Path Planning Using a One-Shot Sampling Skeleton Map

Gabriel O. Flores-Aquino, Octavio Gutierrez-Frias, and Juan Irving Vasquez

***Abstract*—Path planning algorithms fundamentally aim to compute collision-free paths, with many works focusing on finding the optimal distance path. However, for several applications, a more suitable approach is to balance response time, path safety, and path length. In this context, a skeleton map is a useful tool in graph-based schemes, as it provides an intrinsic representation of the free workspace. However, standard skeletonization algorithms are computationally expensive, as they are primarily oriented towards image processing tasks. We propose an efficient path-planning methodology that finds safe paths within an acceptable processing time. This methodology leverages a Deep Denoising Autoencoder (DDAE) based on the U-Net architecture to compute a skeletonized version of the navigation map, which we refer to as SkelUnet. The SkelUnet network facilitates exploration of the entire workspace through one-shot sampling (OSS), as opposed to the iterative or probabilistic sampling used by previous algorithms. SkelUnet is trained and tested on a dataset consisting of 12,500 two-dimensional dungeon maps. The motion planning methodology is evaluated in a simulation environment with an Unmanned Aerial Vehicle (UAV) in 250 previously unseen maps and assessed using several navigation metrics to quantify the navigability of the computed paths. The results demonstrate that using SkelUnet to construct the roadmap offers significant advantages, such as connecting all regions of free workspace, providing safer paths, and reducing processing time. These characteristics make this method particularly suitable for mobile robots in structured environments.**

Link to graphical and video abstracts, and to code:
https://latamt.ieeer9.org/index.php/transactions/article/view/10253

***Index Terms*—Motion planning, map-based navigation, mobile robots, denoising autoencoder, deep learning.**

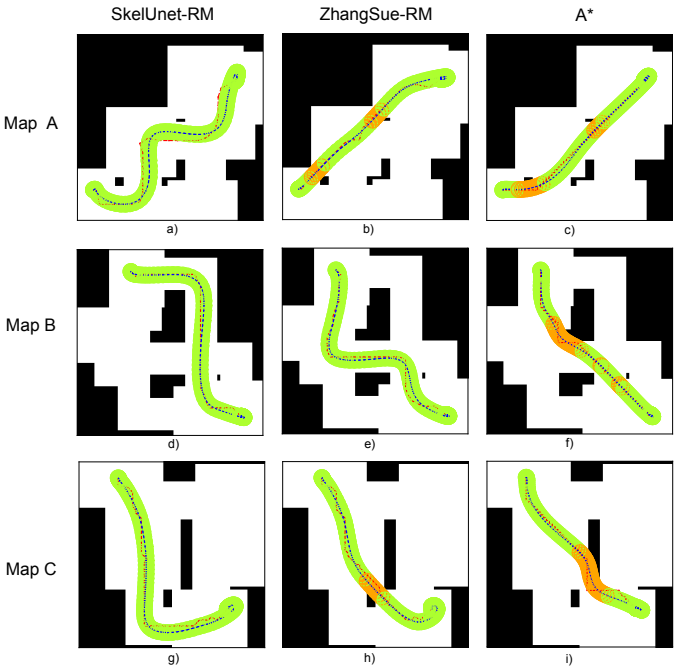


Fig. 1. Comparative path planning results across three indoor maps. Collision robot footprints are shown in green, while configuration within a collision radius are highlighted in orange. The planned path is indicated by a red line, and the executed trajectory is shown in blue. The first column is the trajectory using the skeleton obtained by SkelUnet the second row is the trajectory using the skeleton obtained by Zhang-Sue method and the third column is the trajectory computed with A* algorithm.

