IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2022, accepted April 12, 2022, date of publication April 18, 2022, date of current version April 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3167763

Benchmark Analysis of Semantic Segmentation Algorithms for Safe Planetary Landing Site Selection

THOMAS CLAUDET, KENTO TOMITA, AND KOKI HO
Georgia Institute of Technology, Atlanta, GA 30332, USA

Corresponding author: Koki Ho (kokiho@gatech.edu)

This work was supported in part by the National Aeronautics and Space Administration (NASA) through the NASA Early Career Faculty Program under Grant 80NSSC20K0064; and in part by the Research Cyberinfrastructure Resources and Services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, GA, USA.

ABSTRACT This paper presents an in-depth analysis of state-of-the-art semantic segmentation algorithms applied to spacecraft safe planetary landing via hazard detection and avoidance. Several architectures are trained from binary safety maps and the rich dataset of the High-Resolution Imaging Science Experiment (HiRISE) embedded on Mars Reconnaissance Orbiter for realistic purposes. The study incorporates several metrics comparisons such as recognition accuracy, computational complexity, model complexity, and inference time. The proposed performance indices and combinations are analyzed and discussed. The experiments were performed using a Raspberry Pi 4B, which is a relevant commercial-of-the-shelf microcontroller surrogate of NASA's High-Performance Spaceflight Computer (HPSC) that will thrive within the next decades in space exploration. This paper allows researchers to know what has been tested on the subject and serves as a catalog for users to pick the most relevant architecture for their own application.

INDEX TERMS Artificial neural networks, image segmentation, space exploration, aerospace safety.

I. INTRODUCTION

Safe landing is by far the most critical part of every space mission aiming at conducting experiments on the ground. First performed by humans during the Apollo program with Apollo 11 [1], it switched to fully autonomous with the exploration of Mars as distances increased. Engineers even call the entry, descent, and landing phase on Mars the seven minutes of terror, as it takes 7 minutes to the spacecraft to safely land on the planet, and from which the outcome can only be seen after the landing has occurred due to this important delay.

At first, these landers developed and managed by NASA's Jet Propulsion Laboratory (JPL) had a predefined trajectory in their flight computer, and diversion was not possible. This was the case for all landers before the year 2020 with Viking 1 & 2 [2], [3], Pathfinder and its Sojourner rover [4], Spirit [5], Opportunity [6], Phoenix [7], Curiosity [8], and Insight [9]. On February 18, 2021, Terrain Relative Navigation (TRN), a revolutionary technique able to safely and autonomously land between hazards was first tested in real conditions during the Mars 2020 mission embedding the Perseverance rover and the Ingenuity helicopter [11], as shown on Fig. 1. The lander would take photos during its descent, compare them to its orbital map, and divert if necessary.

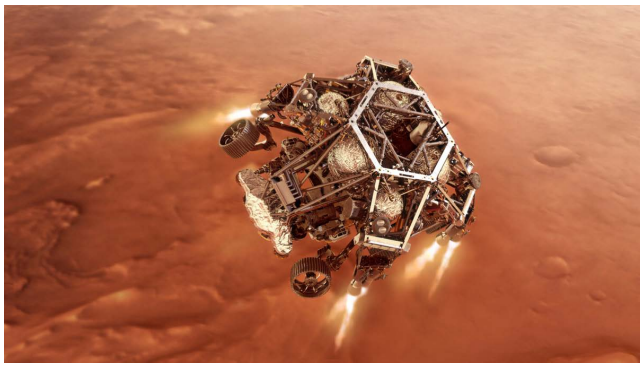


FIGURE 1. Powered descent for perseverance [10]): NASA's perseverance rover fires up its descent stage engines as it nears the martian surface in this illustration. Credit: NASA/JPL-Caltech.

Limited by the capabilities of space-rated flight computers, continuous analysis of the terrain and computation of new

The associate editor coordinating the review of this manuscript and approving it for publication was Kumaradevan Punithakumar.

trajectories combined with increasing degrees of safety are becoming computationally too demanding and new technologies are now required to overcome this challenge. This is why NASA's JPL, along with industry partners, is developing the High-Performance Spaceflight Computer (HPSC) [12]. A hundred times more efficient than its monolithic counterpart, it will push the boundaries of space exploration, and allow missions that were thought impossible to happen in the next decades.

On the other hand, artificial intelligence, and especially machine learning and computer vision have been greatly explored and are today two of the most active fields studied by researchers around the world [13]. Offering the capabilities of learning optimal decisions to be retrieved orders of magnitude faster than classical approaches has shed light on attainable breakthroughs in various disciplines and domains in modern science and engineering. Specifically, deep neural networks have allowed semantic segmentation, a computer vision technique, to arise and enable the classification of each pixel in an image from a predefined set of classes, and now starts playing a role in spacecraft safe planetary landing.

Throughout the landing of the spacecraft, many technical challenges are to be anticipated in order to robustify mission success. The martian surface, and more specifically the regions of scientific interest, are full of obstacles such as craters, cliffs, cracks and jagged boulders, which in turn set high requirements on the safety level prediction accuracy of the landing sites [14]. Furthermore, as for Hayabusa's first rehearsal descent, it could be possible that the original landing site was surrounded by small but fatal hazards, which could not be detected until the vehicle is close enough to the surface [15]. This then constraints the algorithms to handle increased resolutions. Further, these vision processes have to fit on space-rated hardware and satisfy mission requirements such as inference time or prediction accuracy, thus being memory and computationally efficient. These algorithms must finally be trained on realistic data and be robust to sensor noise.

The main contributions of this paper can be summarized as follows:

- Choice of the HPSC surrogate microcontroller for realistic performance predictions.
- Training on realistic and noisy data to obtain performance metrics during inference time of state-of-the-art semantic segmentation algorithms for binary safety maps generation.
- Benchmark of all the algorithms and architectures in several metric spaces such as accuracy, memory consumption and inference time to find the most suitable one for the landing problem.

The structure of this paper is as follows: Section § III finds a relevant commercial-off-the-shelf surrogate of the HPSC. Section § IV presents the algorithms and the metrics being compared. Section § V introduces and discusses the numerical results. Section §VI finally draws conclusions and perspectives on this work.

II. LITERATURE REVIEW

In prior studies, the necessity of providing safe landing for autonomous systems started with the emergence of Unmanned Aerial Vehicles (UAVs). The computer vision community played an essential role with primal developments in visual odometry ranging from monocular vision [16] to stereo vision [17], [18], providing depth estimation, or feature-based methods [19]. Moreover, Simultaneous Localization and Mapping (SLAM) [20], mostly using LiDAR-based point clouds, allowed the generation of 3D maps, and thus to estimate the topology of the terrain to land on. Safe landing was also thoroughly studied from the perspective of LiDAR-based ground filtering algorithms based on the process of Digital Elevation Maps (DEMs) [21]. Many other techniques were studied such as Stereo-Ranging [22], using two cameras with different angles to analyze pixel disparity and estimate depth, Structure from Motion (SFM) [23] to reconstruct 3D terrain from 2D-image sequences, Homography Estimation and Adaptive Control (HEAC) for image rectification and registration [24], but also Color Segmentation [25] and Optical Flow [26].

