DISC: A Large-scale Virtual Dataset for Simulating Disaster Scenarios

Hae-Gon Jeon[1], Sunghoon Im[2], Byeong-Uk Lee[2], Dong-Geol Choi[3], Martial Hebert[1] and In So Kweon[2]

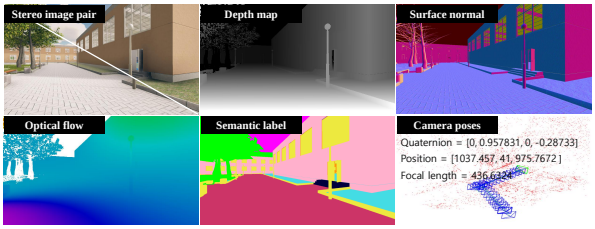
***Abstract*— In this paper, we present the first large-scale synthetic dataset for visual perception in disaster scenarios, and analyze state-of-the-art methods for multiple computer vision tasks with reference baselines. We simulated before and after disaster scenarios such as fire and building collapse for fifteen different locations in realistic virtual worlds. The dataset consists of more than 300K high-resolution stereo image pairs, all annotated with ground-truth data for semantic segmentation, depth, optical flow, surface normal estimation and camera pose estimation. To create realistic disaster scenes, we manually augmented the effects with 3D models using physical-based graphics tools. We use our dataset to train state-of-the-art methods and evaluate how well these methods can recognize the disaster situations and produce reliable results on virtual scenes as well as real-world images. The results obtained from each task are then used as inputs to the proposed visual odometry network for generating 3D maps of buildings on fire. Finally, we discuss challenges for future research.**

I. INTRODUCTION

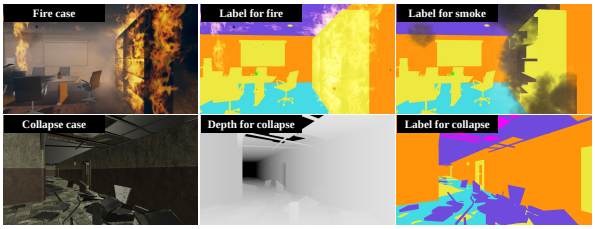
There have been many works on high-quality image acquisition and recovery in bad weather conditions such as fog, noise, and haze. Convolutional neural networks (CNNs) trained with large-scale datasets allow obtaining high-quality images from images captured under bad weather conditions. Works in [1], [2] address an image dehazing problem via multi-scale CNNs which learn effective features to infer the scene transmission of single hazy images. Deep convolutional generative adversarial networks have shown promising results in generating clean images from images corrupted by noise [3] and rain [4] etc.

However, few works for extreme conditions such as disaster environments have been studied even though vision-based autonomous vehicles and robots to survey damage after a disaster are in demand for rescue systems. One reason for the lack of works under disaster conditions is a lack of suitable datasets. It might be impossible to accurately obtain images with well-annotated ground-truth before and after the disaster conditions.

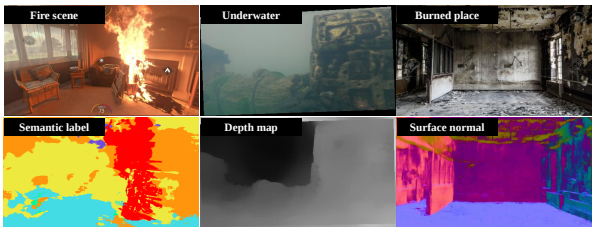
As a solution to this problem, one research direction that has gained interest involves generating synthetic images with the corresponding ground truth using 3D computer graphics [5], [6], [7], [8], [9], [10], [11], [12], [13]. The various features of graphics tools, such as texturing and shading effects, allow full control over the virtual 3D environment, thus ensuring lower costs, greater flexibility, limitless variety, and quantity. Moreover, the physics engine built into the tools supports an approximate simulation of certain physical systems, such as fire, smoke, and fluid dynamics. This synthetic dataset simulated using physical phenomena can extend the range of visual perception to include severe conditions which the existing public datasets do not cover [5], [6].



(a) Before disaster situations



(b) After disaster situations



(c) Real-world results

Fig. 1: Examples of the DISC: We provide stereo image sequences with corresponding ground-truth data including depth map, surface normal, optical flow, semantic label and camera poses for both before and after disaster scenarios.

In this paper, we present a large-scale synthetic Dataset for dIsaster SCenarios, referred as DISC, simulated using a physics engine in normal and disaster scenarios as shown in Fig.1(a) and (b), respectively. Our main contributions are summarized as below:

- We introduce a dataset with 300K images with two types of damage scenarios: collapsing and fire scenarios. The input modality is high-resolution stereo video and its well-annotated ground-truth is provided for scene understanding and visual perception tasks.
- We present a manual process for augmenting and retouching disaster effects to achieve photorealistic disaster effects, which could not be made automatically.

[1]The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA. (Primary contact: haegonj@andrew.cmu.edu)
[2]The Robotics and Computer Vision Lab., KAIST, Daejeon 34141, Republic of Korea.
[3]Department of Information and Communication Engineering, Hanbat National University, Daejeon 34158, Republic of Korea.

