributions of this work are:

1) We propose a UAV image to map matching solution based on descriptors learned in a manner that provides invariance to seasonal appearance difference. We train the descriptor networks using only satellite images, without requiring any labeled data such as semantic classification of the terrain.
2) We present the problem of UAV visual localization as state estimation utilizing a point mass filter, and integrate it to our map matching approach. This combination allows running our algorithm in real time, with constant time and memory consumption onboard a UAV at sufficiently large scale for UAV missions.
3) We explore different architecture choices for image description, vector dimensionality, and likelihood computation, and evaluate their impact on probability to convergence, time to convergence and positioning error after convergence, on real data.
4) We show that our approach is able to tolerate natural ambiguities and resolve location of a UAV, starting from an uncertainty corresponding to an area of 100 km^2, converging to an average translation error of 12.6–18.7 m after no more than 23.2–44.4 updates with UAV camera observations when flying over ambiguous terrains under significant seasonal appearance difference between UAV image and map. Our formulation does not require a digital elevation model and can operate using 2D maps.
5) We present a simple method allowing assessing probability of convergence to true pose, enabling self-diagnostics of the localization solution, which is a key component in the robust global localization problem.
6) We compare our solution to two state-of-the-art approaches for UAV localization.
7) We demonstrate the operation of our localization solution in real time onboard a commercial UAV.

We believe this is the first work in the visual localization area that is able to find true position of the UAV, starting from a scale of uncertainty of 100 km^2, over ambiguous terrains, under significant seasonal appearance change. A block diagram of our algorithm is shown in Figure 1.

The paper is structured as follows. Sec. II goes over related work. Sec. III defines preliminaries of the localization problem. We introduce our method in Sec. IV and detail a vital part of the solution, map matching, in Sec. V. Sec. VI describes our localization experiments and we conclude the paper with discussion and conclusions in Sec. VII and Sec. VIII, respectively.

II. RELATED WORK

Visual localization of UAVs is a topic which has attracted interest especially over past few years. The state-of-the-art is covered by relatively recent surveys [6], [7]. Within the area of visual localization, most works are limited by one or more of the following assumptions: accurate initialization is required [8]–[11], the size of the operating area is limited [12], [13], the movements of the UAV are constrained to specific paths [10], the operation takes place in conditions that are very close to map in terms of appearance [8], [12], the map resolution and detail requirement is significant, such as requiring a topography map [13], or a very high flying altitude is required for successful georeferencing [14]. The following overview of related work focuses on the works that—similarly to LSVL—do not require of knowledge of initial pose and that use an easily obtainable planar 2D map for localization.

A common choice in UAV localization is to detect semantic features such as roads and intersections [14]–[17] or buildings [18]. Choi *et al.* [18] proposed a method where the UAV image is semantically segmented to find buildings in the camera view. Based on detected buildings, a rotation invariant descriptor, building ratio map (BRM), is computed from the proportion of building pixels visible in camera view, and a precomputed map with similar descriptors is used for localization. The authors demonstrated convergence of position estimate on a 6.17 km^2 map after 27 updates with 25 meters of translation between each update, with 12.01 m root-mean-square (RMS) error after convergence. The demonstration flight takes place over a residential area. The main drawback of this approach is the requirement that features of a specific semantic class (*i.e.*, buildings) are required for successful localization, which may be unavailable when flying over natural environments.

Mantelli *et al.* [19] demonstrate localizing a UAV on a map of size 1.34 km^2 using a particle filter, where particle likelihood is determined based on a handcrafted descriptor called abBRIEF. The authors initialize the particle filter with 50 000 particles and show robustness of their localization solution in comparison to BRIEF [20] descriptors on trajectory lengths up to 2.4 km, showing convergence in less than 50 m to an average translation error of 17.78 m. The descriptor is developed such that it is tolerant to illumination changes and allows fast computation of particle likelihood over a large number of pose hypotheses.

We evaluate against BRM and abBRIEF and show superior performance.

We build on the idea of compressing UAV camera observation into a compact embedding space, to allow fast testing of pose hypotheses on a large map. This has been proposed recently by *e.g.*, Bianchi *et al.* [10] who use a bottleneck autoencoder approach to compress visual observations to descriptor vectors of dimension 1000. Also Samano *et al.* [21] and Couturier *et al.* [22] train Resnet models [23] to project map tiles and UAV images to a low-dimensional (16D) embedding space. Only Samano *et al.* [21] allow movements of the UAV outside precomputed paths by precomputing a grid of embedding vectors, on which hypothesis testing is performed by interpolating a vector from this grid by each pose hypothesis and measuring distance of observed image descriptor to interpolated descriptor vector in embedding space. Authors of [21] demonstrate with simulated flight experiment the convergence of pose estimate to less than 95 m translation error in 78.2% of simulated flights by 200 updates. We take a similar approach but forgo interpolation in embedding space, to avoid unjustified implicit assumption of smoothness of embedding space, and explore other learned descriptor architectures. Compared to [21], we show significantly faster convergence, tolerance to seasonal variation and we demonstrate performance with real experiments.

III. PRELIMINARIES

We want to resolve the pose of an UAV using measurements available at the UAV during flight, with no dependency on localization or communication infrastructures. Formally, we define the pose of the UAV pose in a common, global reference frame as the state

$$\mathcal{X}_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T \quad (1)$$

where x, y, and θ represent longitude, latitude, and heading, respectively, in a Cartesian coordinate system and k is index for time. We consider localization of the UAV in a limited region in longitude and latitude: we assume $x_k \in [x_{min}, x_{max}]$, $y_k \in [y_{min}, y_{max}]$, $\theta_k \in [0, 2\pi)$. The localization problem is to compute the marginal posterior distribution $p(\mathcal{X}_k|\mathbf{Y}_{1:k}, \mathcal{M})$ of state $\mathcal{X}_k$, given history of measurements $\mathbf{Y}_{1:k}$ and map $\mathcal{M}$.

In this work, we concentrate on the *wake-up robot problem*, *i.e.*, at the start of a flight, we assume an uniform prior distribution $p(\mathcal{X}_0)$ over all values of x_k and y_k, across all values of heading. This represents a situation where the only initial information about the UAV position is that it is located within the area of a map defined over a rectangular area in latitude and longitude. Our formulation allows inclusion of more informed initialization.

We take the typical approach of considering localization on preacquired map as a Bayesian filtering problem (see *e.g.*, [24]). This amounts to maintaining a representation of belief of current state and updating that representation when new measurements are available. At each time step k, we obtain three types of measurements from the onboard sensors of the UAV: an odometry measurement $\mathbf{u}_k$, an heading measurement v_k, and an image from the UAV camera $\mathcal{I}_k$ to be used for map matching. Given the measurment $\mathbf{Y}_k = $

$\{\mathbf{u}_k, v_k, \mathcal{I}_k\}$, at each sampling time k, we first perform a prediction step based on the odometry measurement:

$$p(\mathcal{X}_k|\mathbf{u}_k, \mathbf{Y}_{1:k-1}) =$$
$$\int p(\mathcal{X}_k|\mathbf{u}_k, \mathcal{X}_{k-1}) p(\mathcal{X}_{k-1}|\mathbf{Y}_{1:k-1}) d\mathcal{X}_{k-1} \quad (2)$$

We then use the heading and map matching measurements as our observation model:

$$p(\mathcal{X}_k|\mathbf{Y}_{1:k}) = \frac{1}{\eta_k} p(v_k, \mathcal{I}_k|\mathcal{X}_k, \mathcal{M}) p(\mathcal{X}_k|\mathbf{u}_k, \mathbf{Y}_{1:k-1}) \quad (3)$$

where η_k is a normalizing constant. The heading and map matching measurements are considered independent, and likelihood of heading measurement is not conditional on map:

$$p(v_k, \mathcal{I}_k|\mathcal{X}_k, \mathcal{M}) = p(v_k|\mathcal{X}_k) p(\mathcal{I}_k|\mathcal{X}_k, \mathcal{M}) \quad (4)$$

IV. METHOD

In order to find the pose on a large map in presence of matching ambiguities presented in the introduction, a method is needed for utilizing a sequence of as many UAV images as needed in order to converge to a single, correct pose estimate. We propose a recursive *localization method* consisting of the following components, illustrated in Fig. 1. We use an *odometry measurement* for predicting belief of state $\bar{\mathbf{X}}_{k,xy\theta}$ at time k based on state $\mathbf{X}_{k-1}$ at previous time instant $k-1$. We use a *map matching measurement* for computing likelihood of pose hypotheses $\mathbf{W}_{M,k}$ based on a single UAV image and, optionally, a *heading measurement* for computing the likelihood of pose hypotheses $\mathbf{W}_{H,k}$ based on a compass heading measurement. The localization method updates belief of state using all the measurements, providing $\mathbf{X}_k$, belief of state at time k.