Pixel classification from semantic segmentation was used for numerous applications ranging from autonomous driving [27], robotic navigation and localization [28], to scene understanding [29]. Semantic segmentation based on deep learning was also involved in UAV visual landing site detection, such as with the PSPNet architecture [30]. Since recently, semantic segmentation has started playing a role in space engineering, and in particular for spacecraft safe planetary landing with networks like U-Net [31], [32]. Another main technical challenge of this task is that the state-of-the-art algorithm, namely the Autonomous Landing and Hazard Avoidance Technology (ALHAT) [33], [34] only considers local features, whereas a global understanding of the entire image could provide more insight, and this is where semantic segmentation motivates this work. During the descent phase, every time the LiDAR-based cameras retrieve a point cloud of the ground, a digital elevation map is created. The semantic segmentation algorithm is then able to classify every pixel and predict its safety level, namely identify the nature of each landing site. This method has already proven to provide better results than ALHAT [35]. The benchmark study of the proposed paper is for the base architectures without uncertainty quantification, and adding them would increase the performance as shown in [36].

III. NASA's HIGH-PERFORMANCE SPACEFLIGHT COMPUTER's SURROGATE

To make realistic experiments, a relevant substitute of NASA's High-Performance Spaceflight Computer Surrogate (HPSC) must be chosen. This section gives the latest specifications of the HPSC and finds a surrogate of it.

To do that, the comparison is based on FLOPS, which is a measure of how many floating-point operations a microcontroller is able to perform per second. It is a function of the number of cores, their clock frequency, the number of

floating-point operations performed at each cycle, and the computing method that enables processing of multiple data with a single instruction, also known as SIMD capabilities. The FLOPS are then defined by the following equation:

$$\text{FLOPS} = \text{cores} \times \frac{\text{cycles}}{\text{second}} \times \frac{\text{FLOPs}}{\text{cycle}} \times \text{SIMD} \quad (1)$$

In the case of NASA's HPSC, some information could be found from the preliminary design [12]. Here are listed the paper's assumptions about the expected HPSC specifications.

- Cores: A minimum of eight 64b CPU cores on the HPSC System-on-Chip (SoC) die, each of which contains integer, double-precision floating-point, and vector processing capability. It is desired that all eight cores be fully coherent.
- Integer Performance
  - Minimum CoreMark (per core, @ 1GHz): 5,100
  - Minimum CoreMark (overall, for 8 cores @ 1GHz): 36,720
- Floating-Point Performance
  - Minimum FP performance (per core): Two double-precision (64b) FP ops per cycle
  - Minimum FP performance (overall): Sixteen double-precision (64b) FP ops per cycle
- Vector/SIMD Performance
  - Minimum Vector performance (per core): Two 64b vector ops per cycle
  - Minimum Vector performance (overall): Sixteen 64b vector ops per cycle
- Clock: All of the CPU cores, floating-point engines, and vector units shall operate at a minimum frequency of 1 GHz. Note the microcontroller core used for boot, health, and configuration can operate less than 1GHz.

Taking into account the highly used parallelization for neural networks (e.g. matrix multiplications and additions), it is reached:

$$8 \text{ cores} \times 1\text{GHz} \times 2 \text{ FLOPs/cycle/core} \times 2 \text{ SIMD} = 32 \text{ GFLOPS} \quad (2)$$

In comparison, Raspberry Pi 4B [37] achieves a theoretical maximum of:

$$4 \text{ cores} \times 1.5\text{GHz} \times 3 \text{ FLOPs/cycle/core} \times 2 \text{ SIMD} = 36 \text{ GFLOPS} \quad (3)$$

This means that to compare effectively with Commercial off-the-shelf (COTS) hardware, the easiest approach is to reduce the clock frequency of the Raspberry pi 4B to match the 32 GFLOPS of HPSC, theoretically by

$$\eta = 32/36 = 0.89 \quad (4)$$

.

So the frequency applied to Raspberry Pi 4B becomes:

$$f_{RPI_{4B}} = \eta \times 1.5GHz = 1333\ MHz. \quad (5)$$

IV. ARCHITECTURES AND METRICS

In this section, it is presented the semantic segmentation architectures and metrics that are being compared.

A. ARCHITECTURES

The following architectures are all composed of convolutional layers which govern the overall time complexity of the network. They are based on the 2D cross-correlation operation and lead, following [38], for squared input and kernel, to:

$$\mathcal{C}_t = \mathcal{O}\left(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right), \quad (6)$$

with $\mathcal{C}_t$ the time complexity, l the index of the layer, n_{l-1} the number of input channels of layer l, n_l the width, or number of filters, s_l the spatial size of the filter, and m_l, the size of the output.

1) SegNet: 5, 4, AND 3 ENCODING-DECODING LAYERS

The first semantic segmentation model with which this paper deals with is SegNet [39], depicted on Fig. 2. This is an encoder-decoder network with a pixel-wise classification layer. His encoder is similar as the VGG16 network. Unlike other models, SegNet stores the pooling indices and retrieves them during the umsampling step. In this study, it is benchmarked three types of SegNets. The first one is the original with 5 layers for the encoding and 5 others for the decoding step. It is also tested with 4 and 3 on each side of the low-dimensional (or latent) space to see the effect of the parameters reduction on the accuracy.

2) FCN (FULLY CONVOLUTION NETWORK): 32s, 16s, AND 8s

Fully Convolutional Networks [40], as shown on Fig. 3, share the same encoding process as for SegNet, but the indices are not stored to perform the upsampling. For that specific task of retrieving the input size, there are several types of options. The first one is called FCN-32s. From the latent space, it is directly upsampled or interpolated to the original image size. A lot of information is then lost. Then, it is compared FCN-16s.

In that case, the last step is upsampled by 2, the scores are summed with the step before, and the result is upsampled to the original image size.

In the final case, FCN-8s, the result of FCN-16s are upsampled by 2, the score are added with the step before, and the result is upsampled. The accuracy should increase from FCN-32s, to FCN-16s, to FCN-8s, as the upsampling loses less and less information. However, it is more computationaly heavy.

3) ICNet (IMAGE CASCADE NETWORK)

The Image Cascade Network (ICNet) [41], represented on Fig. 4, works in several steps. First, the low-resolution images reach the end of the network and a first mapping is obtained.

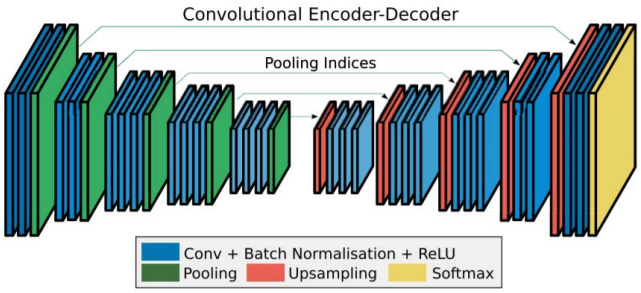


FIGURE 2. SegNet [39] is an encoder-decoder network with a pixel-wise classification layer. His encoder is similar to the VGG16 network. Unlike other models, SegNet stores the pooling indices to retrieve them during the umsampling step. Licensed under CC BY 4.0.

