Depth distillation: unsupervised metric depth estimation for UAVs by finding consensus between kinematics, optical flow and deep learning

Mihai Pirvu[1,2] Victor Robu[1,2] Vlad Licaret[1] Dragos Costea[1,2] Alina Marcu[1,2]
Emil Slusanschi[1] Rahul Sukthankar[3] Marius Leordeanu[1,2]
[1]University Politehnica of Bucharest [2]IMAR [3]Google Research
{mihai_cristian.pirvu,vlad.licaret,dragos.costea,alina.marcu}@upb.ro
victor_eduard.robu@stud.acs.upb.ro
{marius.leordeanu,emil.slusanschi}@cs.pub.ro sukthankar@google.com

Abstract

Estimating precise metric depth is an essential task for UAV navigation, which is very difficult to learn unsupervised without access to odometry. At the same time, depth recovery from kinematics and optical flow is mathematically precise, but less numerically stable and robust, especially in the focus of expansion areas. We propose a model that combines the analytical, vision-with-odometry approach, with deep unsupervised learning, into a single formulation for metric depth estimation, which is both fast and accurate. The two pathways – analytical and data-driven – form a robust ensemble, which provides supervisory signal to a single deep net that distills the consensus between scene geometry, pose, kinematics, camera intrinsics and the input RGB. The distilled net has low runtime and memory costs, being suitable for embedded devices. We validate our results against an off-the-shelf SfM-based solution. We also introduce a new real-world dataset of almost 20 minutes of continuous UAV flight, on which we demonstrate better accuracy and capabilities than the deep learning and analytical approaches.

1. Introduction

Retrieving metric depth information is especially important in autonomous robotics for their safe navigation in complex environments. Monocular depth estimation is useful for many robotics tasks, including SLAM, obstacle avoidance, object detection, tracking and semantic segmentation. In particular, in the context of UAVs, estimating depth is vital for ensuring a safe flight. In this paper we consider the scenario of a flight within a given perimeter, with focus on unsupervised learning to estimate depth within the larger context of aerial scene understanding.

During training, our approach takes full advantage of the complementary properties of two depth estimation pathways as shown in Fig. 1: the analytical path using odometry and the optical flow, data-driven, deep learning path, respectively. While the analytical path is mathematically perfect, it lacks robustness in the presence of noise and numerically fails in focus of expansion areas. On the other hand, the unsupervised data-driven approach is robust and smooth over time, but it is not as accurate and metric. By forming an ensemble from the two pathways and distilling them into a single net, we manage to improve both accuracy and speed, which is perfectly suitable for real-time computation on UAVs. We introduce the following **main contributions**:

1. A novel approach to metric depth estimation for UAVs, from monocular RGB, capable of learning unsupervised from a single continuous UAV flight. We distill depth into a single net, by using as "teacher" the ensemble formed by an analytical approach (based on odometry and flow) and an unsupervised data-driven, deep learning method. We show experimentally that the final net has superior performance to its teacher.

2. An improved and robust analytical depth estimation from camera velocities and optical flow, based on the simultaneous estimation of the angular velocity error and depth.

3. A novel dataset of approximately 20 minutes of UAV flight, with kinematics and GPS information, covering two European mountain town resorts.

Related work: Combining vision and sensors to obtain a better scene representation is well studied [3]. However, most methods are either challenging to deploy on UAVs (e.g., LIDAR), provide sparse information (keypoint-based), are of low resolution [9] or are computationally in-