In this section we describe each of these components in detail, putting particular attention in describing how the proposed solutions target the complexity that arises from both dealing with a large map and with considerable appearance change, central limitations of current methods that we address in this work. Sec. IV-A lists all measurements we use, and Sec. IV-B specifies our localization method.

A. Measurements

1) Odometry measurement: We assume that the UAV is running a VIO algorithm which, at time k, provides $\mathbf{u}_k = \begin{bmatrix} u_{k,x} & u_{k,y} & u_{k,\theta} & u_{k,o} \end{bmatrix}^T$, a measurement of translation, rotation and distance traveled since time $k-1$. $u_{k,x}$ and $u_{k,y}$ are translation since time instant $k-1$ in x and y coordinates, respectively, with respect to pose at time $k-1$. Similarly, $u_{k,\theta}$ is rotation around vertical axis. $u_{k,o}$ is the integral of distance traveled since instant $k-1$ according to odometry. The measurements are visualized in Fig. 2a.

We approximate the odometry pose uncertainty with a multivariate normal distribution. We further assume the covariance of pose is isotropic in x, y directions and that rotation noise is independent from translation noise. More formally, we assume $\mathbf{u}_k$ has posterior density $p(\mathcal{X}_k|\mathbf{u}_k, \mathcal{X}_{k-1})$ over possible successor states $\mathcal{X}_k$, given the previous state

$\mathcal{X}_{k-1}$ and the odometry measurement $\mathbf{u}_k$ and we approximate the posterior density with a multivariate normal distribution with covariance $\boldsymbol{\Sigma}_u$:

$$p(\mathcal{X}_k|\mathbf{u}_k, \mathcal{X}_{k-1}) = \mathcal{N}(\mathcal{X}_{k-1} + \begin{bmatrix} u_{k,x} \\ u_{k,y} \\ u_{k,\theta} \end{bmatrix}, \boldsymbol{\Sigma}_u(u_{k,o})) \quad (5)$$

where $\boldsymbol{\Sigma}_u(t)$ is a diagonal matrix:

$$\boldsymbol{\Sigma}_u(t) = diag(\sigma_{u,xy}(t)^2, \sigma_{u,xy}(t)^2, \sigma_{u,\theta}(t)^2). \quad (6)$$

While the isotropic noise for translation and independent noise in heading are simplified approximations of the true distribution [25], we consider this a sufficient upper bound approximation to the odometry noise. This approximation enables a computationally fast method of prediction as elaborated in Sec. IV-B.3.

In addition, we assume the movements of the UAV produce sufficient IMU excitation to make scale observable or that the UAV is equipped with additional sensors with which scale can be resolved.

2) Map matching measurement: The purpose of map matching is to provide a means for verifying or disputing pose hypotheses, given camera image $\mathcal{I}_k$ and map $\mathcal{M}$. A method for computing the likelihood $p(\mathcal{I}_k|\mathcal{X}_k, \mathcal{M})$ is thus needed. Our work focuses especially on finding means for computing the likelihood that work well over ambiguous terrains under significant seasonal appearance difference, in a computationally efficient way. Since the method used for map matching and likelihood computation based on a UAV image is a considerable part of our contribution, a separate section (Sec. V) has been given to the detailed description of this component.

3) Heading measurement: We assume the UAV is equipped with an attitude and heading reference system (AHRS), relying on fusion of a compass and an IMU, which provides a measurement of heading, θ, with respect to map East, corrupted by Gaussian noise $n_\theta \sim \mathcal{N}(0, \sigma_v^2)$:

$$v_k = v(t_k) = \theta(t_k) + n_\theta \quad (7)$$

B. Localization method

We have to consider how to formulate the pose estimation problem in a computationally feasible way. In choosing the estimation approach, we need to consider characteristics of the problem. Specifically, as we start from a very uninformed state and expect natural ambiguities in the environment we are operating in, it is expected that before converging to the correct pose, our estimator has to be able to track a large number of multiple possible hypotheses.

1) Choosing representation for state: Typical solutions used in multimodal location estimation problems include particle filtering and point mass filtering [24]. The use of a particle filter in UAV localization is a common choice [12], [19], [21]. However, in cases of very uninformed initialization, a risk exists that a particle is initially not placed in vicinity of the true pose, leaving the probability of converging on the correct pose over time to chance. Furthermore,

during a flight, it is possible that the close proximity of true state is left without sufficient particle density in cases such as when flying for long periods of time over ambiguous areas, or in case of large but local map inconsistencies (e.g. forest clearcutting having taken place between map acquisition and flight), leading to divergence in a poorly predictable way. Since the particle filter is stochastic, different instantiations of the filter using the same data may provide different results and, depending on selected resampling scheme, computation time may vary.

Instead of representing belief through the use of particles, we choose to use a point mass filter and compute the belief on a discrete grid. This selection ensures coverage of full state space throughout the mission, at the resolution specified by our choice of grid, and offers deterministic performance and constant computational time and memory requirement. Early adaptations of point mass filtering-based approaches on 2D robot localization include the work by Burgard *et al.* [26] and in terrain navigation of flying platforms by Bergman [27].

2) Definition of state using point mass filter: We approximate belief $p(\mathcal{X})$ on the continuous state space of $\mathcal{X}$ by decomposing it into a grid of regions of equal size with resolution $r_x = r_y = r_{xy}$ in translation and r_θ in heading. Each region is a voxel $s(i,j,l) = \begin{bmatrix} s_x(i), s_y(j), s_\theta(l) \end{bmatrix}$ in the state space with bounds

$$\begin{aligned} s_{xv}(i) &\leq s_x(i) < s_{xu}(i) \\ s_{yv}(j) &\leq s_y(j) < s_{yu}(j) \\ s_{\theta v}(l) &\leq s_\theta(l) < s_{\theta u}(l) \end{aligned} \quad (8)$$

with lower bound $s_{bv}(i) = b_{min} + i \times r_b$ and upper bound $s_{bu}(i) = s_{bv}(i) + r_b$ for each axis $b \in \{x, y, \theta\}$ and $i, j, l \in \mathbb{N}$ such that the whole state space is covered.

We approximate belief over state at time instant k as a piecewise constant probability matrix $\mathbf{X}_k \approx p(\mathcal{X}_k)$ where each element of matrix $\mathbf{X}_k[i,j,l]$ assigns a probability to each voxel $s(i,j,l)$ in state space.

3) Using odometry measurement for prediction: To approximate the prediction step (2) with our state representation, using the VIO measurement model presented in Sec. IV-A.1, we formulate a method in which the probability mass contained in a voxel in $\mathbf{X}_{k-1}$ is projected to other voxels in the belief grid at time k as dictated by the odometry measurement. We repeat this operation for each voxel in the belief grid. An example visualization is shown in Fig. 2. Isotropic (x,y) odometry noise and independent θ odometry noise enable us to run the prediction computationally efficiently as three consecutive 1D convolutions.