I. Introduction

MOBILE robots designed for autonomous operation in structured environment face one of the field's persistent challenges: autonomous navigation [1] [2]. A key component of autonomous navigation is motion planning, which aims to find a collision-free trajectory between an initial configuration, x_{init}, and a goal configuration, x_{goal}. In this context, a robot is defined as a rigid object $\mathcal{A}$, moving in a Euclidean space $\mathcal{W} \in \mathcal{R}^N$, called workspace. The configuration, $\mathcal{X}$, of any arbitrary object is the specification of the position of every point in this object relative to a workspace frame, $\mathcal{F}_\mathcal{W}$. A configuration x of $\mathcal{A}$ is a specification of the position and orientation of the cartesian robot frame, $\mathcal{F}_\mathcal{A}$, relative to the global frame. The purpose of mapping the robot to the configuration space is to represent its geometry as single a point. In consequence, the path planning problem is reduced to find a sequence of robot configurations, $\mathcal{A}(x)$, that establish a collision-free path. A free path between two free configuration (x_{init}, x_{goal}) is a continuous map $\mathcal{S} : [0, 1] \rightarrow \mathcal{X}_{free}$, where $\mathcal{X}_{free} = \{x \in \mathcal{X} / \mathcal{A}(x) \cap (\bigcup_{i=1}^{x} \mathcal{B}_i)\}$ and $\mathcal{B}$ is every obstacle in the workspace. Similarly, $\mathcal{X}$-obstacles can be described as $\mathcal{XB} = \{x \in \mathcal{X} / \mathcal{A}(x) \cap \mathcal{B} \neq \emptyset\}$.

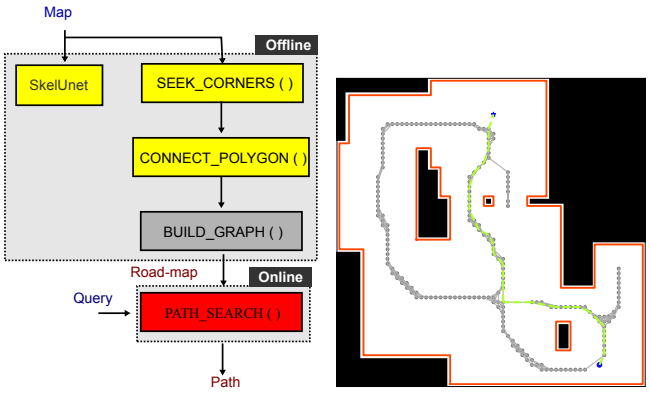
In many path planning problems, a map of the environment is available and the best strategy for finding a path is using the map information. The motion planning paradigm that uses maps is called map-based planning. A robot-friendly represen-

The associate editor coordinating the review of this manuscript and approving it for publication was Javier Moreno-Valenzuela (*Corresponding author: Juan Irving Vasquez*).

G. O. Flores-Aquino is with the Centro de Investigación en Matematicas, A.C. Guanajuato, Gto 36023, Mexico (e-mail: gabriel.flores@cimat.mx).

O. Gutierrez-Frias is with the Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas, Instituto Politécnico Nacional, Mexico City 07340, Mexico (e-mail: ogutierrezf@ipn.mx).

Juan Irving Vasquez is with the Centro de Innovación y Desarrollo Tecnológico en Cómputo, Instituto Politécnico Nacional, Mexico City 07700, Mexico (e-mail: jvasquezg@ipn.mx).



(a) Workflow (b) Example

Fig. 2. SkelUnet-OSS planning scheme. (a) Methodology workflow. The routine `SEEK_CORNERS` found the corners inside the map, and the routine `CONNECT_POLYGON` connects these corners in the correct order. In `BUILD_GRAPH` we connect the vertices for the nearest neighborhoods and determine if an obstacle exist. The routine `PATH_SEARCH` is responsible for finding a path within the graph. (b) An example of application on a map.