Fig. 2: Virtual 3D models to generate the DISC with various scene contexts, light conditions and materials. (Indoor) furniture shop, living room, office, police station, residence, warehouse, school and old castle. (Outdoor) city scape1, city scape2, suburban and park. (Underground) subway station, tunnel and underpass.

In addition, we show a technique for extracting a mask for fire and fog, and labels for the rubble of collapsed buildings in disaster scenes. (see Sec.II)

- We perform extensive experiments to evaluate the performance of state-of-the-art methods for semantic segmentation, surface normal estimation, depth estimation, optical flow estimation, and camera relocalization for before and after disaster scenarios of DISC. In particular, the state-of-the-art methods finetuned on DISC show significant performance improvements for disaster scenes as well as real-world disaster imagery in Fig.1(c). (see Sec.III)
- We propose a visual odometry network for generating 3D map of buildings on fire. For this, our network takes the experimental results as input, and predicts accurate camera poses in disaster conditions. (see Sec.IV)

II. Data Generation

In this work, we simulate and render before and after disaster scenarios of 15 virtual places including indoor, outdoor and underground scenes in Fig.2. We use public 3D models to ensure the scalability of our synthetic data. For each 3D model, we capture stereo video sequences following pre-defined camera paths in normal situations. We then composite realistic disaster effects on the 3D model and re-capture them in the same paths. Our project page is `https://sites.google.com/site/hgjeoncv/disc-project-page`.

A. Simulation of disaster effects

To generate realistic disaster scenarios, we introduce efficient ways to composite the disaster effects on normal 3D models. The disaster effects are inevitably composed manually because there are no public 3D models with realistic disaster effects. We then present an approach for rapidly producing 3D models with corresponding ground-truth data. For this task, we utilize Unity [14] which is widely used for modern 3D computer games.

Fire cases We define fire scenes as combinations of smoke and flame, and soot. The colors of flame and smoke depend on several factors, such as the materials being burned and the ambient temperature. In addition, smoke from a fire in an indoor environment significantly reduces visibility. With this observation, we create 3D models including flames, soot on object surfaces and shorten visibility distances of cameras caused by smoke. To produce these effects, however, a great deal of manual work is required. We introduce efficient ways to reduce the amount of manual work.

(a) (b) (c)

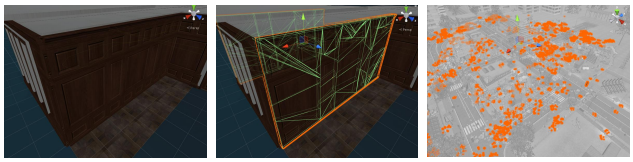
Fig. 3: An example of simulating fire scenarios. (a) Soot image samples. (b) Soot patch composition. (c) Adding light sources

Instead of making flames with the corresponding smoke for every 3D model, we generate a number of flame samples and smoke, and randomly augment them in the 3D models. Using particle effects in Unity, we design 25 different flames and synthesize them with various colors on the 3D models. The shapes of the flames are determined by the direction of the surface from which they originated, such as walls, floors, or ceilings. Note that we can generate more diverse flames if we employ a variety of colorful flame images, such as with green or blue. Smoke effects are generated by the particle tools, and their colors are randomly selected from black to white. The scale of the effects is also determined by manually tuning the starting points and end points of the flame.

Before compositing the fire effects, we synthesize the smoke images in advance with ignition spots in the 3D model in Fig.3(b). Using the spray effect in Unity, we generate diverse smoke and soot images, as shown in Fig.3(a). Their sizes depend on the scale of the fire effects. For indoor scenes, we additionally simulate thick smoke in rooms with reduced visibility of the cameras. We utilize a particle tool for fog, which involves overlaying a color onto objects based on their distance from the camera. The fog color is randomly augmented, and the fog density is manually set. As the last step for realistic fire effects, we add emissive lighting sources at each of the ignition spots, as shown in Fig.3(c). The intensity of the emitted light is also set in proportion to the flame scale.

Collapse Collapse is a phenomenon commonly observed in disasters such as earthquakes and hurricanes. In this case, it is necessary to bring down an object in the 3D model or to disintegrate debris from a wall or ceiling and then scatter it onto the floor. We use destructive models in the physics engine of Unity to produce various collapse scenarios.

We use a mesh collider in Unity with a manual fracture as the destructive method. We directly apply the mesh collider into target objects or 3D meshes such as walls and ceilings. We are able to control the strength of the collider and then generate randomly distributed fragments on a scene floor. More diverse collapse scenarios which cannot be achieved



(a) (b) (c)

Fig. 4: An example of simulating collapse scenarios. (a) Before collapse. (b) Cracking 3D model. (c) Scattering small debris.

with the mesh collider are generated by a manual fracture. For this, we select a particular 3D model in Fig.4(a) and make cracks using the fracture model in Unity in Fig.4(b). These cracked 3D objects are then manually spread over the 3D space. We perform rotation and flipping of the cracked objects to create a greater variety of screen configurations.

Lastly, we produce the effect of scattering the debris objects of the building on the floor. Small objects such as sample building debris provided by Unity are sprinkled around the collapsing 3D object in the scene (see Fig.4(c)). The arrangement, rotation, and size of each piece of the debris are manually determined to reflect the position and type of destruction of the collapsed 3D model. Because the types of building debris provided in Unity are limited, this augmentation is very useful for creating diverse scenarios without sacrificing the realistic nature of the result.