We compute offsets $o_x(\alpha), o_y(\alpha), o_\theta$ and kernel vectors $\kappa_\theta, \kappa_x(\alpha)$ and $\kappa_y(\alpha)$ such that for a given initial heading α in the global frame, the kernels span a region of at least four standard deviations around mean. We then perform prediction by running 1D convolutions in sequence:

$$\bar{\mathbf{X}}_{k,x}[i,j,l] = \sum_{h=1}^{q_x} \mathbf{X}_{k-1}[i - o_x(\bar{s}_\theta(l)) - h, j, l] \kappa_x(\bar{s}_\theta(l))[h]$$

$$(9a)$$

$$\bar{\mathbf{X}}_{k,xy}[i,j,l] = \sum_{h=1}^{q_y} \bar{\mathbf{X}}_{k,x}[i, j - o_y(\bar{s}_\theta(l)) - h, l]\kappa_y(\bar{s}_\theta(l))[h] \tag{9b}$$

$$\bar{\mathbf{X}}_{k,xy\theta}[i,j,l] = \sum_{h=1}^{q_\theta} \bar{\mathbf{X}}_{k,xy}[i, j, l - o_\theta - h]\kappa_\theta[h] \tag{9c}$$

Here, $\bar{s}_\theta(l)$ returns the centerpoint of the angle corresponding with index l in $\mathbf{X}$. Note that the offsets and convolution kernel vectors in x and y direction depend on the value of α, *i.e.*, probability mass is shifted in the direction defined by the heading value, using the nominal value of that cell. At the edges of state space, (9a) and (9b) are filled with zero for values outside state space and (9c) is wrapped around to the opposite edge.

The end result of this step is $\bar{\mathbf{X}}_{k,xy\theta}$, which states how $\mathbf{X}_{k-1}$ shifted and spread according to odometry measurement and odometry noise from time $k-1$ to k. A clarifying visualization can be found in Fig. 2.

4) Weighing belief with heading measurement: We approximate the circular Gaussian presented in Sec. IV-A.3 by von Mises distribution and we compute a weight matrix $\mathbf{W}_{H,k}$ for all grid indices:

$$\mathbf{W}_{H,k}(i,j,l) = \Phi_\mathcal{V}(s_{\theta u}(l), v_k, 1/\sigma_v^2) - \Phi_\mathcal{V}(s_{\theta v}(l), v_k, 1/\sigma_v^2) \tag{10}$$

where $\Phi_\mathcal{V}(t, \theta, 1/\sigma^2)$ is the cumulative density function of von Mises distribution with parameters θ, $1/\sigma^2$ evaluated at t.

5) Weighing belief with map matching measurement: For all grid indices, we compute a weight matrix $\mathbf{W}_{M,k}$, from likelihood of the observation $\mathcal{I}_k$ representing the voxel $s_x(i), s_y(j), s_\theta(l)$, given map $\mathcal{M}$:

$$\mathbf{W}_{M,k}(i,j,l) = p(\mathcal{I}_k|s_x(i), s_y(j), s_\theta(l), \mathcal{M}) \tag{11}$$

The likelihood computation methods are described in detail in Sec. V-D.

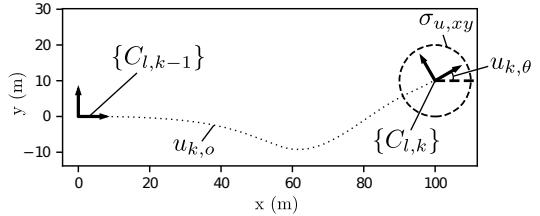*6) Updating with all measurements:* Our updated state estimate is computed as

$$\mathbf{X}_{k,u} = \bar{\mathbf{X}}_{k,xy\theta} \odot \mathbf{W}_{H,k} \odot \mathbf{W}_{M,k}, \tag{12}$$

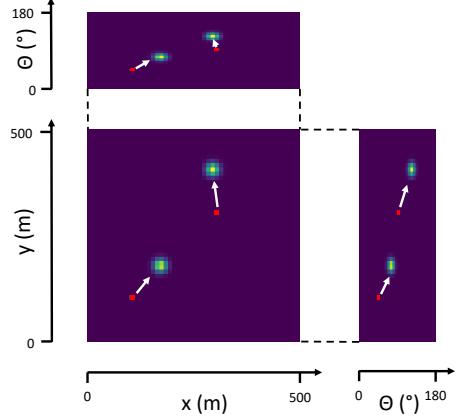where $\odot$ is elementwise multiplication, and finally normalized:

$$\mathbf{X}_k = \frac{\mathbf{X}_{k,u}}{\sum_{i,j,l} \mathbf{X}_{k,u}(i,j,l)} \tag{13}$$

The end result is a recursive discrete approximation of equation (3).

7) Interval for running algorithm: Our localization algorithm is run at fixed intervals of travel, when the UAV has traveled more than a specified distance u_l since the latest update. The amount of travel since latest update is approximated by odometry. The value for u_l can be chosen to balance computational load, ensure enough movement with respect to to selected grid size, and have independence between UAV images used in map matching.

(a) Odometry measurements are stated with respect to frame at previous update, $\{C_{l,k-1}\}$. $u_{k,o}$ states distance traveled since previous update, according to odometry.

(b) Part of $\mathbf{X}$ after marginalizing different axes individually. Two voxels at $(100, 100, 45°)$ and $(300, 300, 90°)$ representing belief before prediction are shown in red. Voxels highlighted in green and yellow represent belief after prediction. White arrows show how prediction using these odometry measurements with the assumed noise shifts and smooths $\mathbf{X}$ according to the measurement, and that shifts are performed in direction determined by odometry measurement and voxel's θ value.

Fig. 2: Example of prediction based on odometry. In this example, $u_{k,x} = 100$ m, $u_{k,y} = 10$ m, $u_{k,\theta} = 30°$, $\sigma_{u,xy} = 10$ m, $\sigma_{u,\theta} = 5°$. Fig. 2a visualizes odometry measurements between times $k-1$ and k. Fig. 2b visualizes how odometry measurements are used in prediction.

V. MAP MATCHING

Our formulation this far has considered how to fuse odometry, heading and map matching measurements using a point mass filter. We have still to define how to compute the likelihood $p(\mathcal{I}_k|\mathcal{X}_k, \mathcal{M})$, *i.e.*, assess how well the observed UAV image $\mathcal{I}_k$ supports each possible value of state $\mathcal{X}_k$, when using map $\mathcal{M}$.

Instead of detecting image features and using them as landmarks as done in *e.g.*, [28], we prefer an area-based approach, *i.e.*, using a large part of the observed image for discriminating between plausible and incorrect poses. The main motivation for this choice is that feature-based approaches require detection of spatially local features, which may be very sparsely detectable when flying over ambiguous terrains, especially across significant seasonal change (*e.g.*, after accumulation of snowfall) and when image footprint on ground is small. Using a large section of UAV image allows

us to assess plausibility of matching with respect to a map even if locally distinct features cannot be detected.

Besides the targeted robustness when flying in ambiguous terrains, there are a number of other desirable characteristics for a map matching method. The method should work in matching patches of ground that are of a reasonable size; we should not develop a solution which requires a very large observed ground footprint and thus a high flying altitude in order for the solution to work. As we don't want to restrict the UAV trajectories, it is desirable to have a map matching method that is tolerant to viewpoint change (especially camera pitch and roll) and works at different flight altitudes and across a range of camera intrinsics. In addition, in developing a map matching approach trained with data samples, we cannot assume that a large amount of UAV imagery containing all types of expected variability (seasonal apperance change, camera angle with respect to ground and camera intrinsics, flight altitudes) is available for this purpose. Finally, the approach should be such that it works in the targeted scale and is computationally feasible for an onboard deployment. As a synthesis of all these needs, we propose an approach that splits the problem in two.

We first perform a rudimentary orthoprojection of the observed UAV image such that a section of the image that corresponds to a patch on ground of a specific size, at specific ground sampling distance, is generated (see Fig. 3). This acts as a means for abstracting away the flight altitude, camera intrinsic parameters and camera orientation with respect to ground and renders the problem of likelihood computation into that of finding likelihood between patches of orthoimages. This is beneficial also in training deep learning-based matching methods that are robust against seasonal variance as we demonstrated in earlier work [29], since our method abstracts away parameters that are irrelevant for the map matching problem (*i.e.*, camera intrinsics and altitude differences) and, unlike UAV image datasets, satellite images containing seasonal variation are plentiful.