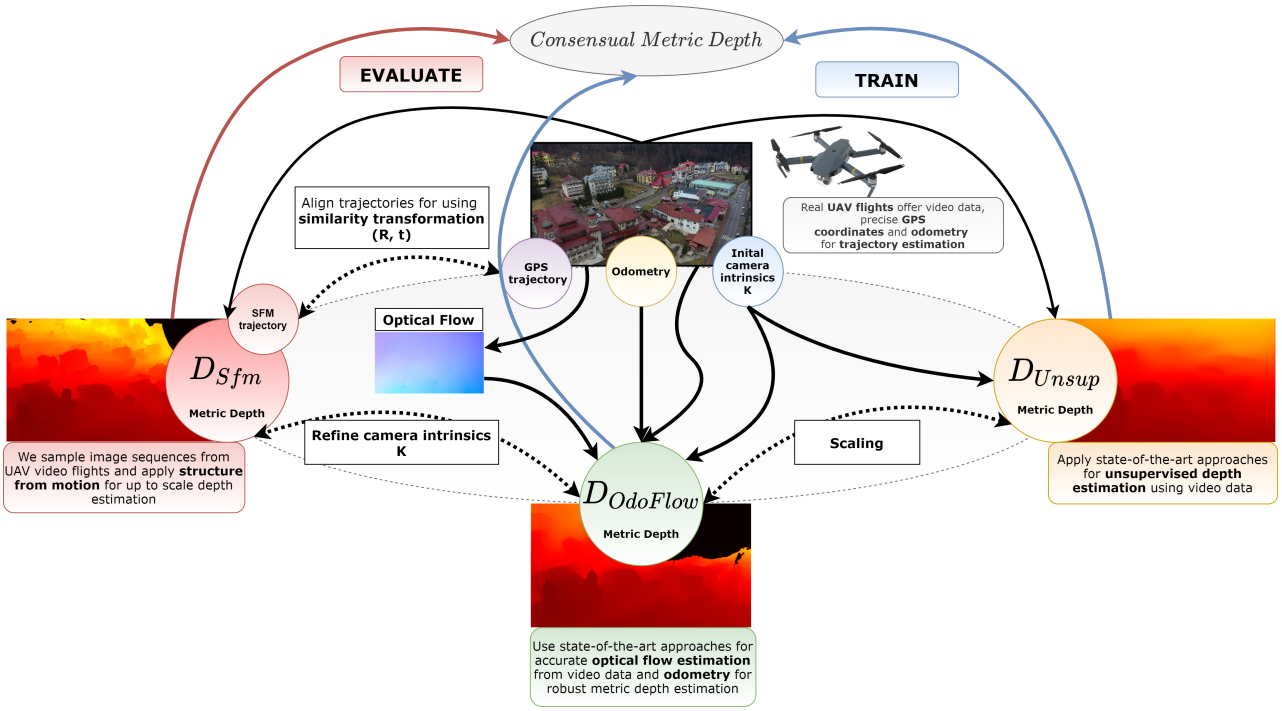

Figure 1. Overview of our approach, combining several complementary pathways for accurate metric depth estimation. Along one path, we estimate consistent non-metric depth in an unsupervised way (D_{Unsup}). Along a different path we use odometry and optical flow to estimate exact, metric depth ($D_{OdoFlow}$).$D_{OdoFlow}$ is used to scale D_{Unsup} and make it metric, then the two form an ensemble teacher, used to distill a student net for metric depth estimation (Fig. 3). Along a third path, depth is reconstructed with structure from motion software (D_{SfM}), which is made metric by aligning its predicted trajectory with the metric trajectory from GPS. D_{SfM}, computed offline, plays the role of ground truth and is used for evaluation only. By finding consensus between different data-driven and analytical paths, we exploit the complementary benefits of both approaches in order to efficiently learn metric depth estimation from single images.

tractable for on-board use. Our approach combining vision and sensors is very fast at test time (one feed-forward pass through a small net) and during training, i.e. additionally requiring only linear and angular camera velocities and optical flow to output dense depth. As ground truth depth is difficult to obtain, many recent approaches [11], [12] or [4] focus on self-supervised learning from monocular video. However, they do not generalize very well to unseen data and their estimations are arbitrarily scaled with respect to the real world. Our approach handles these issues by combining the monocular unsupervised approach with an analytical solution that offers a precise and metric depth from video, once odometry data is given.

Geometric constraints are popular for depth learning [11]. Spatial and temporal consistency is also used [12], as well as constraints of constant speed [26] or constant camera height above ground [21]. Structure from motion systems are the most popular for providing accurate depth maps, but they are computationally very heavy and often slow to converge, which makes them impractical for real time usage or embedded systems [23]. Moreover, they usually do not provide metric depth. A similar work resulting in distilled depth is [16] where the authors fine-tune a depth CNN to output consistent results, by selecting image pairs, using optical flow and back-propagating the depth error. Nevertheless, the method does not output metric depth and it must be fine-tuned at test-time for each new video, which requires SfM poses for all images (that is, the most computationally expensive part of SfM needs to executed beforehand), thus limiting its applications.

Some works [27] use consensus for estimating depth, but using stereo, not monocular video as in our case. Others [1] also use distillation in combination with sparse, feature-based matches, as supervisory signal for improving a stereo matching network. We are interested in developing a computationally tractable algorithm, which remains accurate on novel scenes that are sufficiently different from the training ones. The most successful approaches use a variation of SLAM, keypoint-based (ORB-SLAM3) [6] or planar-based (TT-SLAM) [25]. Unfortunately, there are far fewer options for arbitrary dense outputs, as required for robust UAV navigation. For example, DeepFactors [9] is limited to 256×192 px (on a desktop GPU). Also, SLAM-like methods do not output metric depth without external cues. Different from SLAM, we do not provide a 3D model of the whole scene from multiple views, but focus on the less expensive task of

instantaneous depth prediction from a single image.

Datasets for UAV research. Most datasets for depth estimation involve objects close to the camera (indoors or driving). Although UAVs, with six degrees of freedom, make video acquisition easily accessible, there is a strong need for high quality videos, with proper annotations (segmentation) and additional information (sensors) in the research community, to train accurate models on various tasks for on-board deployment. Some efforts were made in this direction and we will also make all our data fully accessible. Most public datasets target specific tasks, such as object-detection [18], traffic surveillance [5] or semantic segmentation [20]. While some offer video data for both real [20, 17] and synthetic environments [15, 10], most lack the information needed for proper scene understanding (e.g. GPS, velocity and camera angles). Our dataset has multiple advantages: real-world video data of two well-surveyed complex scenes, at different altitudes, with full data analytics provided, suited for several tasks, such as estimating depth and trajectories.

2. Metric depth distillation

Trajectory estimation from GPS: In order to recover metric depth from a single camera, additional sensors are required. We assumes having a sensing system (available on commercial drones), which gives GPS positions, camera orientation, linear and angular velocities. In practice, measurements are not perfectly synchronized with the RGB frames since their frequency is considerably lower. We propose fitting 3rd-degree time polynomials on fixed-size windows of consecutive GPS samples.

Analytical depth from odometry and flow: The metric depth, optical flow and the spatial velocities of the camera, which consist of the linear and angular velocities, are mathematically related and this can be completely described using the *Image Jacobian* [8]. Provided that instant velocities are available from the odometry system and dense optical flow can be robustly computed from consecutive frames, a dense metric depth map can thus be solved. Please refer to section 3 for detailed derivation, where we also propose an improved, robust estimation method.