tation of a two-dimensional map is an occupancy grid map. In an occupancy grid map, we discretize the configuration space $\mathcal{X}$ into a rectangular grid $\mathcal{GX}$ as configuration in $\mathcal{GX}$ is labeled 0 if it lies in free space and 1 otherwise [3]. If we have a map representation, the initial approach from a path planning perspective is to perform a cell search using a forward search algorithm, such as breadth-first, depth-first, Dijkstra's algorithm, or A* [3]. These algorithms are systematic methods that visit each reachable configuration in a finite amount of time and return a solution if one exists. This type of problem is known as discrete planning, and it faces the challenge of being computationally expensive because each planning exercise requires a new search, and the algorithms need to visit all reachable configurations. To address these issues, we can use a paradigm known as roadmap methods [3], where the map is utilized to build a representation of free configuration space. The roadmap is a type of topological graph that should provide an accurate and functional representation for navigation in free configuration space [4]. It must contain selected information about the workspace and, additionally, serve as a helpful and reusable data structure for quickly finding collision-free paths. There are many methods for building roadmaps, including visibility maps, deformation retracts, piecewise retracts, and silhouettes [3], to name a few. A roadmap is composed of vertices and edges. A vertex ($\mathcal{V}$) represents a reachable configuration while an edge ($\mathcal{E}$) is the connection between a pair of vertices. The roadmap has the aim of spreading out over the whole free configuration space $\mathcal{X}_{free}$ to be able to connect the initial vertex to a departure vertex and the goal vertex with an arrival vertex. Therefore, an adequate roadmap should capture the connectivity of the free configuration space in the form of a graph of one-dimensional curves. Consequently, path planning is simplified to connecting the start and goal configurations to the roadmap and searching for a path in the graph $\mathcal{G}$ [3].

We propose a method that builds a roadmap, when the configuration space is two-dimensional with polygonal $\mathcal{X}$-obstacles, using a classic task in computer vision known as skeletonization [5]. From the perspective of digital image processing, skeletonization is a morphological operation [6]. The skeleton is defined as the locus of points where wave-fronts propagating from a shape's contour meet and undergo cancellation. It serves as a central line of symmetry consisting of points equidistant from the pattern, providing a unique and invertible description, [7].Some characteristics that the skeleton must have are:

1) End points must not be removed.
2) The connection must not be broken.
3) Excessive erosion must not be caused.

Several methods exist for generating a skeleton; some examples include Voronoi diagrams, distance transformation [3], thinning [8], morphological operations [9], and the Zhang-Suen method [10]. Calculating the skeleton is an expensive and complicated task in terms of processing time, which is why deep learning methods have been proposed in recent years [5], [11]. However, the effectiveness of these methods is closely tied to the configuration of the original image. Therefore, the algorithms are generally suitable only for specific datasets, typically composed of objects such as animals or human shapes. Their application in path planning schemes remains an under-explored area.

Our approach introduces a tiny neural network architecture (SkelUnet) capable of learning a free workspace representation from navigation maps and suitable for mobile robots. SkelUnet has the advantage of removes the lines that connect the rooms in crosswise, Fig. 5. This characteristic improves the performance in path planning schemes. The path planning methodology SkelUnet-OSS is a forthright method suitable for being implemented in mobile robots, whose tasks are deployed in structured environments because its processing time is short and the graph is far from the obstacles, increasing the robot's safety, see Fig. 1. In addition, SkelUnet-OSS methodology is capable of performing the whole sampling process in one computing cycle and therefore reduces the computing cost. Additionally, to evaluate the paths obtained in terms of navigability we present the results of implementing the benchmarking metrics described in [12].

The paper is organized as follows. Section II describes the recent advances in the field. In section III, we detail the path planning approach, and in section IV, we present the results. In section IV-D, we compare our method against the medial axis transform and in section V we talk about the conclusions and future work.

II. Related Work

Some works without a learning approach. In [13], the authors propose an algorithm that builds a roadmap from a skeleton map and subsequently replaces the intersection edges with connected polygons. This change improves the transitions in crossing areas. The work also stands out the fact that in many cases the optimal distance path is not always the safest, so using some type of skeleton as a framework for the roadmap is a feasible option. Another recent work is presented

in [14] where it is proposed to use the skeleton obtained by the *Wavefront* method [3] as a growth guide for the RRT algorithm [15]. From a machine learning perspective, some works implement neural networks to compute a roadmap, for example [16] where is used a single layer Kohonen neural network or [17] where a deep convolutional neural network (CNN) along with A* algorithm is used to compute an optimal path. These works are focused in reduce processing time and they are not interested in the quality or navigability of the paths. Other works focus on ensuring that the skeleton represents navigable areas, such as [18] and [8], where they modify the map before obtaining the skeleton, widen the obstacles considering the geometry of the robot. However this pre-processing may be unnecessary if we generate critical nodes to build the roadmap, a critical node is a key point that divide a complex path into sub-paths more manageable. This is the case of [19], where CNN ResNet50 is used to compute a path. Works focused on UAVs are, for example, [20], [21], and [22].