As a second step, we compute a compact descriptor vector from the UAV image patch. We compute likelihood of each pose hypothesis by comparing that descriptor vector to a set of descriptor vectors that have been precomputed from map $\mathcal{M}$. Likelihood estimation is done by computing distance in the embedding space spun by the descriptor vectors. The choice of operating on compact descriptor vectors instead of template matching individual pose hypotheses as *e.g.*, in previous work [29] is a key enabler for fast likelihood estimation over a large map, but raises the question of how to engineer a descriptor vector computation method in a way that allows as small vector size as possible while enabling robust localization.

In this section, we first introduce the orthoprojection method, after which we describe the methods of computing descriptor vectors, precomputing a map and finally computing the matching likelihood.

A. *Orthoprojecting UAV image*

The map matching measurement is generated based on the view from a single UAV image, which may be tilted from nadir and thus contains perspective difference with respect to top-down view. To reduce the impact of this perspective change, we orthoproject the UAV image and make the assumption that the ground beneath the UAV is planar. Our approach resembles earlier work [30] with a few key differences and we report an overview of the full approach for completeness.

Based on direction of gravity estimated by AHRS, we first define local frame $\{C_{l,k}\}$ for image sampled at time t_k whose origin is at the origin of the UAV camera frame, z axis points in opposite direction to gravity, and the component of camera image plane horizontal axis perpendicular to z is aligned with y axis.
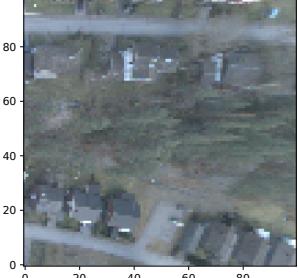
Our localization approach is based on using an orthoprojection of the camera view of the UAV. We detect image feature points using Shi-Tomasi detector [31] and track movement of the features across a batch of ten consecutive camera image frames around keyframe sampled at t_k using a pyramidal Lucas-Kanade tracker [32]. Besides image sampled at t_k, the batch consists of four images prior to the image corresponding with time t_k and five after. We estimate the 3D locations of tracked feature points in frame $\{C_{l,k}\}$ using the linear triangulation method in [33], using relative pose transformations between frames in batch that we compute from ground truth data. Exploration of VIO frontends is beyond the focus of this work and we believe the use of noiseless relative transformations is a sufficient approximation of a generic VIO algorithm over the sequence of ten frames.

We then find the best-fitting plane whose normal is aligned with z axis; *i.e.*, we assume the ground below the UAV is planar and horizontal. We find a square with size 100m by 100m lying on the best-fitting plane that is closest to nadir and fully visible in the camera image $\mathcal{I}_k$. We project the cornerpoints of this square into the image plane of $\mathcal{I}_k$ and, by homograpy, transform the pixels corresponding with the square in $\mathcal{I}_k$ to an 100px by 100px image at 1 m/px resolution which we call I_k. I_k is the observation we use in map matching. We also compute $T_{l,k}^{s,k}$, a transformation from frame $\{C_{l,k}\}$ to a frame centered in the middle of the square, which we label $\{C_{s,k}\}$. There is no rotation between $\{C_{l,k}\}$ and $\{C_{s,k}\}$ and translation is defined such that $\{C_{s,k}\}$ is at the center of the square. An example of one orthoprojected UAV image and a visualization of the coordinate frames is shown in Fig. 3.
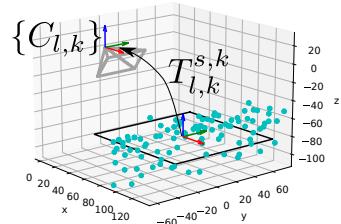
The presented approach enables us to consider the motion of a UAV as a sequence of 2D translations and rotations while allowing extraction of observation I_k at correct scale independent of fight altitude, while $T_{l,k}^{s,k}$ contains information about position of its centerpoint relative to camera, including estimated ground plane altitude.

(a) Original UAV image $\mathcal{I}_k$

(b) Orthoprojected image I_k

(c) Coordinate frame $\{C_{l,k}\}$, points tracked in VIO (cyan dots), square lying on plane fit to VIO landmarks and transformation $T_{l,k}^{s,k}$ to $\{C_{s,k}\}$
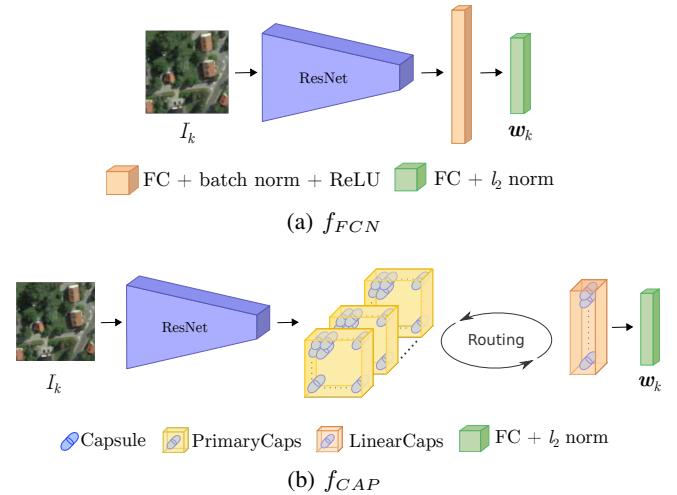
Fig. 3: Example of UAV image (100m by 100m area used for orthoprojection highlighted in red), its orthoprojection, and a visualization of the coordinate frames.

FC + batch norm + ReLU $\quad$ FC + l_2 norm

(a) f_{FCN}

Capsule $\quad$ PrimaryCaps $\quad$ LinearCaps $\quad$ FC + l_2 norm

(b) f_{CAP}

Fig. 4: Architectures for fully connected and capsule networks

B. From orthoimage patch to descriptor vector

Inspired by other works projecting the UAV observation into a single compact descriptor vector [10], [21], [22], to provide a fast way to compare an observation to a large number of pose hypotheses, our map matching method is based on transforming both UAV observations and a reference map into a suitable descriptor space, and computing probability of pose hypotheses using descriptor vector values. Based on image I_k, we compute a descriptor vector $\mathbf{w}_k$ using function f_i: $\mathbf{w}_k = f_i(I_k)$. This function f_i is composed of a deep neural network backbone b such as Resnet [34] followed by a projection module m. Therefore $f_i(I_k) = m_i(b_i(I_k))$. The projection module is stacked on top the ResNet model where the the last average pooling and final fully connected layer specified in [34] are removed. The last layer of the projection module has D neurons in order to compute D-dimensional l_2-normalized descriptor vectors. The projection model thus produces vectors in unit D-sphere. Therefore, $f : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^D$, where h, w and c are the height, width and channels dimensions of the input image I_k.

1) Projection modules: In order to extract D-dimensional embedding vectors, we used two different types of projection modules: one composed only of fully connected layers, m_{FCN}, and one composed by a Capsule Network (CapsNets) model [35], m_{CAP}. We refer to f_{FCN} and f_{CAP} as the models where the ResNet backbone is followed by m_{FCN} and m_{CAP}, respectively.

Inspired by the architecture choices proposed by [21], [22], the m_{FCN} module is composed by two fully connected layers, one with N neurons and one with D outputs. A visualization of the network structures is shown in Fig. 4a.

CapsNets have gained great attention recently since they are more robust to input perturbations compared to other convolutional neural network (CNN) architectures with a similar number of trainable parameters [35], [36]. Their main innovation lies in two major distinctions from the CNNs: (i) the encoding of object poses (position, size, orientation) and visual attributes (*e.g.*, color, texture, deformation, hue) into groups of neurons called *capsules* (ii) the routing-by-agreement mechanism, which models the connections between capsules of different layers. Namely, it models the part-whole relationships among objects without losing spatial information. We explore the use of CapsNets as projection modules for two main reasons: first, CapsNets are well known for being robust to affine transformations and novel viewpoints; secondly, they have also been shown to achieve higher generalization with less data compared to CNNs [35], [36]. Owing to these properties, extracting descriptor vectors using f_{CAP} can help to solve the UAV localization task described in this work. Furthermore, in the literature [37], [38], capsule layers have been widely stacked on top of ResNet backbones instead of traditional fully connected layers to achieve better performance. A visualization of f_{CAP} is shown in Fig. 4b.