B. Acquisition of ground-truth data

A virtual environment has the benefit of easily obtaining ground truth data such as the depth, optical flow, surface normal and camera poses. Unity supports depth, optical flow and surface normal information with sub-pixel precision, but camera parameters for computer vision tasks cannot be directly recovered from this graphics tool. Moreover, although the semantic segmentation data is also directly generated by outputting a unique color on the surface of an object, accurately labeling materials such as flame and smoke using only graphics tools, including Unity is not feasible. We present two ways to solve these problems.

Camera Setting We generate dataset with a fixed vertical field-of-view (FOV) for both left and right camera as 60° (horizontal FOV varies depending on the aspect ratio a). In out setup, we set an optical axis of a right camera to be parallel to that of left camera with $0.2m$ baseline on a horizontal axis. The focal length f of the cameras can be simply computed as:

$$f_x = \frac{\hat{f} \cdot p_x}{m_x}, f_y = \frac{\hat{f} \cdot p_y}{m_y}, \text{ where } \hat{f} = \frac{m_y}{2\tan(\frac{\text{FOV}}{2})}, \quad (1)$$

where m_x, m_y are the size of the senor in the x, y axes. The stereo cameras used have the same intrinsic parameters as each other, do not suffer from optical distortions. The principal point of the camera is located in the image center.

Fluids labeling strategy Because fluids such as flames and smoke are created by particle effects, directly labeling each particle is the most intuitive idea to obtain the ground truth labels for a fluid. We paint each particle and render label

(a) Direct labeling in Unity (b) Our labeling strategy

Fig. 5: Comparison labeling fluids such as flame and smoke in Unity with in the video editing program.

Fig. 6: An example of provided labels for fire cases. Dark gray: smoke (soft label), light yellow: fire (soft label), gray: smoke, red: fire, yellow: furniture, orange: wall, purple: ceiling. The soft label represents a highly detailed transparency-preserving segmentation of flame and smoke.

images of these types of fluids as an example. However, this approach fails to obtain the correct labels as shown in Fig.5(a). This problem may be due to a resolution issue related to labeling functionality for particles in Unity or Blender. To address this, we introduce a fluid labeling technique using a video editing program (Adobe Premiere).

First, we assign color labels to objects in the 3D models, but not for the fluids. In this state, we render the 3D model into sequential images. We select each colored region and remove all of them using the background removal functionality of the video editing program. After removing the colored regions, only the pixels corresponding to the flames are left. By colorizing the remaining pixels in the video editing program, we are able to obtain an accurate label for the flames as shown in Fig.5(b). This process is equally applied to the smoke. Finally, we synthesize the labels on the scene for normal situations. In this process, we provide three types of semantic labels: object-smoke, object-fire and object-smoke-fire in Fig.6.

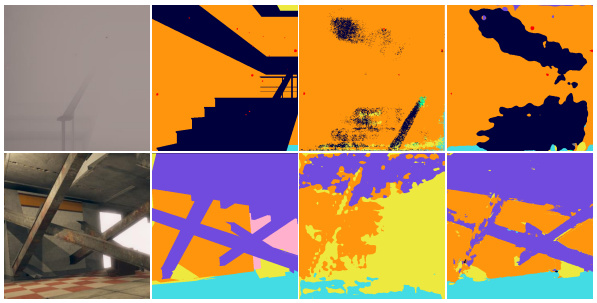
C. Dataset

We generate a total of 300,000 stereo image pairs with 1280×720 resolution as video sequences. We create a synthetic dataset suite that consists of 15 subsets and provide a complete ground-truth depth in metric scale, optical flow, semantic label, surface normal and camera poses. We render all image data using a virtual FOV of 60°. DISC provides 14 semantic labels (loads, sidewalks, cars, walls, ceilings, furniture, trees, sky, buildings, electric devices, fences, stairs, fire and smoke). For applications to indoor navigation systems, we individually label fences and stairs.

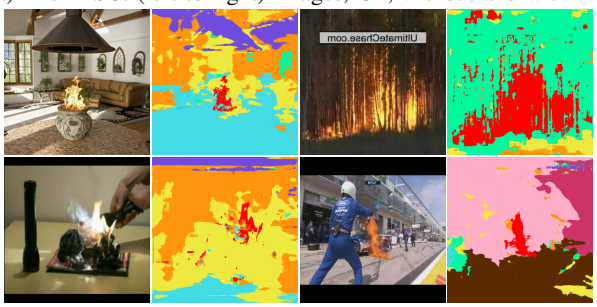
A machine equipped with an Intel i7 3.40GHz CPU, 32GB RAM and GTX 1070 Ti GPU is used for the rendering step. The rendering time is 15 seconds for one stereo pair with the corresponding ground truth on average. When rendering scenes in the presence of particle effects, the average rendering time is significantly increased. To accelerate the

Method	Normal	Fire		Collapse	
		w/o FT	w/ FT	w/o FT	w/ FT
FC-DenseNet	0.573	0.427	**0.500**	0.481	**0.540**
SegNet	0.706	0.466	**0.574**	0.517	**0.687**
DeepLab	0.549	0.420	**0.527**	0.506	**0.521**
PSPNet	0.636	0.458	**0.593**	0.478	**0.590**
DenseASPP	0.560	0.471	**0.533**	0.431	**0.499**

TABLE I: Semantic segmentation for 16 classes without and with finetuning (FT) on the DISC. (Measure: mean IoU)



(a) The DISC. (left to right) Images, GT, without and with FT.