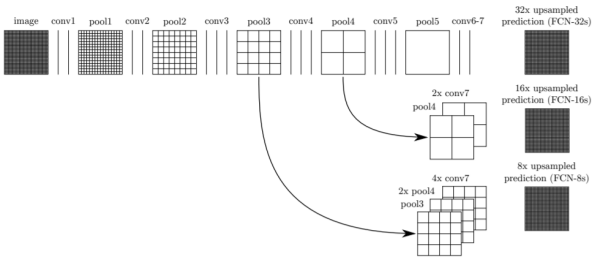


FIGURE 3. Fully Convolutional Networks [40] share the same encoding process as for SegNet, but the indices are not stored to perform the upsampling. The three types of FCNs differ by their upsampling part, which incorporate different levels of feature resolutions. © [2017] IEEE. Reprinted, with permission, from [40].

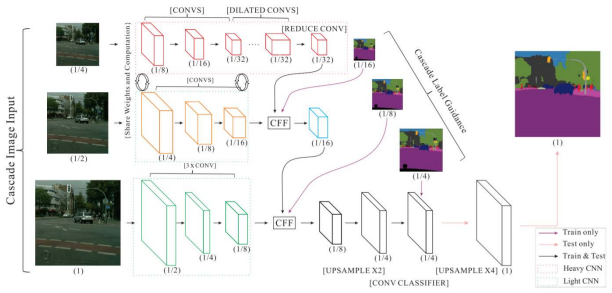


FIGURE 4. The Image Cascade Network (ICNet) [41] works in several steps. First, the low-resolution images reach the end of the network and a first mapping is obtained. On top of that, medium and high resolution features refine it with cascade feature fusion unit and cascade label guidance strategy. Reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature Computer Vision @ ECCV 2018 [41], © (2018).

On top of that, medium and high resolution features refine it with cascade feature fusion unit and cascade label guidance strategy.

4) U-NET (U-SHAPE NETWORK) AND TransUNet

U-Net [42], on Fig. 5, is based upon FCN but should in theory provide a better precision with less training images.

The main idea is to replace pooling operations by upsampling operators to improve the resolution of the output. Also, the skip connections (copy and crop) technique in U-Net copies the image matrix from the earlier layers (left-hand side of Fig. 5) and uses it as a part of the later layers (right-hand side layers). With this, the model preserves rich information, thus reducing information loss. The many channels in the decoder part helps propagating context information to layers with higher resolution and this symmetry leads to the U-shaped architecture.

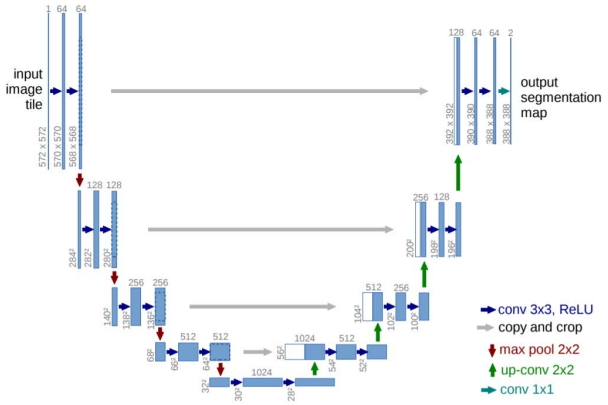


FIGURE 5. U-Net [42] is based upon FCN but should in theory provide a better precision with less training images. The main idea is to replace pooling operations by upsampling operators to improve the resolution of the output. Also, the skip connections (copy and crop) technique in U-Net copies the image matrix from the earlier layers (left-hand side) and uses it as a part of the later layers (right-hand side). With this, the model preserves rich information, thus reduces information loss. The many channels in the decoder part helps propagating context information to layers with higher resolution and this symmetry leads to the U-shaped architecture. Reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature MICCAI 2015 [42], © (2015).

Regarding the TransUNet architecure [43], it adds Vision Transformers (ViT, concept of self-attention) to record comprehensive localization information at all network stages while conveying the context over a long-range within the network.

5) ENet (EFFICIENT NEURAL NETWORK)

ENet [44], also uses an encoder-decoder scheme and is more adapted to real-time applications. The idea is to early downsample in the decoder part to reduce the cost of processing large input frames. ENet also takes advantage of PReLUs as an activation function, dilated convolutions and spatial dropout.

6) ConvDeconv

ConvDeconv [45] is a semantic segmentation network following similar ideas as for SegNet but with a lot less layers and with smaller kernel sizes.

7) DeepLabV3 WITH A ResNet BACKBONE

The last architecture that was benchmarked is the DeepLabv3 [46]. Atrous convolution are used in cascade or in parallel to segment objects at multiple scales. State-of-the-art networks attach a ResNet (Residual network) backbone to it, acting as its main feature extractor. There exist several variants of this tandem, such as when changing the number of layers (ResNet-N), the number of groups / width per group (ResNeXt), or making the bottleneck number of channels twice larger in every block (Wide).

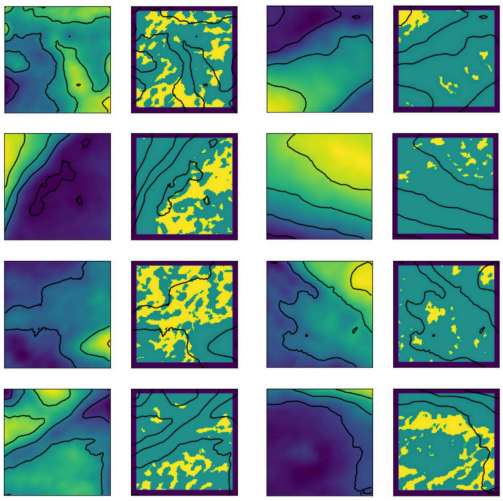


FIGURE 6. Examples of the training dataset. The digital elevation maps are processed from the Mars HiRISE experiment and fed to the network to learn what are the safe (green) and unsafe (yellow) parts of a camera view. Bright colors indicate relatively high elevation in the image.

B. PERFORMANCE METRICS

It is now presented the performance metrics that are being used to benchmark the different algorithms and architectures on the same basis.

1) RECOGNITION ACCURACY

The accuracy is representing the pixel accuracy in this study, which is the number of pixels that were correctly judged as safe or unsafe.

2) COMPUTATIONAL COMPLEXITY

The computational complexity is represented by the number of floating-point operations (or GFLOPs). The requirement is that is must be below the capabilities of HPSC of 32 GFLOPS.