Although the present work utilizes the previous idea of skeletonization for roadmap generation. It addresses specific limitations regarding computational efficiency and the topological quality of the resulting graphs. Unlike traditional methods that require geometric pre-processing to ensure safety [8], [18], or deep learning approaches that rely on computationally heavy backbones like ResNet50 [19] often ignoring path quality in favor of speed, our proposal balances lightweight inference with high-quality roadmap construction.

III. Path Planning with OSS Maps

Our path planning methodology relies on the concept of graph-based planning, where a set of vertices in the workspace are connected to create a graph, subsequently, the graph is used to find a feasible path between the start and goal configurations. In Sampling-based Motion Planning (SBMP) algorithms, the vertices are the result of sampling either the workspace or the configuration space. In our case, the sampling is performed via a one-shot sampling using the SkelUnet network (see sub-section III-A). The path planning methodology (SkelUnet-OSS) converts the generated vertices to a graph and resolves navigation queries. In the following sections, we present SkelUnet-OSS, comprising a construction stage and a query stage, detailed in subsections III-B and III-C respectively. The general procedure is summarized in Fig. 2 and described in Algorithm 1.

A. SkelUnet

U-Net architecture was originally designed for image segmentation and is formed by two neural networks in a U-shaped design: an encoder and a decoder. The encoder consist of convolutional and max pooling layers that progressively extract features i.e. encode the inputs to a latent representation; the decoder restores the original resolution using transposed convolutions [23]. Based on the U-Net architecture, we propose SkelUnet as a way of sampling the workspace. Unlike the original U-Net, in our approach we consider the input image as a noisy version, and the target like a denoising output. SkelUnet, illustrated in Fig. 3, is designed to be trained in a supervised fashion, [24], where the input is a navigation map and the target is its skeletonized version. In this case, the target is obtained using the Zhang-Suen's method [10]. Among the geometric methods, notable examples focus on either processing speed or solution accuracy, such as Voronoi-based skeletonization [4], thinning, and the Medial Axis Transform [3]. Formally, SkelUnet is a convolutional deep denoising autoencoder (DDAE) composed of two parametric functions. The first one:

$$h = f_\theta(x) = S_f(\text{Conv2d}(\theta, x)) \tag{1}$$

for encoding, and the second one:

$$g_\phi(h) = S_g(\text{UpConv2D}(\phi, h)) \tag{2}$$

for decoding, where θ is the set of the convolution parameters. ϕ is the set of transposed convolution parameters, and S_f with S_g are activation functions [25]. The rational for using an autoencoder-based architecture is motivated by the manifold hypothesis. This hypothesis posits that the probability distribution of the data of interest is highly concentrated in a reasonably small number of connected regions (manifolds), rather than being uniformly distributed over the entire space [26]. Learning such a manifold is useful for sampling regions of interest, such as cluttered scenes or narrow passages, from a 2D workspace. In this paper, We use the original map representation, the workspace $\mathcal{W} \subset \mathbb{R}^2$, to feed the trained SkelUnet, obtaining a skeletonized version of the free workspace, $\widetilde{\mathcal{W}}_{free} \subset \mathbb{R}^2$. Note that for robots with higher degrees of freedom their free configuration space $\mathcal{X}_{free}$ may differ. Such a problem is out of the scope of this paper and it will be addressed in future work.

B. Building Stage

We describe building stage in Algorithm 1. First, in lines 1 and 2 we build a geometric representation of the map. In line 1, the routine `SEEK_CORNERS` finds all corners of the map and return their pixels coordinates in an array $\mathcal{C}$. In line 2, the routine `CONNECT_POLYGON` connects each element of $\mathcal{C}$ and returns n polygons in the set $\mathcal{XB}$. In line 3, we build a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ by calling the function `BUILD_GRAPH`. We need $\mathcal{XB}$ to verify the free transit between vertices. In line 6, we get a representation of the free workspace $\widetilde{\mathcal{W}}_{free}$. The propose of $\widetilde{\mathcal{W}}_{free}$ is get the set of vertices $\mathcal{V} \in \mathcal{W}_{free}$. In lines 7 to 14, we connect each vertex with its k nearest neighbors. The notation $\overline{x_{near}, x}$ refers to the edge $\mathcal{E}$ between this pair of vertices. Finally, we return the data structure $\mathcal{G}$.