(b) Real-world results

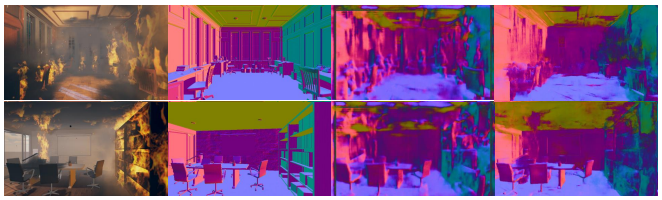
Fig. 7: Examples of semantic segmentation benchmark.

rendering speed, we deactivate the physical engine during the rendering procedure, after which we are able to achieve a time of approximately 20 seconds per one stereo pair.

III. Baselines and Analysis

In order to demonstrate the usefulness of DISC, we set up several experiments on semantic segmentation, surface normal estimation, depth estimation, optical flow and camera relocalization. In particular, we show that CNNs used for these experiments achieve remarkable results in disaster scenarios after training on DISC. The performance improvements are indicated in bold. To establish concrete benchmarks, we split the DISC dataset into a BEfore Disaster set (BE-D) and an AFter Disaster set (AF-D) such as fire and collapse scenarios.

For qualitative evaluations on real-world disaster imagery, we downloaded real-world images for the semantic segmentation and surface normal from the Internet using the Google image search tool. We captured real-world smoke scenes in the public safety training center for firefighters to acquire calibrated stereo pairs and sequential images. These images and videos were used for stereo matching and optical flow estimation, respectively.



(a) The DISC. (left to right) Image, GT, without and with FT.

(b) Real-world results. (left to right) Images, without and with FT.

Fig. 8: Surface normal estimation. Baseline: FCN-Skip

A. Semantic segmentation

One of major difficulties with visual perception tasks is the extraction of features of objects in the presence of flame and smoke, which interfere with vision, and collapses, resulting in significant changes of scene configurations. In this experiment, we trained the networks on the BE-D dataset, and compared the performance degradations on the AF-D dataset. We also demonstrate performance improvements in the disaster scenarios when the networks were finetuned on the AF-D dataset.

We evaluated five semantic segmentation networks on the DISC dataset: FC-DenseNet [15], SegNet [16], DeepLab [17], PSPNet [18] and DenseASPP [19], and we report the mean intersection over union (IoU) as a performance measure. We used approximately 20K images for training and allocated 3K images for testing and validation from eight indoor scenes. There was no temporal overlap between the training and test splits.

As shown in Table.I, we draw several conclusions. First, DISC is more challenging than [5] consisting of mainly road scenes: while PSPNet is above 0.73, it is at 0.63 on DISC. Second, we observed that performance degradations occurred in disaster scenes. In particular, the performance degradation in dense smoke regions is particularly remarkable as compared to that in other disaster scenarios. To achieve a robust solution to the smoke conditions, we trained state-of-the-art networks on AF-D. The training sets (about 5% of AF-D) and the test sets for disaster scenes were temporally separated[1]. Fig.7(a) shows that AF-D helps the networks to recognize scene conditions, and improves the performance for both disaster cases in Fig.7(a). We also show real-world results of fire scenes from SegNet, which achieved the best performance in our experiment. As shown in Fig.7(b), SegNet finetuned on AF-D works with real-world fire scenes well.

B. Surface Normal Estimation

Next, we evaluated state-of-the-art surface normal estimation using single images, specifically in relation to VGG-

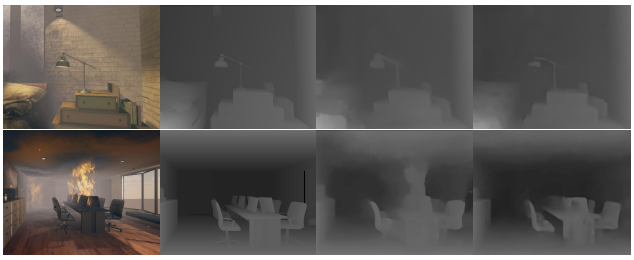
[1]The same experimental strategy in [5] is adopted in our benchmarks.

Method	Normal	Fire (° / ° / %)		Collapse (° / ° / %)	
		w/o FT	w/ FT	w/o FT	w/ FT
VGG-Multiscale	22.19 / 19.57 / 30.12	40.03 / 35.14 / 11.19	**27.61 / 22.59 / 25.45**	28.90 / 22.07 / 22.31	**25.42 / 20.36 / 27.09**
FCN-Skip	13.92 / 8.78 / 57.23	28.87 / 25.14 / 19.42	**22.24 / 16.30 / 31.37**	16.65 / 14.09 / 41.88	**14.20 / 10.18 / 45.67**
HourglassNet	19.75 / 13.02 / 39.34	38.07 / 31.92 / 12.48	**24.58 / 18.37 / 28.75**	22.71 / 16.35 / 30.02	**20.53 / 15.55 / 38.58**

TABLE II: Surface normal from single images. Mean and median of angular error (the lower the better), and percentage of pixels with error smaller than 11.25° (the higher the better).