Unsupervised metric depth learning: Unsupervised deep learning approaches for depth estimation, provide dense maps arbitrarily scaled w.r.t the real world. However, by comparing the un-scaled depth maps with the odometry-based metric depth, a suitable scaling ratio r can be determined. We use a two-step approach. We first employ median scaling, commonly used in unsupervised evaluations, for valid pixels that lie within a certain depth range of interest ($r = median(D_{OdoFlow} * Valid)/median(D_{Unsup} * Valid)$, with $Valid$ representing a $True, False$ pixel map). Second, we refine r by iterative recalculation the median ratio for pixels from the two depth maps that lie into a certain maximum distance from one another ($Valid = l1(D_{OdoFlow}, D_{Unsup}) < d_{max}$). The 2nd step helps reduce the influence of errors (pixels that assign wildly different Z values for the same point in space) on the final scaling ratio.

Distillation of multiple depths: Knowledge distillation [13] is essential in machine learning. It consists of a teacher-student scheme, to distill the knowledge of the "teacher" into a more compact "student". The teacher (T) can be any process that yields predictions of type $f_T : A \rightarrow B$. The student (S) has to mimic the teacher, instead of learning the actual process f_T, by minimizing: $\min L(f_S(A), f_T(A))$. To perform unsupervised distillation and evaluation, we define 3 depth maps per frame:

- D_{SfM} depth estimated from the 3D model of the scene computed with structure from motion software, using the flight video, after automatic alignment of the SfM trajectory (with a similarity transformation) to the real GPS trajectory. D_{SfM} is used for evaluation only.
- $D_{OdoFlow}$ instantaneous depth computed analytically (Sec. 3), from the optical flow between consecutive frames and sensors information, by carefully matching 3 coordinate spaces: camera, UAV and GPS logs. Once they are synchronized, dense metric depth is computed via instantaneously angular and translational speeds. Note that we consider the camera intrinsic parameters $\mathbf{K}$ known, after prior calibration.
- D_{Unsup} by using a deep neural network specifically fine-tuned, unsupervised, on the training flight video, to maximize performance on this particular scene. D_{Unsup} becomes metric by scaling it according to $D_{OdoFlow}$ (find a single scale parameter per frame, by matching the depths per pixel). Both $D_{OdoFlow}$ and D_{Unsup} form the unsupervised teacher.

During training, the student learns to mimic the "teacher", which is the ensemble formed by averaging the analytical depth $D_{OdoFlow}$ with the data-driven, deep learning depth D_{Unsup}. The student becomes a compact representation of both pathways (Fig. 1): $f_S : RGB \rightarrow Depth$, and it is used for real-time metric depth prediction from single RGB frames, with no sensor information required.

SfM alignment: For evaluation, we use an off-the-shelf structure from motion (SfM) solution, able to reconstruct the 3D surface of the surveyed area (Sec. 4.1). This is a computationally demanding, offline stage, yielding a single globally optimal result, very useful for evaluation, but impractical for real-time on-board use. Since SfM provides a

3D model that is arbitrarily scaled, w.r.t to an arbitrary coordinate system, we bring it to the correct scale in the world coordinates by automatic matching of the GPS and SfM trajectories using scale invariant graph matching, once the matches are found, a similarity transformation between the world and the SfM 3D models estimated by least squares. Then, by interpolating the camera poses along the SfM trajectory we can obtain virtual depth maps that, while incomplete (as the surface is not necessarily continuous) map surprisingly well the real RGB frames and the 3D SfM model.

3. Mathematical Formulation

Analytical trajectory estimation: The GPS coordinates are first transformed into 2D Cartesian coordinates (x_i, y_i), where i indexes odometry measurements. Polynomials are fit separately on the two coordinates on a fixed-size sliding window of samples. In the case of the first axis and third degree polynomials, the coefficients a_0, a_1, a_2, a_3 are the least squares solution to the system formed by the equations $x_k = a_0 + a_1 t_k + a_2 t_k^2 + a_3 t_k^3$, where k indexes the measurements x_k in the window and t_k is their associated timestamp. These can be expressed relative to timestamp of the middle sample for improved numerical stability. Given the timestamp of a camera frame, the position of the device at that time is computed using the polynomial with the closest middle sample timestamp. In the considered UAV flights the motion is mostly planar, so only GPS is taken into consideration. However, height information can be incorporated for more complex motion.

While the spatial velocities of the device could be solved in closed-form using the analytical trajectory, the orientation of the UAV and that of the camera do not coincide most of the time. Thus, it is preferred to use information about camera pose from the odometry system instead. In this case, the provided linear velocity and camera orientation are interpolated linearly and with *slerp*, respectively. The interpolated orientation provided by odometry R_j, where j indexes frames, is used to estimate angular velocities ω_j as:

$$\omega_j = \frac{(log(R^j_{j+1}))^\vee - (log(R^j_{j-1}))^\vee}{2\Delta t} \tag{1}$$

where $R^j_{j+1} = R_j^T R_{j+1}$ is the relative rotation between frames j and $j+1$, Δt is the time difference between consecutive frames, log is the matrix logarithm and $()^\vee$ maps skew-symmetric matrices to the corresponding 3-dimensional vectors.