C. Precomputing descriptors for map

We precompute offline a map around the expected operating region. The map holds descriptor vector values that have been computed from a georeferenced orthophoto RGB bitmap $\mathcal{M}$. For each map coordinate $X_h = (x_h, y_h, \theta_h)$, we crop a w m by w m square image $I_{\mathcal{M},h}$, translated from origin by (x_h, y_h) in map coordinates and rotated by θ_h, from the map image $\mathcal{M}$ and scale it to 1 m/px resolution. We then compute an embedding vector $\mathbf{w}_h = f_i(I_{\mathcal{M},h})$. We compute these embedding vectors using the centerpoint coordinate of each voxel in $\mathbf{X}$. This process yields a precomputed map $\mathbf{M}(i,j,l) \in \mathbb{R}^D$ where indices i, j, l correspond with indices of grid cells in our belief representation $\mathbf{X}$. We compute a

separate map for each tested network architecture.

D. Determining map matching likelihood

To assess whether a UAV observation corresponds with a state hypothesis, we approximate

$$p(\mathcal{I}_k|s_x(i), s_y(j), s_\theta(l), \mathcal{M}) \approx p(\mathbf{w}_k|i, j, l, \mathbf{M}) \quad (14)$$

and compute $\mathbf{W}_{M,k}$, which contains the weight for each state space element on the chosen voxel grid.

$$\mathbf{W}_{M,k}(i, j, l) = p(\mathbf{w}_k|i, j, l, \mathbf{M}) \quad (15)$$

We compare two solutions for this. The first one, labeled *linear*, is similar to the choice in [21]. It assumes that Euclidean distance between the embedding obtained from the UAV and from the map $c_k(i, j, l) = \|\mathbf{w}_k - \mathbf{M}(i, j, l)\|_2$ is inversely proportional to the probability of correct pose:

$$\mathbf{W}_{M1,k}(i, j, l) = \frac{2 - c_k(i, j, l)}{2} \quad (16)$$

The second one is the method for computing importance factor presented in earlier work [29], where we estimate the probability density of distances in Euclidean space for true and false matches from satellite image data and compute the probability that the observation is from "match" class, for each element in $\mathbf{X}_k$ individually. We label this weighing method *bayesian* and we name weight matrix $\mathbf{W}_{M2,k}$.

VI. EXPERIMENTS

A. Overview of experiments

We experiment the performance of our solution with respect to baseline methods on real-world datasets. We evaluate probabillity of convergence, time to convergence and localization error after convergence with flights taking place in two areas in Sweden. In both areas, we experiment with the problem of localization starting from 100 km^2 uncertainty. In addition, we experiment with real-time implementation on a UAV onboard computer.

For evaluating the critical design choices in our map matching approach, we vary the projection module type (fully connected or capsule network), likelihood vector dimensionality D (8, 16, 32 or 128) and likelihood conversion method (linear or bayesian), evaluate the impact of these choices on probability of convergence, time to convergence and mean localization error after convergence.

We evaluate localization performance by the criteria defined in Sec. VI-B. The datasets we use for experimentation and model training are described in Sec. VI-C. Training methods are outlined in Sec. VI-D.1, baseline methods are described in Sec. VI-E, followed by evaluation of localization performance in Sec. VI-F and learnings from real-time experiments in Sec. VI-G.

B. Evaluation criteria in localization experiments

1) Translation error: We compute the estimated (x, y)-coordinates and heading using (17a) and (17b), respectively.

$$\widehat{X}_{s,k}^{xy} = \sum_{i,j,l} \mathbf{X}(i, j, l) \begin{bmatrix} \bar{s}_x(i) & \bar{s}_y(j) \end{bmatrix}^T \quad (17a)$$

$$\widehat{X}_k^\theta = atan2(\sum_{i,j,l} \mathbf{X}(i, j, l) sin(\bar{s}_\theta(l)), \sum_{i,j,l} \mathbf{X}(i, j, l) cos(\bar{s}_\theta(l))) \quad (17b)$$

Here, $\bar{s}_x(i)$, $\bar{s}_y(j)$ and $\bar{s}_\theta(l)$ are the centerpoint coordinates of voxel corresponding to indices (i, j, l) and $atan2$ is the 2-argument arctangent. We then transform these estimates from $\widehat{X}_{s,k}^{xy}$ to drone-centric coordinates $\widehat{X}_k^{xy}$.

We compute the Euclidean distance to ground-truth (x, y)-coordinates $X_{k,gt}^{xy}$:

$$\widetilde{X}_k^{xy} = \|\widehat{X}_k^{xy} - X_{k,gt}^{xy}\|_2 \quad (18)$$

We also compute $\sigma_{\widehat{xy}}$, the weighted standard deviation of Euclidean distances to $\widehat{X}_k^{xy}$ in (x, y) plane, weighing with $\mathbf{X}_k$. We claim that $\sigma_{\widehat{xy}}$ is a suitable measure of convergence of the localization solution, and a large spread of uncertainty signals the need for re-initialization of the estimator.

C. Datasets

1) Satellite images: We trained our networks on 100m by 100m samples at 1 m/pixel randomly drawn from Google Earth satellite images of size 4800 by 2987 meters collected from 9 regions, in arbitrarily selected places in southern Finland that cover urban and non-urban areas. For each region, 4 to 15 satellite images collected from the same area at different times, containing seasonal variation, are used. The Google Earth datasets are the same as in an earlier work [29].

2) UAV images: Our experiments are run on datasets that have been collected with a UAV in two locations in Sweden at different times[1]. The dataset contains posed images sampled at 10 Hz. Ground-truth camera poses are fused from real-time kinematic (RTK)-corrected global positioning system (GPS) in uninterfered conditions and IMU measurements with a proprietary algorithm. The use of RTK-GNSS ensures position precision in the centimeter range. A listing of the flight experiments is given in Table I. A representative image of each dataset is shown in Figure 5 and flight trajectories over an orthophoto map are shown in Figure 6. The flights take place over terrains with forest areas, a lake, agricultural fields and some residential areas. At the time of running this experiment, the original IMU data was not available.

3) Maps used for localization: Each original map $\mathcal{M}$ is an orthophoto bitmap constructed from aerial images taken over the operating area in summer 2021. We use orthophoto bitmaps with an original ground sampling distance[2] of 0.16 m/px provided by a local map information supplier[2].

---

[1]Data provided by Saab Dynamics Ab.
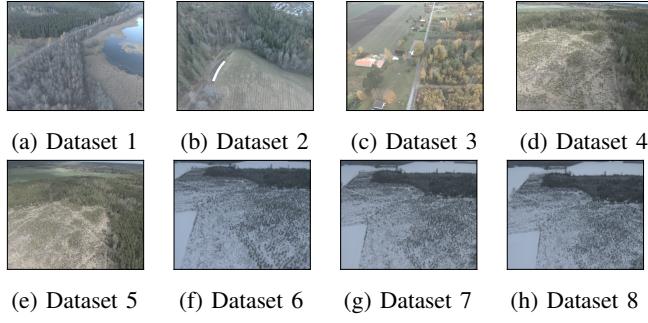[2]© Lantmäteriet, https://www.lantmateriet.se/.

(a) Dataset 1   (b) Dataset 2   (c) Dataset 3   (d) Dataset 4

(e) Dataset 5   (f) Dataset 6   (g) Dataset 7   (h) Dataset 8

Fig. 5: Example images from datasets. Note difference in seasonal appearance.
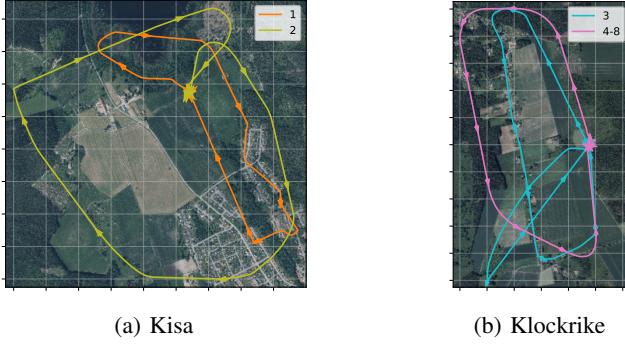

(a) Kisa   (b) Klockrike

Fig. 6: Flight trajectories of each dataset. Grid spacing 200 m. Datasets 4-8 have same trajectory with varying altitude and season.