C. Query Stage

Query stage is described in the lines 18 to 23. In line 18, we make a query $\mathcal{Q}$ assigning $x_I \leftarrow q_I$ and $x_G \leftarrow q_G$. In lines 19 to 21, we connect the edge from initial configuration x_I with departure vertex x_s and the goal configuration x_G with the arrival vertex x_f. Additionally, to add the current query to the roadmap, we need to verify that it has collision-free edges.

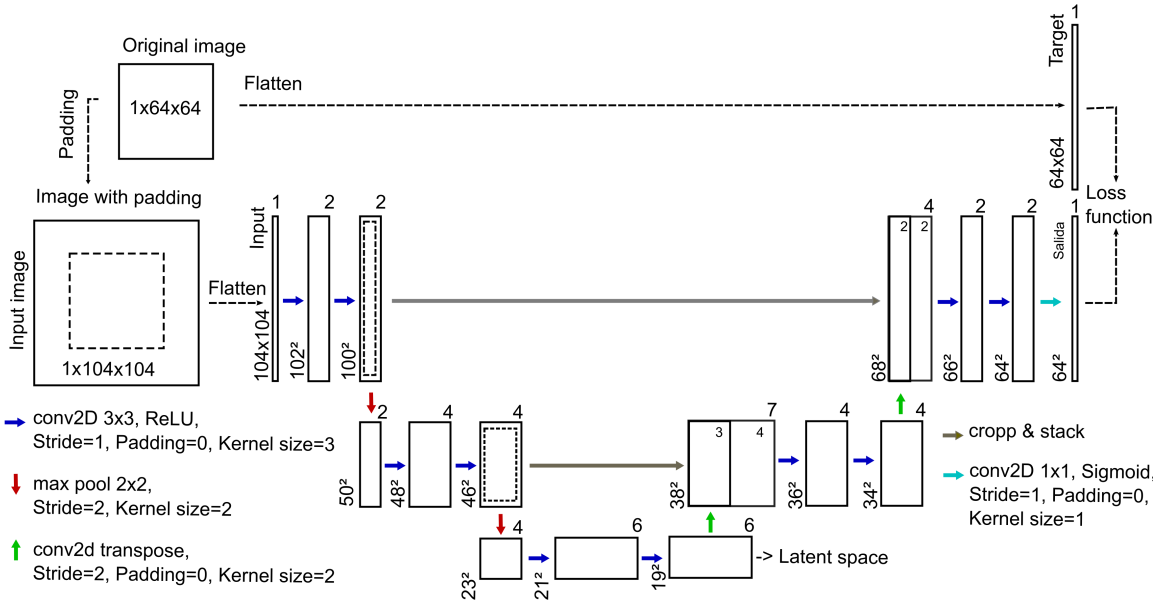


Fig. 3. SkelUnet architecture. The scheme shows the different operations performed by each layer and how it affects the size and depth of the image. The arithmetic of these operations can be consulted in [27].

Finally, we search for a feasible path $\mathcal{S}$ using the graph $\mathcal{G}$ as a roadmap, line 22. The algorithm returns the sequence of vertices and edges that resolves the current query. Note that the method works as a multi-query planner.

IV. Results

To validate the proposed path planning approach, the SkelUnet architecture was implemented, trained, and integrated into a motion planner. In the following sections, we describe the dataset used, the training of SkelUnet, the performance of the whole motion planning method in quadrotor navigation tasks, and provide a comparison against a previous skeletonization approach.

A. Dataset

For training and testing SkelUnet, we utilize a dataset comprising 12,500 maps. We split the dataset into 80.0% for training and 20.0% for testing. Each map is a binary image of 64x64 pixels. The maps consist of hallways and rooms similar to classic dungeon maps, serving as a good representation of residential and office buildings. A complete review of the map generation process and dataset is available in [28]. For the training stage, we use the Zhang-Suen's method [10] to generate the target images.