Depth from odometry, flow and trajectory: the optical flow of a pixel is proportional to the movement of the camera, but inversely proportional to the relative distance between the camera and the corresponding world point. More precisely, a point $P = (X, Y, Z)$ expressed in the camera frame is projected by the pinhole model in the point $(x, y) = (X/Z, Y/Z)$ with the temporal derivatives:

$$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2}, \quad \dot{y} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} \tag{2}$$

Considering the linear velocity ν and angular velocity ω of the camera, the temporal derivative of P is:

$$\dot{P} = -\omega \times P - \nu \tag{3}$$

which can be written in scalar form as:

$$\begin{aligned} \dot{X} &= \omega_z Y - \omega_y Z - \nu_x \\ \dot{Y} &= \omega_x Z - \omega_z X - \nu_y \\ \dot{Z} &= \omega_y X - \omega_x Y - \nu_z \end{aligned} \tag{4}$$

The normalized image-plane coordinates (x, y) are related to the pixel coordinates (u, v) and the centered coordinates $(\bar{u}, \bar{v})$ by the intrinsic camera parameters (u_0, v_0, f):

$$fx = u - u_0 = \bar{u}, \quad fy = v - v_0 = \bar{v} \tag{5}$$

By substituting Eq. 4 and 5, along with their derivatives, into Eq. 2 and rearranging the terms it follows that:

$$\begin{pmatrix} \dot{\bar{u}} \\ \dot{\bar{v}} \end{pmatrix} = \begin{pmatrix} -\frac{f}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{f} & -\frac{f^2+\bar{u}^2}{f} & \bar{v} \\ 0 & -\frac{f}{Z} & \frac{\bar{v}}{Z} & \frac{f^2+\bar{v}^2}{f} & -\frac{\bar{u}\bar{v}}{f} & -\bar{u} \end{pmatrix} \begin{pmatrix} \nu_x \\ \nu_y \\ \nu_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{6}$$

The matrix in 6 is called the *Image Jacobian* matrix. The depth can be solved by rewriting 6 as:

$$(J_\nu \nu)\frac{1}{Z} = \dot{\bar{p}} - J_\omega \omega \tag{7}$$

where J_ν and J_ω are the two halves of the Jacobian, after factoring out the $\frac{1}{Z}$ term. This is a linear equation in the scalar $\frac{1}{Z}$ with $A = J_\nu \nu$ and $b = \dot{\bar{p}} - J_\omega \omega$, which are 2-dimensional vectors. The b vector represents the optical flow of the point caused only by translational motion, after removing the rotational component $J_\omega \omega$. Similarly, vector A is the flow explained by the linear velocity of the camera and should ideally be equal to b, up to the scaling factor $\frac{1}{Z}$. Thus, the least squares solution is simply:

$$Z = \frac{\|A\|^2}{A^T b} \tag{8}$$

In order to account for the errors in the numerically estimated angular velocity, we augmented equation 7 to account for a correction $\Delta\omega$:

$$J_\nu \nu \frac{1}{Z} + J_\omega \Delta\omega = \dot{p} - J_\omega \omega \tag{9}$$

which is a linear system in both Z and $\Delta\omega$. As there are 4 unknown variables and only 2 equations, the equations for multiple p_i points, $1 \leq i \leq N$, $N > 2$, are stacked in order to solve for their respective depths Z_i and $\Delta\omega$:

$$\begin{pmatrix} J_\nu^1\nu & 0 & \dots & 0 & J_\omega^1 \\ 0 & J_\nu^2\nu & \dots & 0 & J_\omega^2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & J_\nu^N\nu & J_\omega^2 \end{pmatrix} \begin{pmatrix} \frac{1}{Z_1} \\ \frac{1}{Z_2} \\ \dots \\ \frac{1}{Z_N} \\ \Delta\omega \end{pmatrix} = \begin{pmatrix} \dot{p}_1 - J_\omega^1\omega \\ \dot{p}_2 - J_\omega^2\omega \\ \dots \\ \dot{p}_N - J_\omega^N\omega \end{pmatrix} \tag{10}$$

A subset of the pixels in a frame is first used to solve for the angular velocity correction with Eq. 3 and then the depth of each pixel is solved individually using the corrected angular velocity by 8. Our robust depth computation which takes into account errors in angular velocity measurements is another theoretical contribution of our work.

Post-processing for depth computation: the dot product $A^T b$ in the denominator of Eq. 8 causes numerical issues when the norm of either vector is close to 0 or they are close to perpendicular. The norm of the optical flow $\|b\|_2$ reaches very small values around the focus of expansion. Moreover, the angle between vectors A and b should be 0 for them to be equal up to the scaling factor $1/Z$. Thus, depths are considered invalid if $\|b\|_2$ is below a threshold ϵ (20 px/s), the angle between A and b is above a threshold δ (20°), or the depth is below a threshold γ (20 m) and the image point is close to the point where $\|A\|_2$ reaches its minimum.

3.1. Depth distillation algorithm

Suppose we are presented with N independent predictions of the same source and target domain $f_i : A \rightarrow B, i \in \{1, ..., N\}$. We aim to create a direct morphing between them via agreement so that, in the end, we are left with one unique prediction: $f(A) = ensemble(f_i(A))$. In our case, we have 1 RGB input and 3 independent depth maps: D_{Unsup}, ${}_DOdoFlow$ and D_{SfM}. We will keep D_{SfM} as "ground truth", for evaluation, being one that is computed by a sophisticated global optimization process, which is heavy computationally and can be done only offline. Therefore, it is expected that D_{SfM} is the more accurate of the three, even if the 3D SfM model is neither complete, nor perfect (i.e. it contains holes, the surface is often simplified). In contrast, both D_{Unsup} and $D_{OdoFlow}$ are instantaneous and can be computed fast, thus making them suitable for online training and prediction. We employ a simple ensemble function to combine D_{Unsup} and ${}_DOdoFlow$, which computes the element-wise average of the two depth maps.