3) MODEL COMPLEXITY

The model complexity is obtained by counting the number of learnable parameters. This is important regarding the memomy capabilities of the hardware.

4) INFERENCE TIME

The inference time is the duration that needs the network to perform one forward pass. For statistical purposes, it was averaged over 100 runs.

V. NUMERICAL RESULTS

A. INPUT DATASET AND TRAINING

1) DATASET

The data that was used in this study is directly imported from the Mars HiRISE camera on-board the Mars Reconnaissance Orbiter (MRO). The resolution is 1m/pixel and of size 100×100 upsampled with bilinear interpolation to 128×128 before being fed to the network.

For the ground-truth label, the ALHAT probabilistic algorithm is not used. It is instead measured slope and roughness just as ALHAT does, but safety maps are deterministically generated from the noise-free Digital Elevation Maps.

The dataset contained a total of 1000 normalized Digital Element Maps (DEMs), with 800 for training, 100 for validation, and the remaining 100 for testing. Examples of which can be seen on Fig. 6. All images are independent from one another as there was no data augmentation strategies involved.

2) TRAINING

Integrated in the Pytorch framework, the models were trained using the Adaptative Moment Estimation (Adam) stochastic optimizer. A batch size of 8 was used for the 1 million epochs of training, with a learning rate of 0.00001. The training took several days on an NVIDIA Quadro RTX 6000 on the Georgia Institute of Technology's High-Performance PACE Cluster. The benchmark was however performed online with the chosen HPSC surrogate, namely the Raspberry Pi 4B, as discussed in section III.

B. BENCHMARK

1) ACCURACY VS COMPUTATIONAL COMPLEXITY VS MODEL COMPLEXITY

The ball chart in Fig. 7 gives the accuracy of each model with respect to its computational complexity on the X-axis, and its model complexity (number of parameters) as the size of the ball. The highest pixel accuracy is reached by ConvDeconv with 95% which also has the least amount of parameters. It seems to be no direct relationship between the accuracy and GFLOPS observing that ICNet with respect to ENet has similar computational complexity but with 10 less percentage points. Further more, the same example also shows that having a large number of parameters does not imply a better accuracy. This is actually by reducing the number of learnable parameters and computational operations that the best performance occurs.

2) ACCURACY-RATE VS LEARNING POWER

The accuracy density, which is the ratio of the recognition performance by the number of parameters to achieve that result. It is first observed on Fig. 8 that TransUNet and the DeepLabv3s have the lowest results. On the other hand, architectures such as ConvDeconv, Enet, and SegNet 3 make a better use of their model complexity. This accuracy density is compared with the actual accuracy of each model. ConvDeconv, which the the most accurate model, uses 5 times more efficiently its parameters than SegNet 3 and also has a better accuracy of 2.5 percentage points.

3) ACCURACY-RATE VS INFERENCE TIME

The inference time of each model computed from the Raspberry Pi 4B is finally shown on Fig. 9. The grid plot shows 3 classes. The first one regroups the architectures

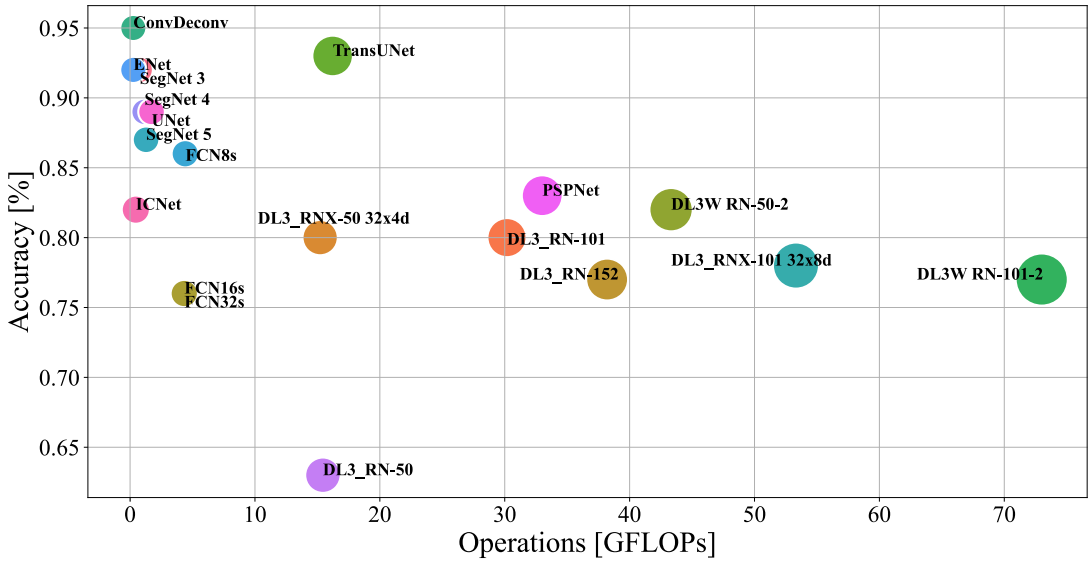


FIGURE 7. This ball chart gives the accuracy of each model with respect to its computational complexity on the X-axis, and its model complexity (number of parameters) as the size of the ball. The highest pixel accuracy is reached by ConvDeconv with 95% which also has the least amount of parameters. It seems to be no direct relationship between the accuracy and GFLOPs observing that ICNet with respect to ENet has similar computational complexity but with 10 less percentage points. Furthermore, the same example also shows that having a large number of parameters does not imply a better accuracy. This is actually by reducing the number of learnable parameters and computational operations that the best performance occurs.