Method	BPR5 (%)		BPR7 (%)		RMSE (pixel)	
	w/o FT	w/ FT	w/o FT	w/ FT	w/o FT	w/ FT
MC-CNN	20.02	**15.97**	12.49	**9.41**	5.37	**4.38**
DispNet	25.26	**24.26**	16.92	**16.65**	5.15	**4.60**
PSMNet	18.03	**17.94**	10.75	**8.99**	4.80	**4.17**

TABLE III: Stereo matching. Averaged bad pixel rate as the disparity error smaller than 5 pixels (BPR5) and 7 pixels (BPR7), and RMSE in smoke scenes (the lower the better).



(a) Input (b) GT (e) w/o FT (f) w/ FT

Fig. 9: Stereo matching results on DISC. Baseline: PSMNet.

(a) Input (b) [25] (c) w/o FT (d) w/ FT

Fig. 10: Stereo matching results on real-world scenes.

Multiscale [20], FCN-Skip [21] and HourglassNet [22]. Similar to Sec.III-A, we show that the performance capabilities of networks trained on BE-D degrade and that these methods yield reliable results after finetuning on AF-D in disaster scenarios.

Our benchmark in Table.II indicates that although the collapse situations change the compositions of scenes, the performance degradation of the networks is relatively minor. On the other hand, fire situations exhibit limited visibility and irregular lighting changes, which are the main causes of inaccurate surface normal estimations. Another cause of the prediction error is the soot, which makes the surfaces of a scene black. In this situation, FCN-Skip and HourglassNet work well after finetuning on AF-D as shown in Fig.8. Both methods use skip links between each pair of corresponding convolution layers in the encoder and decoder of the networks. In particular, the multi-scale feature maps in VGG-16 used in FCN-Skip appear to be advantageous when used to extract informative features in these situations.

The scene understanding benchmarks in Sec.III-A and III-B inform that its performance is mainly determined by how to preserve both high-level and local information and to aggregate context information. The skip link of SegNet and the multi-scale pooling modules of PSPNet are effective in dealing with the disaster effects.

C. Stereo Matching

We evaluate CNN-based stereo matching methods in fire scenarios, especially those with smoke. First, we benchmarked MC-CNN [23] based on a local window matching-based method, DispNet [11] using semantic information from stereo images, and PSMNet [24] which is an end-to-end network exploiting global context information of input images and a cost volume regularization. We subsequently compared the baseline networks with the finetuned networks to validate the effectiveness of the DISC dataset. In Table.III, we report the results of quantitative evaluations using the bad pixel rate (BPR) and the root mean square error (RMSE) as performance measures.

We are able to draw two conclusions from this experiment. First, finetuning the networks using AF-D worked well in smoky conditions. In particular, the performance improvement was apparent on all baseline approaches in all error measurements. We found that the depth map accuracy increased the most for poorly visible pixels, as shown in Fig.9. Second, we observed that the performance drop of DispNet was more drastic on disaster scenes. The performance degradation of DispNet is likely due to its high-level feature matching strategy. Direct matching with local windows of the MC-CNN or the spatial pyramid pooling strategy of PSMNet are suitable for depth estimation in smoky areas, and training on AF-D allows the performance to be improved in challenging conditions.

We also conducted a qualitative experiment on real-world datasets, and compared the performance of PSMNet with [25] which is a specially designed stereo matching method for defogging. As shown in Fig.10, the real-world results indicate that the CNN-based stereo matching with training on the DISC dataset works well in practice. In particular, the PSMNet finetuned on AF-D in Fig.10(d) shows promising results over [25] even with little computation (1s vs. 10min). In contrast, finding correspondences in scattering media [25] (e.g. fog, haze, or turbid water) causes a heavy computational

Method	BPR3 (%)		BPR5 (%)		EPE (pixel)	
	w/o FT	w/ FT	w/o FT	w/ FT	w/o FT	w/ FT
FlowNet2	20.44	**4.53**	8.25	**1.04**	2.81	**0.69**
DCFlow	19,83	**2.25**	10.10	**0.73**	2.17	**0.30**
PWCNet	18.88	**1.30**	9.15	**0.37**	2.37	**0.31**

TABLE IV: Optical flow. Averaged BPR and EPE in smoke scenes (the lower the better).

(a) Input (b) GT (e) w/o FT (f) w/ FT

Fig. 11: Optical flow results on DISC. Baseline: PWCNet

burden, and spatially-variant smoke density levels in real-world data complicate the matching step in Fig.10(b).

D. Optical Flow

We also evaluated state-of-the-art optical flow estimation methods: FlowNet2 [26], DCFlow [27] and PWCNet [28]. Similar to Sec.III-C, we compared these networks trained on BE-D and finetuned networks on AF-D. Errors are measured regarding the average bad pixel rate and end-point error (EPE) in Table.IV.