We are also careful with invalid predictions: $D_{OdoFlow}$ outputs invalid values in areas of small or no motion (e.g. focus of expansion areas) and D_{SfM} outputs invalid values in the areas not properly covered by the 3D model. When applying the *ensemble* function, in the $D_{OdoFlow}$'s case, we fallback to D_{Unsup} when evaluating the result. Since we use D_{SfM} as "ground truth", we choose to mask its invalid regions from the evaluation. As presented in Figures 1 and 3, we will train a student net to predict, from a single RGB image, the metric depth output of the teacher ensemble combining $D_{OdoFlow}$ and D_{Unsup}. As mentioned, we perform all evaluations on D_{SfM}, considered as ground truth.

Algorithm 1: Neural network depth distillation

for $RGB, D_{Unsup}, D_{OdoFlow} \in \mathbb{D}_{train}$ **do**
  $D_{prediction} \leftarrow f(RGB)$;
  $D_{ensemble} \leftarrow ensemble(D_{Unsup}, D_{OdoFlow})$;
  $L \leftarrow L_{train}(D_{prediction}, D_{ensemble})$;
  $Optimize(f, L)$
end
for $RGB, D_{SfM} \in \mathbb{D}_{validation}$ **do**
  $D_{prediction} \leftarrow f(RGB)$;
  $E \leftarrow L_{SfM}(D_{prediction}, D_{SfM})$
end
return f, E

f - neural network; *ensemble* - ensemble-merging function; *optimize* - SGD-based optimization

The training "teacher" ensemble is the average between $D_{OdoFlow}$ and D_{Unsup}, where $D_{OdoFlow}$ is valid, while it considers only D_{Unsup}, where $D_{OdoFlow}$ is invalid. D_{Unsup} is scaled according to $D_{OdoFlow}$ by estimating a single scaling parameter per frame. The ensemble produces a dense depth map per frame, with no holes or invalid depth pixels. Error is computed as pixel-wise average L1 distance between the output and SfM depth. This can be written as: $L = \frac{1}{|\mathbb{N}_{test}|} \sum |D_{pred} - D_{SfM}| * Mask_{SfM}$, where $(D_{pred}, D_{SfM}) \in \mathbb{D}_{test}$, $Mask_{SfM}$ is a binary mask that discards invalid SfM positions.

4. Experimental Analysis

Our main goal is to learn unsupervised metric depth estimation from single RGB images, by using as teacher the consensual output of two pathways, one that uses an analytical solution from odometry, trajectory and flow and another which is a purely unsupervised vision-based approach. For this task we introduce a novel dataset of two continuous UAV flights in two mountain town resorts, termed **Slanic** and **Herculane**. We divide **Slanic** in two non-overlapping subsets, one used for *training* the depth-distillation neural network and the other for *testing* purposes, measuring how well a UAV would perform in the same scene on a new flight. **Herculane** is used for testing only, to estimate the generalization capabilities of the proposed solutions, in different novel scenes, not seen during training.

Figure 2. Trajectories of our UAV flights. We collected data from two different mountain towns in order to evaluate the generalization capabilities of our system (the 2nd scene is used for evaluation only). Images are captured from manual flights - as opposed to the grid-based pattern traditionally used for SfM reconstruction.

Dataset description: Slanic and Herculane sequences include GPS information, linear and angular velocities, as well as camera absolute angles. The total length is approximately 20 minutes, the videos feature 3840x2160 30 FPS images, while the odometry is provided at 10 Hz.
Slanic - The set consists of 14777 frames - 9022 training and 5755 testing frames. It features images from a mountain area, with fairly low altitude buildings surrounded by forest. The relative altitude is fairly constant (≈50m).
Herculane - The set consists of 18044 frames, out of which 9022 were used for testing. It features a resort, with taller buildings (mostly hotels) and higher altitude (≈60-70m).

4.1. Experimental setup

For the $D_{Unsup} \rightarrow D_{OdoFlow}$ scaling procedure (which makes D_{Unsup} metric) we use a range of depth of $[50, 150]$ $meters$ and further refine r by masking all pixels that are more than 5 $meters$ apart, for a maximum of 10 $iterations$. By the last iteration we observe that $30 - 40\%$ of pixels are considered valid. We derive this ratio for each frame independently. We also limit our depth maps by clipping values to the range of $[0, 400]$ $meters$. All depth estimation procedures require knowledge of the intrinsic camera parameters, which we derive from specifications provided by the manufacturer.

For the SfM reconstruction, we settled for Meshroom [2] – a fully automated pipeline – mostly for its speed compared to wider used software, such as COLMAP [23]. We sampled the videos at 7FPS, resulting in approximately 1300 images, depending on the used dateset. Additional frames (up to 3100) quadrupled the computation time, while providing very small gains in mesh quality. We originally used GPS information, but discovered the algorithm failed to match a number of frames and still resulted in offsets. Since we already had metric depth, we decided, as mentioned previously, to align a higher quality mesh by matching GPS and SfM reconstructed trajectories. Z-Depth information was then captured by importing the mesh into Blender [7], interpolating the SfM trajectory, and rendering depth maps in locations corresponding to the real-world trajectory. For unsupervised depth estimation, we used [4], but our approach is agnostic of what approach we used for this step. For optical flow, we achieved the best results with RAFT [24]. Nevertheless, we also explored faster alternatives such as a light version of RIFE [14] which yielded good results at over 30FPS on an embedded platform.