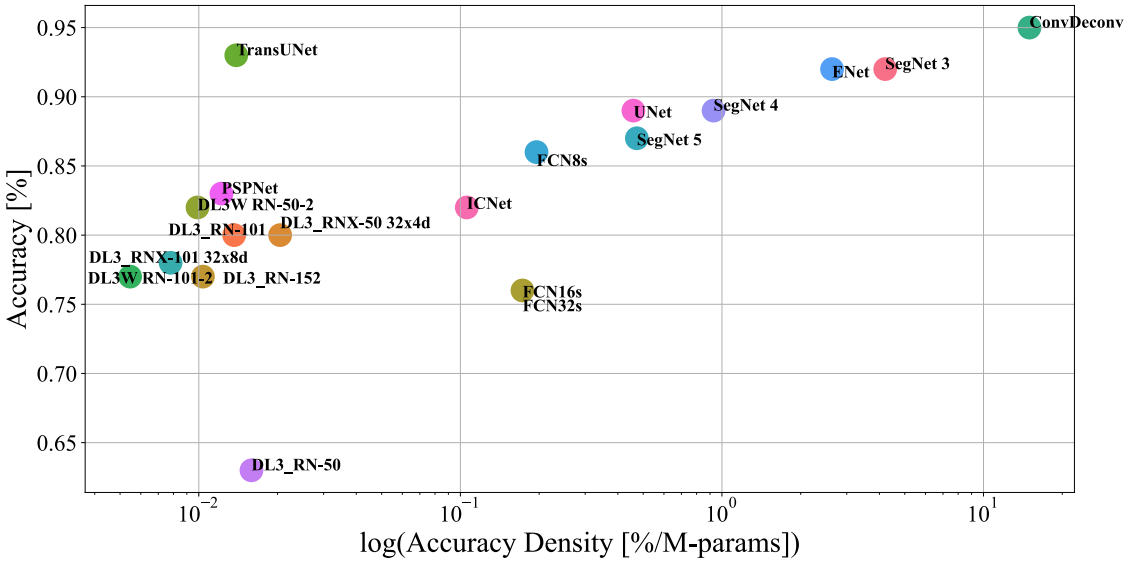


FIGURE 8. The accuracy density, which is the ratio of the recognition performance by the number of parameters to achieve that result. It is first observed that TransUNet and the DeepLabv3s have the lowest results. On the other hand, architectures such as ConvDeconv, Enet, and SegNet 3 make a better use of their model complexity. This accuracy density is compared with the actual accuracy of each model. ConvDeconv, which the the most accurate model, uses 5 times more efficiently its parameters than SegNet 3 and also has a better accuracy of 2.5 percentage points.

that perform 5 to 10 forward passes in lass than 2 seconds, namely ConvDeconv and ICNet. Then, 5 models perform their inference between 0.2 and 1 second. This group contains all the SegNets, as well as U-Net and ENet. Finally, all fully connected networks, TransUNet, and the DeepLabv3s take more than 1 second to perform the prediction. Among the architectures that produce their safety map is less than 1 second, ConvDeconv gives the best accuracy. It is the fastest, but also the most accurate. It can also be noted that among the SegNets, SegNet 3, the smallest one in terms of computational and model complexity, achieves the fastest and most accurate performance.

Since the accuracy and the inference time are the two most important factors, the models that give the best performance with respect to those metrics is ConvDeconv. Following Table 1, this architecture is the best in terms of almost all the computed metrics, namely inference time, specificity (true unsafe rate), fall-out (false safe rate), computational and

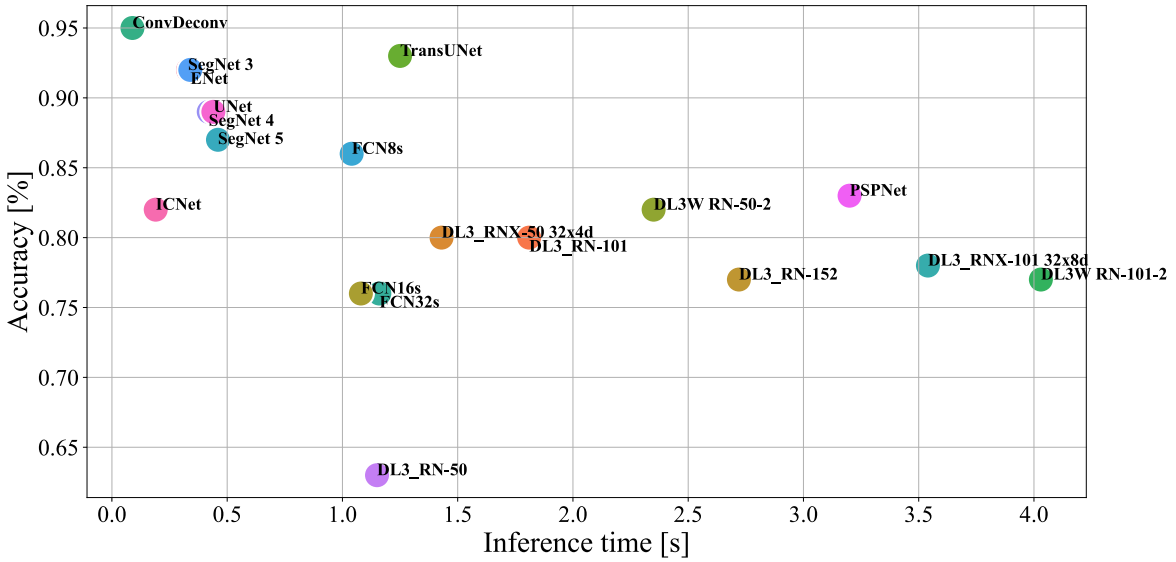


FIGURE 9. The inference time of each model computed from the Raspberry Pi 4B is finally shown. The grid plot shows 3 classes. The first one regroups the architectures that perform 5 to 10 forward passes in lass than 2 seconds, namely ConvDeconv and ICNet. Then, 5 models perform their inference between 0.2 and 1 second. This group contains all the SegNets, as well as U-Net and ENet. Finally, all fully connected networks, TransUNet, and the DeepLabv3s take more than 1 second to perform the prediction. Among the architectures that produce their safety map is less than 1 second, ConvDeconv gives the best accuracy. It is the fastest, but also the most accurate. It can also be noted that among the SegNets, SegNet 3, the smallest one in terms of computational and model complexity, achieves the fastest and most accurate performance.

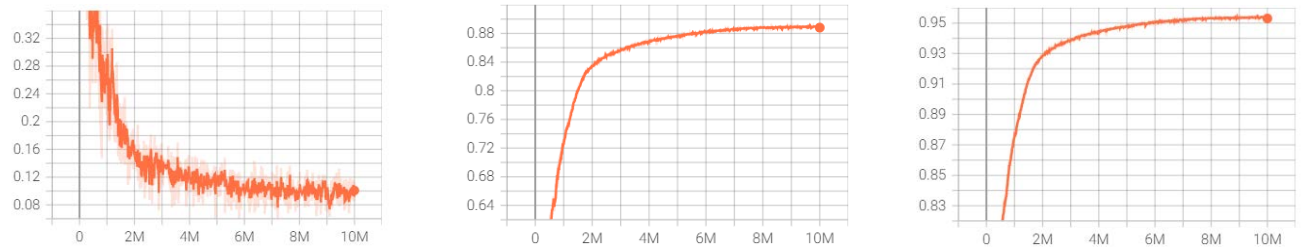
TABLE 1. Results for every model obtained with the Rasberry Pi 4B. mIoU is the mean intersection over union, T-S/S the fraction of true safe pixels (sensitivity), F-US/S the percentage of pixels that were predicted as unsafe when they were actually safe (miss rate), T-US/US (specificity) the ratio of pixels that were correctly labeled as unsafe, F-S/US those which were incorrectly labeled as safe when they were unsafe (fall-out), GFLOPs the number of billions of FLoating OPerations used for one inference (computational complexity), and # of parameters is the number of parameters (model complexity). Best values are written using boldface letters. It can be observed that the shallower the architecture such as with SegNet 3 or ConvDeconv, the better the performance in terms of inference time, accuracy, computational complexity, and memory usage on this specific task.