B. SkelUnet Training

In order to find a set of suitable parameters for the model, we conducted a grid-search across sixteen different configurations. We evaluated three different loss functions, three learning rated, and two optimization algorithms. The loss functions tested were Mean Squared Error (MSE), defined as:

$$l_n = (x_n - y_n)^2, \tag{3}$$

Binary Cross Entropy (BCE), defined as:

$$l_n = -w_n[y_n \cdot \log(x_n) + (1 - y_n) \cdot \log(1 - x_n)], \tag{4}$$

and Weighted Sum (WS), defined as:

$$l_n = \frac{(1 - y_n) \cdot (y_n - x_n)^2}{c_1} + \frac{y_n \cdot (y_n - x_n)^2}{1 - c_1}, \tag{5}$$

where $c_1 = 0.8$. For all three functions, we compute the mean value to obtain the the final loss. The learning rates tested were $1e^{-3}$, $1e^{-4}$, and $1e^{-5}$; similarly the optimization algorithms tested were Adam and Stochastic Gradient Descent (SGD). For weight initialization, we used the He initialization.

After passing the map through the autoencoder, we apply a threshold value between 0 and 1. This threshold is indicated with a subscript in Tables I and II. SkelUnet was implemented in PyTorch. Training was conducted on a workstation equipped with an Intel i7-10700 CPU, an NVIDIA RTX 2070 super GPU, and 16GB of RAM. Training for 1000 epochs took about 2 hours and 50 minutes. To reduce the total training time, we selected the best three models after 1000 epochs; then we extended the training for 9000 epochs. The performance results and parameters for the best models are shown in Tables I and II. The best performance is obtained with the parameters of the first row in the mentioned table and the training and testing curves for the best model are displayed in Fig. 4.

1) Results of SkelUnet Architecture: To evaluate the performance of the model, we used the F1-score (see Table II). This metric is particularly useful for binary data, where accuracy is often a biased indicator of performance due to class imbalance. For all the metrics, we consider skeleton pixels as the positive class. Thus, the precision (Prc) indicates, what proportion of the decoded skeleton is actually part of the target skeleton. The recall (Rcl) is the fraction of the skeleton that was detected. In

Algorithm 1: SkelUnet-OSS

Input : $\mathcal{W}$: Workspace;
Φ: Trained SkelUnet;
$Q = \{q_I, q_G\}$: Query;
Output : $\mathcal{S}$: Path
Parameters: k: Number of neighborhoods;

/* **Building** stage: */
$\mathcal{C} \leftarrow$ SEEK_CORNERS$(\mathcal{W})$;
$\mathcal{XB} \leftarrow$ CONECT_POLYGON$(\mathcal{W}, \mathcal{C})$;
$\mathcal{G} \leftarrow$ BUILD_GRAPH$(\Phi, \mathcal{W}, \mathcal{XB}, k)$;

BUILD_GRAPH()
  $\mathcal{G} \leftarrow \emptyset$;
  $\widetilde{\mathcal{W}}_{free} \leftarrow \Phi(\mathcal{W})$; $\mathcal{V} \leftarrow \widetilde{\mathcal{W}}_{free} \in \mathcal{W}_{free}$;
  for $x \in \mathcal{V}$ **do**
    $\mathcal{X}_{nearest} \leftarrow$ NEAREST_NEIGHBOR$(x, \mathcal{V}, k)$;
    for $x_{near} \in \mathcal{X}_{nearest}$ **do**
      if $\{\overline{x_{near}, x} \cup \mathcal{XB}\} = \emptyset$ **then**
        $\mathcal{E} \leftarrow \overline{x_{near}, x}$
      end
    end
  end
  $\mathcal{G} \leftarrow \{\mathcal{V}, \mathcal{E}\}$;
  return $\mathcal{G}$;
end