D. Training descriptor generator networks

1) Training details: We trained the models f_{FCN} and f_{CAP} for 500 epochs only on satellite images.[3] At each epoch, 45k training satellite samples are generated from the 9 areas. Each sample represents a location and orientation on a map. We draw 5 images corresponding with each location and orientation. For each batch, we randomly select 10 locations. As deep metric learning for encoding the images into low dimensional vectors, we employed the triplet loss [39] with batch-all strategy and margin 0.2, using

[3]We acknowledge the computational resources provided by the Aalto Science-IT project.

TABLE I: Characteristics of flight datasets. Trajectory lengths are computed along (x, y) plane, and camera angles between nadir and camera principal axis. Altitude is with respect to starting position.

#	Area	Date	Traj. length (km)	Alt. (m)	Median camera angle [range] (°)
1	Kisa	2019-11-07	4.1	91	59.0 [56.9 …67.7]
2	Kisa	2019-11-07	6.1	92	52.1 [49.8 …56.0]
3	Klockrike	2019-10-18	6.8	92	51.2 [49.9 …53.9]
4	Klockrike	2020-04-29	4.8	53	59.3 [57.8 …60.5]
5	Klockrike	2020-04-29	4.8	92	59.9 [58.4 …61.3]
6	Klockrike	2021-01-19	4.9	54	58.9 [58.0 …60.4]
7	Klockrike	2021-01-19	4.7	71	58.7 [57.7 …60.0]
8	Klockrike	2021-01-19	4.9	84	58.4 [56.9 …60.9]

vectors representing the same location and orientation as positive samples, and vectors representing other locations and orientations in the batch as negative samples. In addition to random selection of locations and orientations, we add a random translation offset of 0 to 35 meters with uniform distribution to each location, with the intent to add robustness for observations that do not align perfectly on the map grid. With similar motivation, we add normally distributed random rotations with standard deviation of 6 degrees to the orientation of each sample before extracting the image patch from the satellite image. In addition to these, we augment the samples with Gaussian noise, motion blur, random brightness and contrast changes, and hue and saturation changes to add tolerance for changes in imaging conditions. We used Adam optimizer [40] with learning rate 10^{-6}.

2) Architecture details of neural networks: We employed a Resnet-50 model implemented in PyTorch 1.8.0 as backbone b. This network is pretrained on Places365 dataset [41]. As regards the projection module m_{FCN}, we used a fully connected layer of sizes 1024 and D, respectively. We experiment with vector length $D \in \{8, 16, 32, 128\}$. On the other hand, with regards to the module m_{CAP}, the PrimaryCaps layer is a capsule layer with 64 types of 16-dimensional capsules, which are obtained from a convolutional layer of 1024 1×1 kernel size filters. The LinearCaps layer consists of 32 32-dimensional capsules extracted running 3 iterations of the routing algorithm. We stacked on top of the LinearCaps layer a fully connected layer of size D. With the capsule projection module, we explore $D \in \{8, 16, 32\}$. For both projection modules, the D-dimensional output of the last layer is used to produce the l_2-normalized embedding vectors.

E. Comparison methods

We implement two methods to work as comparison methods of map matching for our approach. To provide comparable results, we utilize the same point mass filter implementation, same odometry measurements, prediction method, heading weighing method and same measurement images I_k for all methods and only replace the map matching solution with their approach.

As baseline methods for map matching, we use the solution by Mantelli *et al.* [19], whose formulation is scalable to large maps. To provide comparable results, we compute the descriptor vectors $\mathbf{w}_k = f_i(I_k)$ such that $\mathbf{w}_k$ is the abBRIEF descriptor. In a similar manner, we precompute a grid of abBRIEF descriptors from $\mathcal{M}_b$, using equal grid spacing as with our methods, and with abBRIEF, we compute similarity as specified in [19] and label this similarity computation method *mantelli*.

To provide a reference to a method that operates on semantic maps and allows very compact map representation over large areas, we implement the map matching solution proposed in Choi *et al.* [18]. We trained a U-Net [42] network on the Massachusetts Buildings Dataset [43] to segment buildings in the input images. We employ this network to extract the invariant feature descriptors introduced by Choi

et al. [18] on satellite and UAVs images: in this setting the feature vectors $\mathbf{w}_k$ are built upon building ratio information.

F. Localization performance

1) Experimental setting: We perform all localization experiments on datasets that contain images collected with a UAV. In all experiments, we update our belief after at least $u_l = 50$ m of travel have occurred since the previous update. In all experiments, we use a map grid with resolution $r_{xy} = 10$ m, $r_\theta = 6°$.

2) AHRS and VIO measurements: The datasets used in experiments do not contain IMU or magnetometer measurements and thus we have to simulate them. We simulate odometry measurements by random sampling from distribution (5). The translation standard deviation $\sigma_{u,xy}$ is approximated as 0.05 m per meter of travel and heading standard deviation $\sigma_{u,\theta}$ as $0.15°$ per meter of travel, which approximately correspond with the performance of VIO algorithms reported in literature [4].

For heading, we sample from the distribution specified in (7). Manufacturers of compact commercial AHRS sensors typically report RMS error of $2°$ in heading [44], [45]. In all experiments with AHRS, we simulate the heading measurement from ground-truth orientation data and assume $\sigma_v = 3°$.

3) Evaluating localization performance: We evaluate localization performance for each step k after completing prediction, map matching and AHRS update at that step. We define that a localization solution has converged when the translation standard deviation $\sigma_{\widehat{xy}}$ is less than 100 m. We compute the mean number of updates to convergence $\bar{k}_c$ and mean translation error in converged state, $\bar{\widetilde{X}}_c^{xy}$, for each tested model and likelihood conversion method. We compute the proportion of flights where each compared solution converges, p_c, and tabulate results in Table II. In addition, we visualize the translation error and standard deviation of (x, y) translation in each case in Fig. 7.

4) Statistical significance testing: To explore what significance our results show in the various parameters we have used in the experiment configurations, we run a type II analysis of variance test on time to convergence and on localization error after convergence. We consider the embedding dimension (8, 16, 32, 128), projection module type (fully connected or capsules) and likelihood computation method (linear or bayesian) and combinations of these parameters. We use $p = 0.05$ as limit for significance.

For time to convergence, we reject null hypothesis that chosen likelihood method is not significant ($p = 4.2 * 10^{-16}$). For localization error after convergence, we reject null hypothesis that the parameter is not significant for embedding dimension ($p = 2.5 * 10^{-14}$), projection module type ($p = 2.1 * 10^{-2}$), likelihood method ($p = 3.7 * 10^{-4}$) and combination of embedding dimension and projection module type ($p = 5.0*10^{-5}$) and combination of embedding dimension and likelihood method ($p = 1.3 * 10^{-2}$).

TABLE II: Probability of convergence p_c, time to convergence $\bar{k}_c$ and mean localization error after convergence $\bar{\widetilde{X}}_c^{xy}$ when using our methods with various design choices and when comparing to reference method.