For the training process, we divide the training dataset, Slanic-Train, in two parts, one for optimization and one for validation and model selection, using a standard 80/20 split. The second dataset, Herculane, is kept as is, and only used for testing. All the neural networks are deep convolutional architectures, based on [19], that have been successfully deployed on embedded systems and also offer real-time inference for supervised depth estimation. We employ two variants: SafeUAVTiny and SafeUAVLarge and we analyze the trade-offs between speed and accuracy below.

We use PyTorch [22] as neural network framework. For each dataset, we work with 4 synchronized items: *RGB* and the 3 independent metric-based depths, obtained as described above: D_{Unsup} (Unsupervised neural network), $D_{OdoFlow}$ (Flow and Odometry based) and D_{SfM} (Structure from motion based). Our goal is to obtain a neural network that predicts $f : RGB \longrightarrow Depth$ by creating a morphing consensus between the three independent depths. For this purpose, we use two ensemble algorithms that tries to optimize between D_{Unsup} and $D_{OdoFlow}$, while using D_{SfM} for model selection only. We use a simple L2 loss function between the predicted depth from RGB and the two depth maps we optimize against. Furthermore, one issue with these maps is that they are sparse, containing invalid entries (NaNs) in some situations, such as sky or no motion field zones. For this purpose, we mask these invalid regions, thus optimizing only on reliable depth data. The ensemble algorithms we use are: *alternate* and *simple mean*. The first one alternatively uses $D_{OdoFlow}$ and D_{Unsup} as pseudo ground truth during the training process. The second one combines the two depth maps by doing a pixel-wise average, while being careful about invalid positions in $D_{OdoFlow}$ falling back to D_{Unsup} which provides dense predictions, where needed.

4.2. Results

In Tab. 1 we present the percentage of valid data for each dataset. In Tab 2 and 3 we present the results of the three methods used to extract the metric depth on which the neural networks are trained. Two metrics are employed: namely *L1 metric* and *Relative L1 metric*. The first one computes the average pixel difference between the prediction and the GT (SfM). The second one weighs down errors for far away distances by computing an error relative to distance: $Relative(i) = \frac{Pred(i) - GT(i)}{GT(i)}$. We evaluate the distilled neural nets on the Slanic dataset, proving that the distilled students could improve over their teacher.

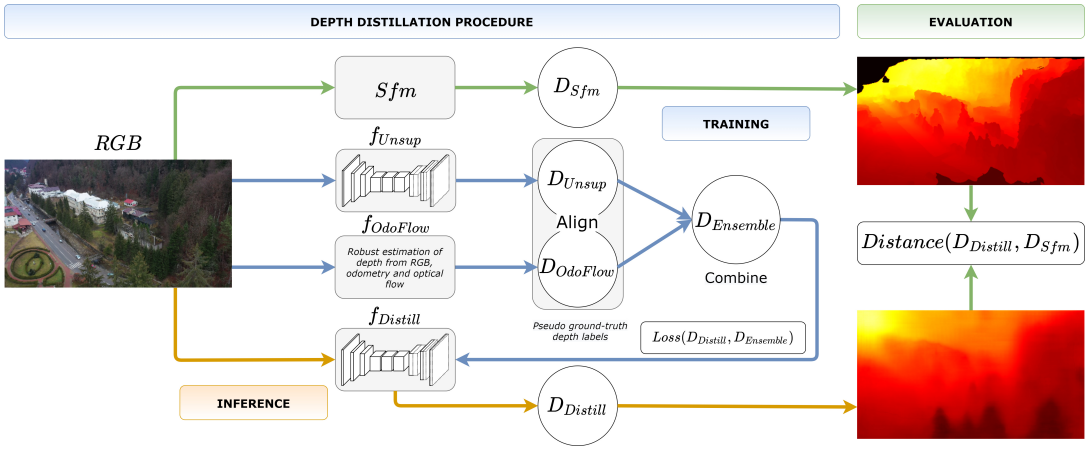

Figure 3. Our depth distillation procedure for accurate metric depth estimation. We combine two label-free methods (unsupervised and analytical) into a single result and evaluate it against the SfM reconstruction. The distilled student is able to significantly improve over its teacher on the test video from the first scene, while remaining competitive on the second scene, unseen during training.

	D_{Sfm}	$D_{OdoFlow}$
Slanic	2.28%	19.70%
Herculane	3.45%	30.61%

Table 1. Percentage of invalid pixels for the analytical method $D_{OdoFlow}$ (sky, remote and focus of expansion areas) and D_{SfM} (usually sky, meshing errors could result in holes inside the mesh).

	Slanic		Herculane	
	Metric	Relative	Metric	Relative
D_{Unsup}	27.28 m	17.10 %	44.39 m	20.29 %
$D_{OdoFlow}$	26.05 m	16.34 %	**39.67 m**	**17.53 %**
$D_{Ensemble}$	25.63 m	15.88 %	41.18 m	18.29 %
$Tiny-16$	**21.58 m**	**14.58 %**	46.77 m	24.09 %
$Large-16$	21.84 m	14.65%	48.00 m	23.97 %

Table 2. Mean absolute and relative errors on entire valid map against D_{SfM} ground truth depth. For $D_{OdoFlow}$ and $D_{Ensemble}$, we use D_{Unsup} prediction in the invalid regions.The distilled students have best results on Slanic test data.