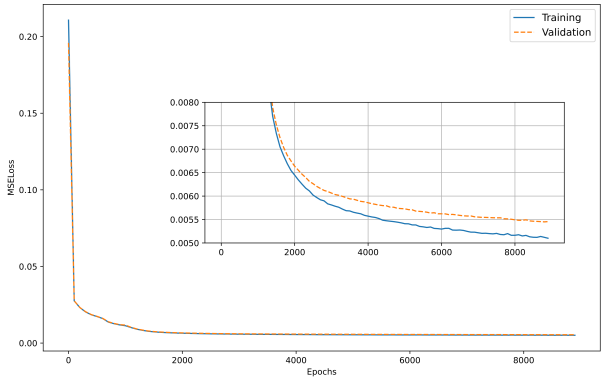
/* **Query** stage: */
$x_I, x_G \leftarrow Q\{q_I, q_G\}$;
if $(\{\overline{x_I, x_s} \cup \mathcal{XB}\} = \emptyset)$ and $\{\overline{x_G, x_f} \cup \mathcal{XB}\} = \emptyset)$ **then**
  $\mathcal{G} \leftarrow \{x_I, x_G\} \cup \mathcal{G}$
end
$\mathcal{S} \leftarrow$ PATH_SEARCH$(\mathcal{G}, Q)$;
Return $\mathcal{S}$

TABLE I
GRID SEARCH RESULTS AFTER 1000 EPOCHS

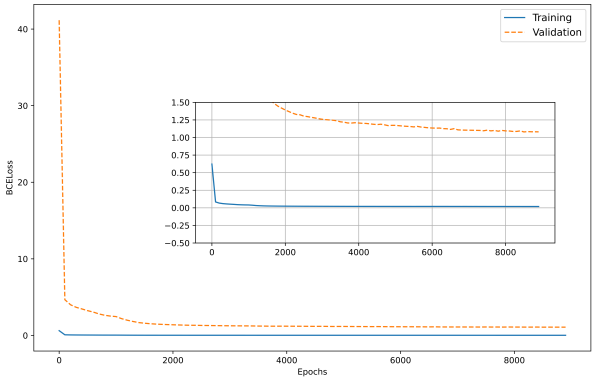
No.	LF	LR	Op	Tr	Tst	$F1_{0.99}$
1	MSE	$1e^{-3}$	Adam	0.0118	0.0113	**0.537**
2	WS	$1e^{-3}$	Adam	0.0039	0.0075	0.521
3	BCE	$1e^{-3}$	Adam	0.0418	2.4407	0.487

other words, a low recall value means that SkelUnet tends to decode overly sparse skeletons (missing connections), whereas a low precision value implies that SkelUnet tends to decode a noisy or branched version of the skeleton (containing spurious artifacts). Values close to one indicate the highest performance.

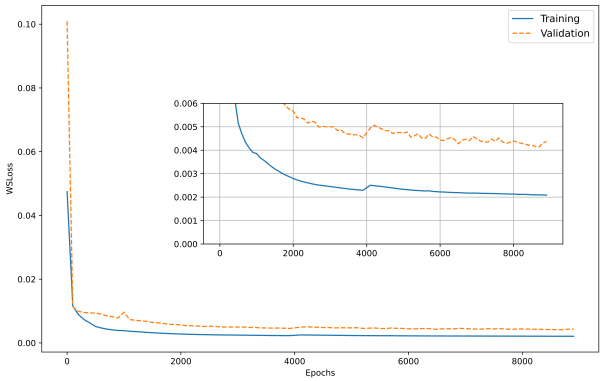
For model 1, we obtained the best score with $F1_{0.5} = 0.8792$, as shown in Table II. These are promising results if we take as a baseline the work of [5], where the authors present a more extensive autoencoder model to skeletonize images containing different animal shapes and others forms; they report a best performance of $F1 = 0.6668$. In Fig. 5, we show one of the advantages of using a learning-based method to compute the skeleton. We observe that SkelUnet removes unwanted connections that occur in diagonals. Since we want to use the skeleton map as a graph, such kind of



(a) MSE Loss



(b) BCE Loss



(c) WS Loss

Fig. 4. Graphics for the training and test data-set for model the best three models with different loss functions. (a) MSE loss, (b) BCE loss, (a) WS loss. In continuous line, the loss in training, and in dotted line, the loss for testing.

connections are not useful because they are generated between not connected rooms. Such a problem can be observed in the second row of the Fig. 5.