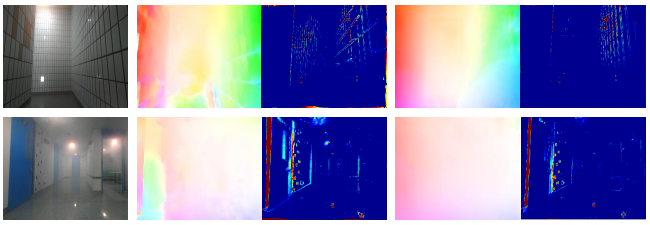
Fires in indoor environments change lighting conditions, which causes inaccurate optical flows. As a result, inferred optical flows exhibit texture-copy artifacts in Fig.11(b), and this tendency could be seen in the real-world imagery as well in Fig.12(b). In this experiment, we observe significant performance improvements when examples are available for training fire situations. The networks trained on AF-D predict accurate optical flows robust to lighting changes and scattering media in Fig.11(c). As shown in Fig.12(c), the realistic disaster simulations of AF-D alleviate the negative influence of varying levels of illumination in real-world scenes.

Note that noticeable performance improvement of PWC-Net is achievable by the use of a pyramid feature extractor and a context network based on dilated convolutions [29]. We found that multi-scale feature extraction and context information from CNNs are beneficial for correspondence estimation, particularly in reduced visibility conditions.

E. Camera Relocalization

Given the 3D geometry of a scene, monocular camera relocalization, which is an important task for rescue robots in disaster scenarios, is used to infer the camera's 6-DoF pose relative to the scene. Following a training and test procedure similar to Sec.III-A, we benchmarked the following state-of-the-art camera relocalization methods: SCoRF [30], which uses a regression forest to predict camera poses by estimating the correspondences between input a RGB-D image[2] and 3D

[2]We used depth maps computed from PSMNet in Sec.III-C as input.



(a) Input (b) w/o FT (c) w/ FT

Fig. 12: Results from PWCNet and interpolation errors between the reference images and warped images on real world scenes.

Method	Normal	Fire (m/ °)		Collapse (m/ °)	
		w/o FT	w/ FT	w/o FT	w/ FT
SCoRF	0.03 / 0.20	0.75 / 6.78	**0.03 / 0.17**	0.58 / 4.48	**0.10 / 0.76**
PoseNet	0.16 / 0.46	4.83 / 31.18	**0.73 / 2.96**	2.09 / 26.35	**0.34 / 1.34**
DSAC	0.03 / 0.19	1.72 / 16.92	**0.18 / 2.49**	0.84 / 17.36	**0.11 / 1.01**

TABLE V: Camera relocalization. Average positional and angular error (the lower the better).

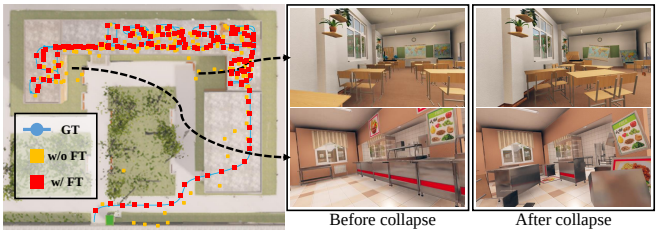


Fig. 13: Camera relocalization result in a collapse scenario.

points in a scene; PoseNet [31], which infers camera poses from a GoogLeNet-style network [32]; and DSAC [33], which is a VGG-style architecture [34] with differentiable sample consensus. We used a furniture shop, an office, a police station and a school, whose spatial extents are 21, 34, 297 and 4942 m^2, respectively.

In Table.V, the significant drop in the pose accuracy can be explained by the fact that disasters drastically affect the appearance of scenes. In particular, because PoseNet does not use any geometry cue to predict camera poses, its result is worse than those by the other methods. After finetuning the networks on AF-D, their overall accuracy improved. We visualize an example of results from PoseNet which shows the most significant performance improvement in Fig.13.

SCoRF achieves the best performance because it takes the RGB-D data of a scene as input. Initial pose hypotheses based on the scene depth account for the uncertainty in the camera location, in spite of scene changes. Scene representation from CNN features and the geometric consensus of DSAC also allow one to infer accurate camera poses even when only partial information is preserved over normal scenes. We conclude that CNN-based camera relocalization robust to disaster scenarios can be achieved if the geometry information of a scene, such as the depth and surface normal, is available.

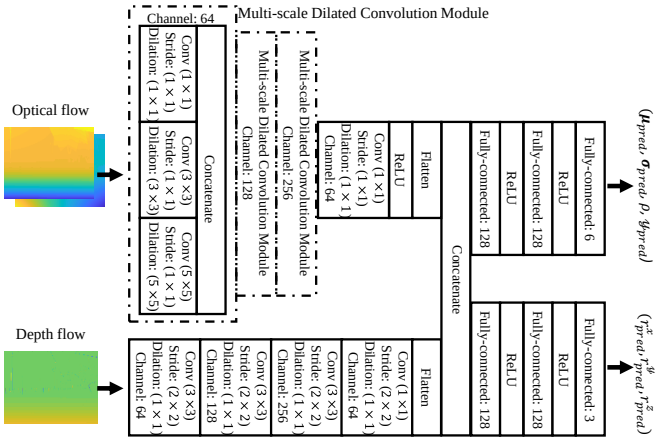


Fig. 14: Overview of the proposed visual odometry network.