Model type	Likelihood conversion	p_c	$\bar{k}_c$	$\bar{\widetilde{X}}_c^{xy}$ (m)
Ours, Caps, D=8	Linear	0.875	60.9	15.7
Ours, Caps, D=16	Linear	0.875	56.3	13.3
Ours, Caps, D=32	Linear	0.875	58.9	13.9
Ours, FCN, D=8	Linear	0.75	65.2	21.1
Ours, FCN, D=16	Linear	0.875	61.1	13.9
Ours, FCN, D=32	Linear	0.875	63.1	12.9
Ours, FCN, D=128	Linear	0.875	62.3	11.2
Ours, Caps, D=8	Bayesian	1.0	44.4	18.3
Ours, Caps, D=16	Bayesian	1.0	35.2	12.8
Ours, Caps, D=32	Bayesian	1.0	30.8	15.0
Ours, FCN, D=8	Bayesian	0.875	43.0	18.7
Ours, FCN, D=16	Bayesian	1.0	36.1	14.6
Ours, FCN, D=32	Bayesian	1.0	33.2	15.7
Ours, FCN, D=128	Bayesian	1.0	23.2	12.6
BRM	Linear	0.0	N/A	N/A
BRM	Bayesian	0.0	N/A	N/A
abBRIEF	Mantelli	0.625	63.8	4112.2
abBRIEF	Bayesian	0.0	N/A	N/A

G. Real-time experiment

To understand the applicability of our solution to an embedded system, we implemented a version of our algorithm running in real time on an embedded computer on an example UAV. We use the Nokia Drone Networks drone (see Figure 8) carrying a camera gimbal. The drone was equipped with an NVidia Jetson Nano computer on which the localization algorithm was run. The algorithm was implemented in Python and we used ROS [46] for inter-process communication. To accommodate the smaller observable ground footprint due to the narrower field of view of the drone camera at the selected flight altitude and observation parameters (50 m altitude at 45 degree camera pitch), we retrained a model with Resnet50 + FCN, D=16 architecture to work on 40m by 40m images at 1 m/px resolution. The size of the operating region and map was 1.62 km by 3.82 km. We used VINS-Mono [47] as the VIO algorithm. We used the output of a non-GNSS-corrected AHRS algorithm implemented on the drone flight controller for heading updates.

The mean network inference time (run on CPU) was 1.02 s, mean prediction time was 0.83 s, map matching took 1.18 s and AHRS update took on average 0.12 s, and all steps took on average 3.15 s. The algorithm was configured to perform an update every 40 meters of travel, while the drone was flying at 5 m/s. There is room for speedup by proper parallelization of the algorithm. The algorithm was run fully onboard the UAV.

Some localization errors appeared over terrain patches with very uniform colouring due to scale drift and rotation estimation errors of the VIO algorithm which are not considered in our odometry noise model. In addition, the AHRS heading estimate from the flight controller appeared to contain non-Gaussian heading errors up to 15 degrees; to
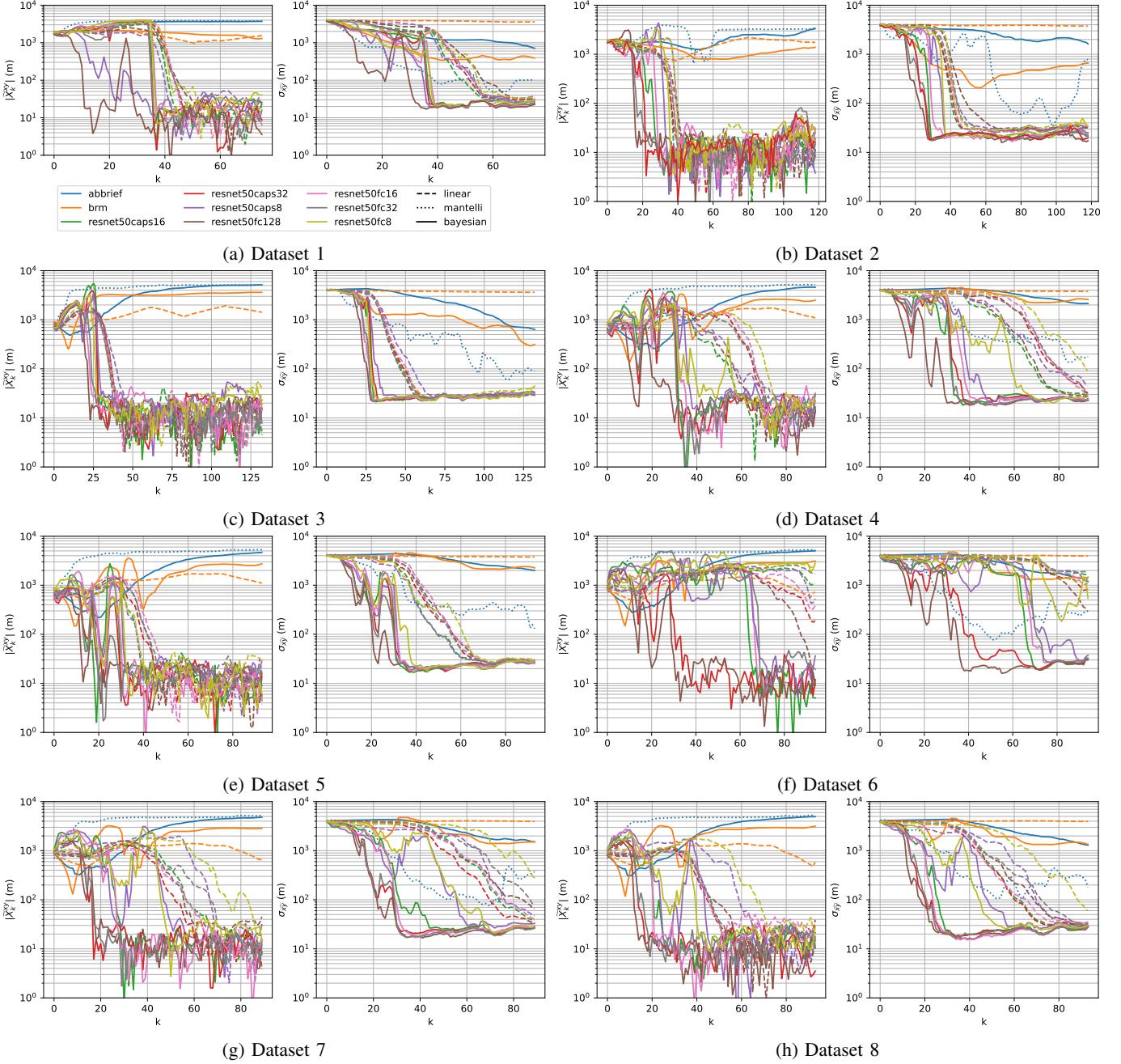
Fig. 7: Translation error $|\widetilde{X}_k^{xy}|$ and standard deviation of translation $\sigma_{\widehat{xy}}$ as function of update index k in different datasets using all methods. Line color represents descriptor computation method, line style (dashed, dotted, solid) represents likelihood computation method. Logarithmic scale. Updates are made approximately every 50 meters of travel.

accommodate this, we used $\sigma_v = 60°$ in this experiment.

The memory footprint of the precomputed map in this experiment was 460.8 MB. On a 100 km^2 map, the precomputed map memory requirement for D=8, D=16, D=32 and D=128 are 3.5 GB, 7.0 GB, 14.0 GB and 55.9 GB, respectively, setting further practical constraints for embedded implementation at very large scale.

This experiment shows that the algorithm can be run in real time on an embedded computer carried by a drone in an operating region of typical size for drone operations.

Fig. 8: UAV used in real time experiment

VII. Discussion

To solve the wake-up robot problem at the presented scale, we derived a solution based on a recursive point mass filter instead of the more common particle filter approach. We believe this avoids issues with particle depletion, which is important in cases of significant initial uncertainty, ambiguity of terrain and potential intermittent but not random correspondence mismatches between observations and maps.

An architecture where an image observation is projected into an embedding space, with a major reduction in data dimensionality, is a key enabler for fast and memory-efficient similarity comparisons over a large number of hypotheses. Our approach for projecting data from different source domains (*i.e.*, UAV camera images and orthophoto maps or satellite images) into a common domain enables the use of learned descriptors. With our approach, extensive training data covering all expected variation is required in only one source domain without need for labeling of data. Our approach of using a learnable embedding appears to be efficient for localization over large areas containing natural and built environments, in scenarios where also significant seasonal appearance change occurs between flights. Other existing methods, *e.g.*, a handcrafted descriptor approach (abBRIEF) or learned description trained to detect pre-specified semantics (BRM), do not converge at this scale.

Bayesian likelihood conversion showed the greatest effect in time to convergence, in comparison to the more common linear approximation. The results in Tab. II appear to hint at the possibility that increasing embedding dimensionality leads to faster convergence, but further experimentation would be required to verify statistical significance. For localization error afer convergence, embedding dimensionality was found to be one of the statistically significant parameters. Results tabulated on Tab. II and error plots in Fig. 7 seem to suggest that there may be a lower bound on localization error that appears independent from dimensionality and is most likely a result of other design choices and the characteristics of the operating environment. In other words, it appears that low-dimensionality descriptors are an efficient way of expressing what is important for localization and dimensionality of description is not the main hindering factor, what comes to localization error.