Model	Inference time (s)	Pixel Accuracy	mIoU	T-S/S	F-US/S	T-US/US	F-S/US	GFLOPs	# of parameters
SegNet 5	0.46	0.87	0.72	0.80	0.20	0.89	0.11	1.3	1.8e6
SegNet 4	0.42	0.89	0.76	0.89	0.11	0.89	0.11	1.2	9.6e5
SegNet 3	0.33	0.92	0.81	**0.94**	**0.06**	0.91	0.09	0.78	2.2e5
FCN32s	1.2	0.76	0.47	0.21	0.79	0.94	0.06	4.3	4.4e6
FCN16s	1.1	0.76	0.47	0.23	0.77	0.94	0.06	4.3	4.4e6
FCN8s	1.0	0.86	0.47	0.66	0.34	0.92	0.08	4.4	4.4e6
ICNet	0.19	0.82	0.68	0.80	0.20	0.83	0.17	0.46	7.8e6
U-Net	0.44	0.89	0.75	0.85	0.15	0.90	0.10	1.7	1.9e6
TransUNet	1.3	0.93	0.83	0.89	0.11	0.94	0.06	16	6.7e7
PSPNet	3.2	0.83	0.62	0.52	0.48	0.93	0.07	33	6.8e7
ENet	0.34	0.92	0.81	0.94	0.06	0.91	0.09	0.26	3.5e5
ConvDeconv	**0.089**	**0.95**	**0.89**	0.92	0.08	**0.97**	**0.03**	**0.26**	**6.4e4**
DeepLabv3_ResNet-50	1.2	0.63	0.41	0.43	0.57	0.70	0.30	20	4.0e7
DeepLabv3_ResNet-101	1.8	0.80	0.57	0.50	0.50	0.89	0.11	30	5.9e7
DeepLabv3_ResNet-152	2.7	0.77	0.51	0.38	0.62	0.89	0.11	38	7.4e7
DLv3_ResNeXt-50 32x4d	1.4	0.80	0.60	0.58	0.42	0.88	0.12	20	3.9e7
DLv3_ResNeXt-101 32x8d	3.52	0.78	0.56	0.52	0.48	0.86	0.14	53.32	1.0e8
DLv3_Wide ResNet-50-2	2.4	0.82	0.61	0.56	0.44	0.90	0.10	43	8.3e7
DLv3_Wide ResNet-101-2	4.0	0.77	0.54	0.49	0.51	0.87	0.13	73	1.4e8

model complexity. Only SegNet 3, which is also a shallow architectures with respect to the others, shows approaching results, even producing better sensitivity (true safe rate), and miss rate (false unsafe rate).

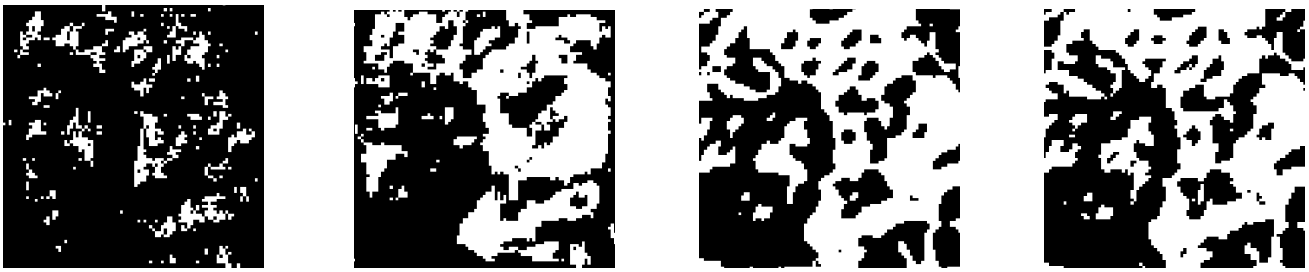
4) RESULTS FOR ConvDeconv

The study finally zooms in on ConvDeconv, the chosen architecture for the landing problem. On Fig. 10, it is shown the training results. The loss function decreases and finally plateaus at the same step at the Mean Intersection over Union (mIoU), and the Pixel Accuracy. As stated in section V, the accuracy of ConvDeconv is of 95%, and its mIoU of 89%. Overfitting did not occur as the accuracy for the testing set did not decrease after a given iteration. On Fig. 11, it is seen the evolution of the prediction throughout the training process. The algorithm is able to predict the biggest unsafe areas, but struggles with the finest details in this setup. However, as the lander continues its descent, details will eventually



(a) **Loss function** (b) **Mean Intersection over Union** (c) **Pixel Accuracy**

FIGURE 10. **Training results for the testing set of ConvDeconv. The loss function decreases and finally plateaus at the same step at the Mean Intersection over Union (mIoU), and the pixel accuracy. As stated in section V, the accuracy of ConvDeconv is of 95%, and its mIoU of 89%. Overfitting did not occur as the accuracy for the testing set did not decrease after a given iteration.**



(a) **1% progress** (b) **5% progress** (c) **100% progress** (d) Target

FIGURE 11. **Progress of the prediction (a), (b), and (c), of the target image (d) for ConvDeconv. It is seen the evolution of the prediction throughout the training process. The algorithm is able to predict the biggest unsafe areas, but struggles with the finest details in this setup.**

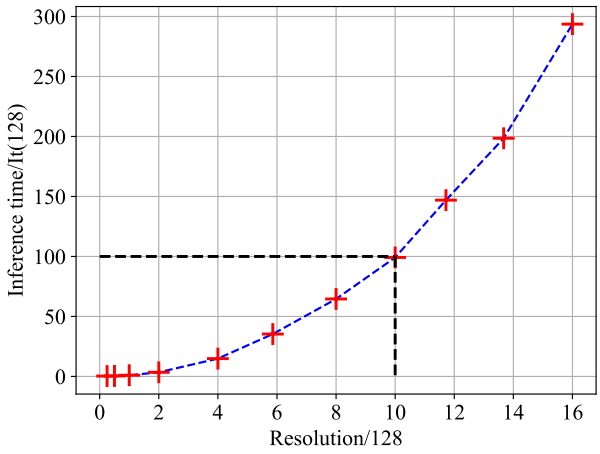


FIGURE 12. **The inference time which follows a quadratic law as a function of the relative resolution, For instance, an increase from 128 × 128 pixels to 1280 × 1280 will lead to an explosion in time by a factor 100. Thus, increasing the resolution of the overall process will be costly and trade-offs would have to be made between accuracy and inference time to satisfy the requirements.**

become large, but this paper wants to stress out that direct conclusions should not be made since the algorithms was only trained at a certain altitude. Nonetheless, as depicted in Fig. 12, the inference time follows a quadratic law with respect to the relative resolution. For instance, an increase from 128×128 pixels to 1280×1280 will lead to an explosion in time by a factor 100. Thus, increasing the resolution of the overall process will be costly and trade-offs would have to be made between accuracy and inference time to satisfy the requirements.

VI. CONCLUSION AND PERSPECTIVES

In this study, after choosing a relevant surrogate of NASA's High-Performance Spaceflight Computer, it was benchmarked several semantic segmentation models and architectures to find the most suitable for the landing problem task, taking into account hardware and mission specifications.