IV. Visual Odometry for 3D Map Generation

We can imagine that it is helpful for firefighters if a 3D map indicating ignition spots is available before they go to a building on fire. As an application, we initially estimate the camera poses of each frame through visual odometry using the depth and optical flow and then make a dense 3D map with ignition spots marked as estimated from semantic segmentation.

For visual odometry, we propose a multi-stream network for encoding x, y and z-axis motions between two frames as shown in Fig.14. In a manner similar to LVONet [35], we use the optical flow $\mathcal{O}$ and depth flow $\mathcal{D}$ between frames $t-1$ and t as the input. The input optical flow and depth map are estimated from PWCNet and PSMNet, which show the best performance for fire cases in Sec.III-C and Sec.III-D, respectively. The depth flow $\mathcal{D}$ is defined as follows:

$$\mathcal{D}_t(u,v) = \hat{D}_t((u,v) + \mathcal{O}_t(u,v)) - \hat{D}_{t-1}(u,v), \quad (2)$$

where $\hat{D}_t$ is the depth map of frame t and (u,v) represents the pixel coordinates.

The input optical flow passes through three multi-scale dilated convolution modules consisting of three convolution layers with different filter sizes and dilations. The output filters are concatenated into a single output vector forming the input of the next layer [32]. Receptive fields with various sizes are beneficial for camera pose estimations because input optical flows computed from fire scenes are noisy. We also encode the depth flow using four convolution layers. We empirically observe that embedding the depth flow feature prior to the fully-connected layers enables a significant improvement in the pose estimation accuracy by correcting the motion distortions caused by noisy optical flows.

Instead of direct regression of camera poses from CNNs like [31], [36], we use the bivariate Gaussian probabilistic density function based on the visual odometry uncertainty from CNNs [37]. Our network predicts the outputs of translation $\boldsymbol{r}_{pred} = (\boldsymbol{\mu}_{pred}, \boldsymbol{\sigma}_{pred}, \rho, y_{pred})$ and rotation $(r^x_{pred}, r^y_{pred}, r^z_{pred})$ in Euler angles. $\boldsymbol{\mu} = (\mu_x, \mu_z)$ are two mean variables in the x, z-axis motion, $\boldsymbol{\sigma} = (\sigma_x, \sigma_z)$ are the corresponding standard deviations, ρ is the correlation coefficient of the translation between the x and z-axis motion and y_{pred} is the translation of the y-axis. Using the predicted outputs, we minimize the loss function $\mathcal{L}$, as below:

$$\begin{aligned}\mathcal{L} = \sum \Big[&- \log\Big(\exp(-0.5\Psi)/\Gamma\Big) \\ &+ \lambda_1 \|y_{gt} - y_{pred}\|_2 + \lambda_2 \|\boldsymbol{r}_{gt} - \boldsymbol{r}_{pred}\|_2 \Big] \\ \text{s.t.} \quad &\Psi = (\boldsymbol{\mu}_{pred} - \boldsymbol{\mu}_{gt})^T \Omega^{-1} (\boldsymbol{\mu}_{pred} - \boldsymbol{\mu}_{gt}), \\ &\Gamma = (2\pi)^2 |\Omega|^{0.5} \quad \text{and} \quad \Omega = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_z \\ \rho\sigma_x\sigma_z & \sigma_z^2 \end{bmatrix}, \end{aligned} \quad (3)$$

where $\|\cdot\|_2$ is the $L2$-norm and $|\cdot|$ is the determinant of the matrix. λ_1 and λ_2 are user-defined parameters and are both set to 0.1 in our implementation. At test time, we draw random samples from the bivariate normal distribution for the mean and standard deviation of the x, z-axis motions. We set the number of samples to 10K, and the final translation is obtained by averaging the samples.

In the training procedure, we use a furniture shop, police station, warehouse, and city from BE-D with the ground-truth depth, the optical flow and the camera poses. We train our network from scratch with 32K iterations in total and use the ADAM optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$), whose starting learning rate is 10^{-4} and the decay is set to 0.5 for every 5K iterations. The spatial resolution of the input optical flow and depth flow is 120×320 and the batch size is 50. The training is performed with Tensorflow on an NVidia 1080 GPU, which usually takes 12 hours.

We compare the performance of the proposed network

Method	Positional error (m)		Angular error ($^\circ$)	
	w/o FT	w/ FT	w/o FT	w/ FT
LVONet	6.6118	**3.7440**	1.1847	**0.7279**
Proposed	6.9023	**1.5479**	0.3756	**0.0690**

TABLE VI: The comparison of the performance of the proposed network against LVONet. We report average positional and angular error (the lower the better).