Baseline results: We test on the full area (Tab. 2), where $D_{OdoFlow}$'s invalid pixels are replaced with D_{Unsup} and on the "good" area (Tab 3), where both $D_{OdoFlow}$ and D_{Unsup} are valid. We notice that $D_{OdoFlow}$ is generally superior to D_{Unsup} in the valid areas, whereas the ensemble is better than both, as expected. Also, errors of both $D_{OdoFlow}$ and D_{Unsup} in the "good" areas are much smaller than the average overall error of D_{Unsup}.

Distilled student network: After training on the mean ensemble teacher combining D_{Unsup} with $D_{OdoFlow}$, testing was done on both datasets (Tab. 2 and 3) with two student nets, one small and another large in terms of number of parameters. Interestingly enough both distilled student nets are superior on Slanic to their teachers and they are also able to generalize to the new scene in Herculane.

	Slanic		Herculane	
	Metric	Relative	Metric	Relative
D_{Unsup}	21.06 m	15.31 %	31.61 m	16.60 %
$D_{OdoFlow}$	19.56 m	14.39 %	**24.97 m**	**12.72 %**
$D_{Ensemble}$	19.03 m	13.81 %	27.10 m	13.79 %
$Tiny-16$	**16.11 m**	**12.90 %**	37.42 m	22.95 %
$Large-16$	16.66 m	13.41 %	37.43 m	22.41 %

Table 3. Absolute and relative errors on the good area, which is defined by masking both invalid D_{SfM} and $D_{OdoFlow}$ predictions. We observe that these areas yield much better and stable overall results. Herculane has higher errors mainly due to the higher height of the dataset. The distilled students have best results on Slanic test data.

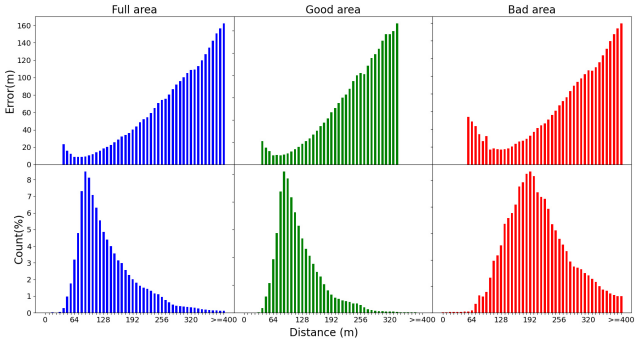

Figure 4. Errors distributions as functions of distance on test set of Slanic (of the distilled net). The overall (blue), good (green) and bad (red) errors are shown. We also plot pixel distribution per distance. Note how the absolute error increases with distance.

Analysis of the good and bad areas: We observe that the good area (where $D_{OdoFlow}$ is valid) is correlated with shorter distances, while the bad area is mostly represented by far away regions. Consequently the "bad" area has larger errors, as expected. Humans also have much better depth

Method	# Parameters	Desktop[FPS]	Embedded[FPS]
D_{Unsup} [4]	14,842,236	166.703 ± 3.27	11.631 ± 0.55)
$OpticalFlow(only)$ [14]	15,263,888	83.404 ± 2.65	43.699 ± 3.13
$D_{OdoFlow}$	n/a	26.807 ± 6.17	10.127 ± 0.73
$Tiny-16$ [19]	1,119,862	54.922 ± 1.33	10.357 ± 0.22
$Large-16$ [19]	2,005,239	51.969 ± 1.32	9.045 ± 0.13

Table 4. Frames per second on desktop and embedded GPUs. The depth from flow algorithm runs on CPU. The desktop features a RTX 2080 GPU and Ryzen 7 3700 CPU. Note that the speed on the embedded platform (Nvidia's Jetson TX2) is near real-time.

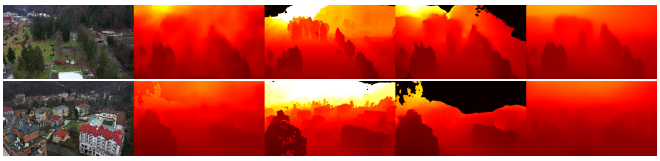

Figure 5. Qualitative results on the test set from each of our flights (Slanic - first row, Herculane - second row). From left to right, we present in order: RGB, $D_{Student}$, D_{SfM}, $D_{OdoFlow}$ and D_{Unsup} [4]. Dark regions ($D_{OdoFlow}$) and white areas (D_{SfM}) indicate invalid or missing depth. Note how the student learns to combine D_{Unsup} and $D_{OdoFlow}$, often improving over both.

estimation nearby or in regions where the motion field is informative. In practice, we are also more interested in objects that are close to the UAV, in order to avoid obstacles, rather than objects at hundreds of meters away.

Qualitative results: We observe the strengths and weaknesses of all the solutions in Fig. 5. First, our nets learn to estimate depth with high confidence, while maintaining a competitive inference time. The SfM solution is precise, revolving around a built 3D model, but its detail accuracy is questionable – since it starts from sparse points. It also takes a very long computation time for global SfM optimization and it is not feasible for real time operation. The unsupervised method has a better accuracy around object features, but it is fairly slow in training and not metric. Finally, the Flow-Odometry method is accurate, needs no training but it does require optical flow, for which fast options are available (see Table 4). Furthermore, it has a considerable amount of invalid predictions at long distances.