The key findings of this paper are the following:

- Raspberry Pi 4B is a relevant surrogate model for NASA's High-Performance Spaceflight Computer. It allows to reproduce the physical limits of a 64-bit 8-core architecture with a computational limit of 32-GFLOPS.
- There is no correlations between most of the different metrics. However, shallow architectures seem to have a prominent impact on the performance.
- ConvDeconv is the architecture that achieves the best accuracy (>95%). It also provides the fastest inference time with more than 10 predictions per second. It is finally the one requiring the least computational power and memory usage, thus the chosen solution for the given problem.

Further studies would have to vary the resolution of the input Digital Elevation Maps, but also the slant distances and angles of the lander with respect to the ground, in a perspective of finding the most robust architecture. Another

idea would be to make use of different models in concert depending on the lander's configuration.

ACKNOWLEDGMENT

The authors thank Carolina I. Restrepo from NASA Goddard Space Flight Center, Richard J. Doyle, Charles E. Dunn, Jim D. Butler, and Nikolas Trawny from the Jet Propulsion Laboratory, California Institute of Technology, David K. Rutishauser, John M. Carson, and Ronald R. Sostaric from NASA Johnston Space Center for fruitful discussions and support.

REFERENCES

[1] J. L. Chen, "Apollo 11," in *How to Find Apollo Landing Sites*, Heidelberg, Germany: Springer-Verlag, 2014, pp. 33–49, doi: 10.1007/978-3-319-06456-7.

[2] G. A. Soffen and C. W. Snyder, "The first viking mission to Mars," *Science*, vol. 193, no. 4255, pp. 759–766, Aug. 1976, doi: 10.1126/science.193.4255.759.

[3] H. Masursky and N. L. Crabill, "Search for the viking 2 landing site," *Science*, vol. 194, no. 4260, pp. 62–68, Oct. 1976, doi: 10.1126/science.194.4260.62.

[4] B. Wilcox and T. Nguyen, "Sojourner on Mars and lessons learned for future planetary rovers," SAE Technical Paper, SAE, New York, NY, USA, Tech. Rep. 981695, Jul. 1998, doi: 10.4271/981695.

[5] R. E. Arvidson *et al.*, "Overview of the spirit Mars exploration rover mission to Gusev Crater: Landing site to backstay rock in the Columbia hills," *J. Geophys. Res., Planets*, vol. 111, no. E2, pp. 1–22, Jan. 2006, doi: 10.1029/2005JE002499.

[6] S. W. Squyres *et al.*, "Overview of the opportunity Mars exploration rover mission to meridiani planum: Eagle Crater to purgatory ripple," *J. Geophys. Res., Planets*, vol. 111, no. E12, pp. 1–19, Dec. 2006, doi: 10.1029/2006JE002771.

[7] P. H. Smith, "The Phoenix mission to Mars," in *Proc. IEEE Aerosp. Conf.*, Mar. 2004, p. 342, doi: 10.1109/AERO.2004.1367617.

[8] M. Kaufman, *Mars Landing 2012: Inside the NASA Curiosity Mission*. Washington, DC, USA: National Geographic Society, 2012.

[9] T. Hoffman, "InSight: Mission to Mars," in *Proc. IEEE Aerosp. Conf.*, Mar. 2018, pp. 1–11, doi: 10.1109/AERO.2018.8396723.

[10] NASA. *Entry, Descent, and Landing*. Accessed: Mar. 7, 2022. [Online]. Available: https://mars.nasa.gov/mars2020/timeline/landing/entry-descent-landing/

[11] K. A. Farley *et al.*, "Mars 2020 mission overview," *Space Sci. Rev.*, vol. 216, no. 8, pp. 1–41, Dec. 2020, doi: 10.1007/s11214-020-00762-y.

[12] R. Doyle, R. Some, W. Powell, K. Avery, G. Mounce, M. Lowry, M. Johnson, L. Bergman, W. Whitaker, and M. Goforth, "Next generation space processor (NGSP) high performance spaceflight computing (HPSC) next steps at NASA and AFRL," JPL Technical Report Server, Jet Propuls. Lab., Nat. Aeronaut. Space Admin., Pasadena, CA, USA, Tech. Rep. 13-2154, 2013. [Online]. Available: https://trs.jpl.nasa.gov/handle/2014/44372?show=full

[13] B. G. Buchanan, "A (very) brief history of artificial intelligence," *AI Mag.*, vol. 26, no. 4, p. 53, Dec. 2005, doi: 10.1609/aimag.v26i4.1848.

[14] Y. Guo, M. Hawkins, and B. Wie, "Waypoint-optimized zero-effort-miss/zero-effort-velocity feedback guidance for Mars landing," *J. Guid., Control, Dyn.*, vol. 36, no. 3, pp. 799–809, May 2013, doi: 10.2514/1.58098.

[15] B. E. Cohanim and B. K. Collins, "Landing point designation algorithm for lunar landing," *J. Spacecraft Rockets*, vol. 46, no. 4, pp. 858–864, Jul. 2009, doi: 10.2514/1.42002.

[16] T. Yang, P. Li, H. Zhang, J. Li, and Z. Li, "Monocular vision SLAM-based UAV autonomous landing in emergencies and unknown environments," *Electronics*, vol. 7, no. 5, p. 73, May 2018, doi: 10.3390/electronics7050073.

[17] L. Matthies, A. Huertas, Y. Cheng, and A. Johnson, "Stereo vision and shadow analysis for landing hazard detection," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 2735–2742, doi: 10.1109/ROBOT.2008.4543625.

[18] W. Kong, T. Hu, and J. Zhang, "Real-time stereo-vision localization system for safe landing of unmanned aerial vehicles," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2017, pp. 2366–2371, doi: 10.1109/ROBIO.2017.8324773.

[19] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi, "A single-camera feature-based vision system for helicopter autonomous landing," in *Proc. Int. Conf. Adv. Robot.*, Jun. 2009, pp. 1–6.

[20] T. Cui, C. Guo, Y. Liu, and Z. Tian, "Precise landing control of UAV based on binocular visual SLAM," in *Proc. 4th Int. Conf. Intell. Auto. Syst. (ICoIAS)*, May 2021, pp. 312–317, doi: 10.1109/ICoIAS53694.2021.00062.

[21] S. Temel, N. Unaldi, and F. Ince, "Novel terrain relative lunar positioning system using lunar digital elevation maps," in *Proc. 4th Int. Conf. Recent Adv. Space Technol.*, Jun. 2009, pp. 597–602, doi: 10.1109/RAST.2009.5158261.

[22] M. Niwakawa, T. Onda, and N. Fujiwara, "Vision-based handling system using model-based vision and stereo ranging," in *Proc. 2nd Int. Conf. Knowl.-Based Intell. Electron. Syst.*, 1998, pp. 199–204, doi: 10.1109/KES.1998.725911.