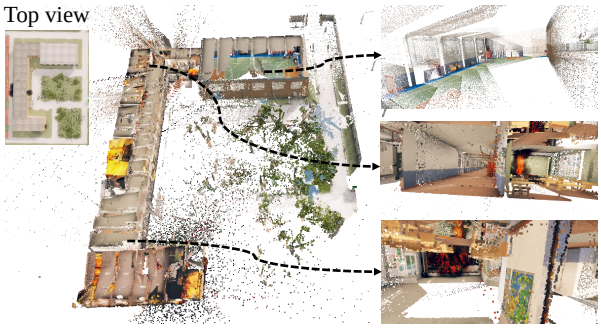


Fig. 15: 3D map reconstruction in a building on fire (School). Ignition spots are marked in red.

with LVONet[3] in Table.VI for a subway station, a park, a school and a residence. The input optical flow and depth map estimated from the finetuned networks provide the most performance improvement of the visual odometry task. Our multi-scale dilated convolution module enables the handling of noisy inputs by aggregating multi-scale motion information. Using the estimated camera poses and depth maps, we make a 3D map indicating the ignition spots, as shown in Fig.15. The fire labels estimated from SegNet are mapped to the 3D map, which represents regions with burring.

V. Conclusion

We present a new dataset for simulating disaster scenarios, termed the *DISC* dataset. The dataset includes ground-truth data for low-level and high-level computer vision tasks. The total number of images exceeds 300K sequential images for 15 different places. As demonstrated by the experiments, state-of-the-art CNNs trained by the DISC dataset were highly effective in disaster conditions, especially those with real-world disaster imagery.

We expect that the release of our challenging datasets with various scenes and realistic disaster effects will stimulate the development of new computer vision tasks, such as obstacle avoidance and trip hazard affordance [38]. As future work, we plan to simulate more challenging disaster conditions such as underwater conditions [39], [40] and to generate multi-spectral images in virtual worlds for fire-fighting equipment [41].

References

[1] W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang, "Single image dehazing via multi-scale convolutional neural networks," in *ECCV*, 2016.

[2] B. Cai, X. Xu, K. Jia, C. Qing, and D. Tao, "Dehazenet: An end-to-end system for single image haze removal," *IEEE Trans. on Image Proc.*, vol. 25, no. 11, pp. 5187–5198, 2016.

[3] J. Chen, J. Chen, H. Chao, and M. Yang, "Image blind denoising with generative adversarial network based noise modeling," in *CVPR*, 2018.

[4] H. Zhang and V. M. Patel, "Density-aware single image de-raining using a multi-stream dense network," 2018.

[5] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *ICCV*, 2017.

[6] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.

[7] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, and Y. Wang, "Unrealcv: Virtual worlds for computer vision," in *ACM on Multi. Conf.*, 2017.

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *ICCV*, 2015.

[9] A. Ley, R. Hänsch, and O. Hellwich, "Syb3r: A realistic synthetic benchmark for 3d reconstruction from images," in *ECCV*, 2016.

[10] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *ICRA*, 2014.

[11] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016.

[12] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for uav tracking," in *ECCV*, 2016.

[13] B. Kaneva, A. Torralba, and W. T. Freeman, "Evaluation of image features using a photorealistic virtual world," in *ICCV*, 2011.

[14] "Unity," https://unity3d.com/.

[3]Because there is no public source code of LVONet, we directly implement it. For a fair comparison, we confirmed that our implementation matches the performance in the original paper [35] on the KITTI dataset.

[15] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *CVPR Workshops*, 2017.

[16] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. on Patt. Anal. and Mach. Intel.*, vol. 40, no. 4, pp. 834–848, 2018.

[18] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[19] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "Denseaspp for semantic segmentation in street scenes," in *CVPR*, 2018.

[20] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015.

[21] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser, "Physically-based rendering for indoor scene understanding using convolutional neural networks," in *CVPR*, 2017.

[22] W. Chen, D. Xiang, and J. Deng, "Surface normals in the wild," in *ICCV*, 2017.

[23] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. of Mach. Learn. Research*, vol. 17, no. 2, pp. 1–32, 2016.

[24] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *CVPR*, 2018.

[25] Z. Li, P. Tan, R. T. Tan, D. Zou, S. Zhiying Zhou, and L.-F. Cheong, "Simultaneous video defogging and stereo reconstruction," in *CVPR*, 2015.

[26] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *CVPR*, 2017.

[27] J. Xu, R. Ranftl, and V. Koltun, "Accurate Optical Flow via Direct Cost Volume Processing," in *CVPR*, 2017.

[28] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *CVPR*, 2018.

[29] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[30] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *CVPR*, 2013.

[31] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2015.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions," in *CVPR*, 2015.

[33] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "Dsac-differentiable ransac for camera localization," in *CVPR*, 2017.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[35] C. Zhao, L. Sun, P. Purkait, T. Duckett, and R. Stolkin, "Learning monocular visual odometry with dense 3d mapping from dense 3d flow," in *IROS*, 2018.

[36] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using lstms for structured feature correlation," in *ICCV*, 2017.

[37] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *Int. J. of Robot. Research*, vol. 37, no. 4-5, pp. 513–542, 2018.

[38] S. McMahon, N. Sünderhauf, B. Upcroft, and M. Milford, "Multimodal trip hazard affordance detection on construction sites," *IEEE Trans. Robot.*, vol. 3, no. 1, pp. 1–8, 2018.

[39] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Robot. Automat. Lett.*, vol. 23, no. 2, pp. 331–341, 2007.

[40] C. Ancuti, C. O. Ancuti, T. Haber, and P. Bekaert, "Enhancing underwater images and videos by fusion," in *CVPR*, 2012.

[41] "Flir k2 thermal imaging camera," http://ww4.flir.com/k2/.