In order to extract low-dimensionality descriptors, we tested both traditional fully connected and capsule layers. Our hypothesis was initially that capsule layers would improve localization performance, extracting better encodings thanks to their ability to model part-whole relationships and robustness to novel viewpoints. Employing capsule layers lead to a slight improvement in error after convergence in comparison to fully connected layers while using less trainable parameters, as also stated in [35]. In fact, for 16-dimensional embeddings, with fully connected layer, the network has 57M trainable parameters, while with capsule layers, it has 42M trainable parameters. Since this improvement is only marginal, we did not conduct experiments with 128-dimensional capsule embeddings, as CapsNets are also well known to be computationally very demanding in terms of memory consumption, training and inference times.

Experiments demonstrate that the proposed approach is suitable for real time implementation on a flying platform at a typical scale of UAV operations. Tailoring design parameters allows the implementation of the solution on a very resource-constrained platform and running it in real time together with an odometry system.

To enable error recovery, the solution provides a measure of position uncertainty by computation of standard deviation. Upon detection of prolonged high standard deviation, this enables the triggering of reinitialization of the pose estimator. The ability to detect localization failures and recover from the loss of location information on a large scale paves way for a failure-aware, failsafe UAV localization system.

VIII. Conclusions

We have shown that the approach utilizing our map matching method together with a point mass filter is able to resolve UAV pose even in the case of highly uninformed initialization corresponding to a map size of 100 km^2, in conditions of significant seasonal appearance change between UAV image and map, even when flying over areas with natural ambiguity. The proposed solution converges to a localization error of 12.6–18.7 m on average in 23.2–44.4 updates, depending on the chosen architecture, while reference methods are not able to converge to the correct pose under the same circumstances. All of these contributions show that real-time localization is possible on a large scale. Going beyond the demonstrated 100 km^2 will require being able to represent even larger hypothesis spaces. Addressing this challenge will potentially require future work in hierarchical models in order to retain the favorable characteristics of high spatial accuracy, extreme initial uncertainty, and complete coverage of the hypothesis space.

References

[1] M. L. Psiaki and T. E. Humphreys, "Gnss spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[2] R. Mason, J. Bonomo, T. Conley, R. Consaul, D. R. Frelinger, D. A. Galvan, D. A. Goldfeld, S. A. Grossman, B. A. Jackson, M. Kennedy, V. R. Koym, J. Mastbaum, T. L. Nguyen, J. Oberholtzer, E. M. Pint, P. Rockstroh, M. Shostak, K. D. Stanley, A. Stickells, M. J. D. Vermeer, and S. M. Worman, *Analyzing a More Resilient National Positioning, Navigation, and Timing Capability*. Santa Monica, CA: RAND Corporation, 2021.

[3] D. Scaramuzza and Z. Zhang, *Aerial Robots, Visual-Inertial Odometry of*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1007/978-3-642-41610-1_71-1

[4] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2502–2509.

[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[6] Y. Xu, L. Pan, C. Du, J. Li, N. Jing, and J. Wu, "Vision-based uavs aerial image localization: A survey," in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, ser. GeoAI'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 9–18. [Online]. Available: https://doi.org/10.1145/3281548.3281556

[7] A. Couturier and M. A. Akhloufi, "A review on absolute visual localization for uav," *Robotics and Autonomous Systems*, vol. 135, p. 103666, 2021.

[8] H.-P. Chiu, A. Das, P. Miller, S. Samarasekera, and R. Kumar, "Precise vision-aided aerial navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 688–695.

[9] B. Patel, T. D. Barfoot, and A. P. Schoellig, "Visual localization with google earth images for robust global pose estimation of uavs," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6491–6497.

[10] M. Bianchi and T. D. Barfoot, "Uav localization using autoencoded satellite images," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1761–1768, 2021.

[11] H. Goforth and S. Lucey, "Gps-denied uav localization using pre-existing satellite imagery," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2974–2980.

[12] M. Shan, F. Wang, F. Lin, Z. Gao, Y. Z. Tang, and B. M. Chen, "Google map aided visual navigation for uavs in gps-denied environment," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 114–119.

[13] S. Chen, X. Wu, M. W. Mueller, and K. Sreenath, "Real-time geo-localization using satellite imagery and topography for unmanned aerial vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2275–2281.

[14] Y. Li, D. Yang, S. Wang, H. He, J. Hu, and H. Liu, "Road-network-based fast geolocalization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 7, pp. 6065–6076, 2021.

[15] S. J. Dumble and P. Gibbens, "Airborne vision-aided navigation using road intersection features," *Journal of Intelligent & Robotic Systems*, vol. 78, pp. 185–204, 2015.

[16] A. Volkova and P. W. Gibbens, "More robust features for adaptive visual navigation of uavs in mixed environments," *Journal of intelligent & robotic systems*, vol. 90, no. 1, pp. 171–187, 2018.

[17] T. Wang, Y. Zhao, J. Wang, A. K. Somani, and C. Sun, "Attention-based road registration for gps-denied uas navigation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 4, pp. 1788–1800, 2021.

[18] J. Choi and H. Myung, "Brm localization: Uav localization in gnss-denied environments based on matching of numerical map and uav images," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4537–4544.

[19] M. Mantelli, D. Pittol, R. Neuland, A. Ribacki, R. Maffei, V. Jorge, E. Prestes, and M. Kolberg, "A novel measurement model based on abbrief for global localization of a uav over satellite images," *Robotics and Autonomous Systems*, vol. 112, pp. 304–319, 2019.

[20] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792.

[21] N. Samano, M. Zhou, and A. Calway, "Global aerial localisation using image and map embeddings," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5788–5794.

[22] A. Couturier and M. A. Akhloufi, "Convolutional neural networks and particle filter for UAV localization," in *Unmanned Systems Technology XXIII*, H. G. Nguyen, P. L. Muench, and B. K. Skibba, Eds., vol. 11758, International Society for Optics and Photonics. SPIE, 2021, pp. 108 – 120. [Online]. Available: https://doi.org/10.1117/12.2585986

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[24] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[25] A. W. Long, K. C. Wolfe, M. J. Mashner, G. S. Chirikjian *et al.*, "The banana distribution is gaussian: A localization study with exponential coordinates," *Robotics: Science and Systems VIII*, vol. 265, 2013.

[26] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'96. AAAI Press, 1996, p. 896–901.

[27] N. Bergman, "Recursive bayesian estimation: Navigation and tracking applications," Ph.D. dissertation, Linköping University, 1999.

[28] H. Hou, Q. Xu, C. Lan, W. Lu, Y. Zhang, Z. Cui, and J. Qin, "Uav pose estimation in gnss-denied environment assisted by satellite imagery deep learning features," *IEEE Access*, vol. 9, pp. 6358–6367, 2021.

[29] J. Kinnari, F. Verdoja, and V. Kyrki, "Season-invariant gnss-denied visual localization for uavs," *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[30] ——, "Gnss-denied geolocalization of uavs by visual matching of onboard camera images with orthophotos," in *2021 20th International Conference on Advanced Robotics (ICAR)*, 2021, pp. 555–562.

[31] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[32] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel corporation*, vol. 5, no. 1-10, p. 4, 2001.

[33] R. I. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146 – 157, 1997.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[35] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017, pp. 3856–3866.

[36] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *International Conference on Learning Representations*, 2018.

[37] T. Hahn, M. Pyeon, and G. Kim, "Self-routing capsule networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, pp. 7658–7667.

[38] J. Gu, "Interpretable graph capsule networks for object recognition," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, 2021, pp. 1469–1477.

[39] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[41] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[42] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[43] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[44] *XSens MTi-3 Attitude, Heading and Reference System*, XSens, 2022. [Online]. Available: https://www.xsens.com/mti-3

[45] *Vectornav VN-100 IMU/AHRS*, Vectornav, 2022. [Online]. Available: https://www.vectornav.com/resources/datasheets/vn-100-imu-ahrs

[46] Open Source Robotics Foundation, "Robotic operating system." [Online]. Available: https://www.ros.org

[47] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.