[23] H. Bi, W. Zheng, J. Zeng, and X. Fan, "Modeling the topography of fault zone based on structure from motion photogrammetry," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2017, pp. 6251–6254, doi: 10.1109/IGARSS.2017.8128437.

[24] S. Bosch, S. Lacroix, and F. Caballero, "Autonomous detection of safe landing areas for an UAV from monocular images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 5522–5527, doi: 10.1109/IROS.2006.282188.

[25] Y. Xu and F. Duan, "Color space transformation and object oriented based information extraction of aerial images," in *Proc. 21st Int. Conf. Geoinformatics*, Jun. 2013, pp. 1–4, doi: 10.1109/Geoinformatics.2013.6626201.

[26] H.-W. Cheng, T.-L. Chen, and C.-H. Tien, "Motion estimation by hybrid optical flow technology for UAV landing in an unvisited area," *Sensors*, vol. 19, no. 6, p. 1380, Mar. 2019, doi: 10.3390/s19061380.

[27] D. Feng, C. Haase-Schutz, L. Rosenbaum, H. Hertlein, C. Glaser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, Mar. 2021, doi: 10.1109/TITS.2020.2972974.

[28] R. Miyamoto, M. Adachi, H. Ishida, T. Watanabe, K. Matsutani, H. Komatsuzaki, S. Sakata, R. Yokota, and S. Kobayashi, "Visual navigation based on semantic segmentation using only a monocular camera as an external sensor," *J. Robot. Mechatronics*, vol. 32, no. 6, pp. 1137–1153, Dec. 2020, doi: 10.20965/jrm.2020.p1137.

[29] J.-Y. Sun, S.-W. Jung, and S.-J. Ko, "Lightweight prediction and boundary attention-based semantic segmentation for road scene understanding," *IEEE Access*, vol. 8, pp. 108449–108460, 2020, doi: 10.1109/ACCESS.2020.3001679.

[30] C. Symeonidis, E. Kakaletsis, I. Mademlis, N. Nikolaidis, A. Tefas, and I. Pitas, "Vision-based UAV safe landing exploiting lightweight deep neural networks," in *Proc. 4th Int. Conf. Image Graph. Process.*, Jan. 2021, pp. 13–19, doi: 10.1145/3447587.3447590.

[31] R. Moghe and R. Zanetti, "A deep learning approach to hazard detection for autonomous lunar landing," *J. Astron. Sci.*, vol. 67, no. 4, pp. 1811–1830, Oct. 2020, doi: 10.1007/s40295-020-00239-8.

[32] R. Moghe and R. Zanetti, "On-line hazard detection algorithm for precision lunar landing using semantic segmentation," in *Proc. AIAA Scitech*, 2020, p. 462, doi: 10.2514/6.2020-0462.

[33] C. D. Epp and T. B. Smith, "Autonomous precision landing and hazard detection and avoidance technology (ALHAT)," in *Proc. IEEE Aerosp. Conf.*, Mar. 2007, pp. 1–7, doi: 10.1109/AERO.2007.352724.

[34] T. Ivanov, A. Huertas, and J. M. Carson, "Probabilistic hazard detection for autonomous safe landing," in *Proc. AIAA Guid., Navigat., Control (GNC) Conf.*, Aug. 2013, p. 5019, doi: 10.2514/6.2013-5019.

[35] K. Tomita, K. Skinner, K. Iiyama, B. Jagatia, T. Nakagawa, and K. Ho, "Hazard detection algorithm for planetary landing using semantic segmentation," in *Proc. ASCEND*, Nov. 2020, p. 4150, doi: 10.2514/6.2020-4150.

[36] K. Tomita, K. A. Skinner, and K. Ho, "Bayesian deep learning for segmentation for autonomous safe planetary landing," 2021, *arXiv:2102.10545*.

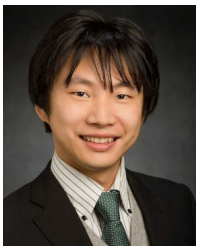
[37] Raspberry Pi. *Raspberry Pi 4 Tech Specs*. Raspberrypi.com. Accessed: Mar. 7, 2022. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[38] K. He and S. Jian, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2015, pp. 5353–5360.

[39] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Jan. 2017, doi: 10.1109/TPAMI.2016.2644615.
[40] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017, doi: 10.1109/TPAMI.2016.2572683.
[41] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," *Computer Vision* (Lecture Notes in Computer Science), vol. 11207. Cham, Switzerland: Springer, doi: 10.1007/978-3-030-01219-9_25.
[42] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput.-Assist. Intervent.* Cham, Switzerland: Springer, Oct. 2015, pp. 234–241, doi: 10.1007/978-3-319-24574-4_28.
[43] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "TransUNet: Transformers make strong encoders for medical image segmentation," 2021, *arXiv:2102.04306*.
[44] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.
[45] P. Garg and S. Jain. *Convolution-Deconvolution-Network*. Accessed: Mar. 7, 2022. [Online]. Available: https://github.com/pgtgrly/Convolution-Deconvolution-Network-Pytorch
[46] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," Jun. 2017, *arXiv:1706.05587*.

THOMAS CLAUDET is currently pursuing the dual master's degree in engineering (control of dynamical systems) with the IMT Atlantique, France, and in mechanical engineering with the Georgia Institute of Technology, writing his master's thesis on safe planetary landing using semantic segmentation. He has been working with the Jet Propulsion Laboratory as a Visiting Student Researcher on the stabilization of planetary-explorer balloons, deep space network scheduling, spacecraft swarm trajectory optimization, and autonomous robotics with team CoSTAR (JPL, Caltech, and MIT) for the DARPA subterranean challenge finals. He is the Founder, a Project Manager, and a Lead Scientist of his CubeSat club.

KENTO TOMITA is currently pursuing the Ph.D. degree with the Georgia Institute of Technology advised by Koki Ho. His specializations are space systems modeling and optimization, and decision making under uncertainty. He is investigating online hazard detection problems for planetary landing missions. He was working as an ADCS Engineer for EQUULEUS, a 6U piggyback satellite of NASA's Artemis-1 Mission, at The University of Tokyo. His research interests include space systems applications under the intersection between artificial intelligence, combinatorial optimization, and probabilistic optimization.



KOKI HO is currently the Director of the Space Systems Optimization Group and an Assistant Professor with the Daniel Guggenheim School of Aerospace Engineering, Georgia Tech. His research interests include space logistics systems design for applications including human space exploration campaigns, on-orbit servicing assembly and manufacturing (OSAM), and large-scale satellite constellations. Motivated by the growing complexity of space missions, he has pioneered a new research direction of logistics-based space systems modeling, substantially improving the efficiency and rigor of the space mission formulation process. His work has been sponsored by NASA, NSF, DARPA, USSF, and industry (including ULA, Maxar Technologies, and so on). He was a recipient of the NSF CAREER Award, the NASA Early Career Faculty Award, and the DARPA Young Faculty Award. He also serves as the Chair for the AIAA Space Logistics Technical Committee.