Computation speed: Tab. 4 presents real-time usage for the external method (D_{Unsup} [4]), compared to the 6 networks we used. For a regular consumer GPU (RTX 2080), most algorithms operate in real-time. Furthermore, the nets or algorithms are also good embedded candidates (near real-time). Our embedded platform (Jetson TX2) can be deployed on a commercial drone, enabling real-time inference in any conditions. For this purpose alone, we will only take the performance numbers on the two smallest networks into account, Tiny-16 and Large-16.

Comments on the numerical results: We still do not know our true absolute error since we compare to SfM not to the absolute ground truth, which is not available. What is really important, besides the absolute numbers, are some key aspects, such as: $\mathbf{1}$) the student can outperform the teacher and generalizes well to new scenes; $\mathbf{2}$) the errors are significantly smaller for shorter distances, which is what we need in practice; $\mathbf{3}$) the errors are also small in "good" areas, where odometry and flow matter, which could be automatically detected. $\mathbf{4}$) the student net is small, it can be deployed on an embedded GPU and it has, besides strong performance, a near real-time speed.

5. Conclusions

There is no silver bullet for fast, robust metric depth estimation, especially when no absolute ground truth is available during the training process. Each of the investigated methods (unsupervised, depth from optical flow, SfM) is prone to specific errors or compute constraints. Nevertheless, our study shows that understanding the validity regions and distilling the knowledge from geometry and deep learning approaches results in a competitive pipeline for embedded use (10FPS, 12% error). What makes our work unique is the idea to combine two complementary solutions, the analytical method based on flow and odometry and the data-driven unsupervised learning approach, to form a powerful ensemble, which becomes a strong unsupervised "teacher" for a lightweight student net, which often manages to improve over both its teachers. The extensive evaluation using the more accurate depth from SfM (which is globally optimized) from familiar as well as novel scenes, proves that our self-supervised training on the teacher ensemble is effective in practice. Future work aims to improve the learning from consensus algorithm by incorporating additional representations and constraints. We are also optimizing the pipeline to achieve real-time inference on embedded devices.

6. Acknowledgements

This work was funded by UEFISCDI, under Projects EEA-RO-2018-0496 and PN-III-P4-ID-PCE-2020-2819. We want to express our sincere gratitude towards Aurelian Marcu and The Center for Advanced Laser Technologies (CETAL) for providing access to GPU resources.

References

[1] Filippo Aleotti, Fabio Tosi, Li Zhang, Matteo Poggi, and Stefano Mattoccia. Reversing the cycle: self-supervised deep stereo through enhanced monocular distillation. In *European Conference on Computer Vision*, pages 614–632. Springer, 2020.

[2] AliceVision. Meshroom: A 3D reconstruction software., 2018.

[3] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1–26, 2016.

[4] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems*, pages 35–45, 2019.

[5] Ilker Bozcan and Erdal Kayacan. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8504–8510. IEEE, 2020.

[6] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial and multimap slam. *arXiv preprint arXiv:2007.11898*, 2020.

[7] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[8] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, volume 118. Springer, 2017.

[9] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5(2):721–728, 2020.

[10] Michael Fonder and Marc Van Droogenbroeck. Mid-air: A multi-modal dataset for extremely low altitude drone flights. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, June 2019.

[11] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3838, 2019.

[12] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[14] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Rife: Real-time intermediate flow estimation for video frame interpolation. *arXiv preprint arXiv:2011.06294*, 2020.

[15] Marius Leordeanu, Mihai Pirvu, Dragos Costea, Alina Marcu, Emil Slusanschi, and Rahul Sukthankar. Semi-supervised learning for multi-task scene understanding by neural graph consensus. *arXiv preprint arXiv:2010.01086*, 2020.

[16] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (TOG)*, 39(4):71–1, 2020.

[17] Ye Lyu, George Vosselman, Gui-Song Xia, Alper Yilmaz, and Michael Ying Yang. Uavid: A semantic segmentation dataset for uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 165:108 – 119, 2020.

[18] Murari Mandal, Lav Kush Kumar, and Santosh Kumar Vipparthi. Mor-uav: A benchmark dataset and baselines for moving object recognition in uav videos. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2626–2635, 2020.

[19] Alina Marcu, Dragos Costea, Vlad Licaret, Mihai Pîrvu, Emil Slusanschi, and Marius Leordeanu. Safeuav: Learning to estimate depth and safe landing areas for uavs from synthetic data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[20] Alina Marcu, Vlad Licaret, Dragos Costea, and Marius Leordeanu. Semantics through time: Semi-supervised segmentation of aerial videos with iterative label propagation. In *Asian Conference on Computer Vision (ACCV), 2020*, pages 2881–2890, 2020.

[21] Zhixiang Min, Yiding Yang, and Enrique Dunn. Voldor: Visual odometry from log-logistic dense optical flow residuals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4898–4909, 2020.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020.

[25] Xi Wang, Marc Christie, and Eric Marchand. Tt-slam: Dense monocular slam for planar environments. In *IEEE International Conference on Robotics and Automation, ICRA'21*, 2021.

[26] Hang Zhou, David Greenwood, Sarah Taylor, and Han Gong. Constant velocity constraints for self-supervised monocular depth estimation. In *European Conference on Visual Media Production*, pages 1–8, 2020.

[27] Lingtao Zhou, Jiaojiao Fang, and Guizhong Liu. Unsupervised video depth estimation based on ego-motion and disparity consensus. *arXiv preprint arXiv:1909.01028*, 2019.