C. Robot Experiments

We used 250 maps and the navigation metrics described in [12] to evaluate the paths obtained with SkelUnet-OSS. We applied the SkelUnet-OSS methodology using as input the skeleton derived from SkelUnet and Zhang-Suen. Additionally, we compared our results against a discrete planning method based on the A* algorithm. The benchmark metrics are shown in Table V. Finally, SkelUnet-OSS results were tested in a

TABLE II
RESULTS FOR METRICS FOR VALIDATION DATASET. FOR MODEL 1 WE OBTAINED THE BEST $F1$ SCORE WITH $F1 = 0.8792$ AND THE BEST RECONSTRUCTION LOSS $l = 0.0020$ IN MODEL 3

No.	TP	FP	TN	FN	Prc	Rcl	$F1_{0.5}$
1	94.264	9.176	3978.076	14.484	0.9014	0.8643	**0.8792**
2	91.484	9.308	3977.944	17.264	0.9023	0.8390	0.8656
3	99.132	21.056	3966.196	9.616	0.8203	0.9083	0.8602

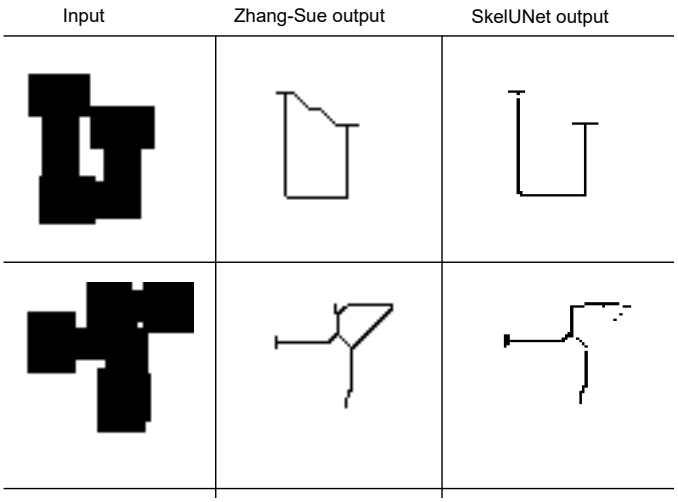


Fig. 5. Comparative skeletonization results using Zhang-Suen's method versus SkelUnet. The first column is the input, the second column shows the output from Zhang-Suen's method, and the third column shows the SkelUnet output.

TABLE III
QUAD-ROTOR PHYSIC PARAMETERS

Parameter	Value
Mass	$0.5[kg]$
Distance from vehicle origin to motors (**L**)	$0.17[m]$
Offset from center of mass on the x-axis (**cx**)	$0.0[m]$
Offset to center of mass on the y-axis (**cy**)	$0.0[m]$
Moments of inertia x-axis (Ixx)	$0.0023[kg \cdot m^2]$
Moments of inertia y-axis (Iyy)	$0.0023[kg \cdot m^2]$
Moments of inertia z-axis (Izz)	$0.0046[kg \cdot m^2]$

TABLE IV
QUAD-ROTOR CONTROLLER GAINS USED IN SIMULATION

Position gains		Velocity gains		Attitude gains	
kpPosXY	1.0	kpVelXY	1.5	kpBank	13.5
kpPosZ	2.5	kpVelZ	7.0	kpYaw	2.5
kiPosZ	2.0			kpPQR	55,55,10

simulation employing a quadrotor robot (see Fig. 1). Fig. 6 shows the environment to simulate drone dynamics. Order to check drone collision, first each path is simulated using drone dynamics, then the followed path and drone bounding box is projected to the 2D map and checked for collision. For the quadrotor, we use the z-axis exclusively for take-off and landing; path tracking is performed while maintaining a fixed altitude because our workspace is two-dimensional. The simulation environment implements a cascade PID controller and a second-order quadrotor dynamic model. The translational dynamics are defined as follows:

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = R(t)\begin{bmatrix}0\\0\\c(t)\end{bmatrix} - \begin{bmatrix}0\\0\\g\end{bmatrix} \implies \begin{cases}\ddot{x} = c\cdot b_x\\ \ddot{y} = c\cdot b_y\\ \ddot{z} = c\cdot b_z - g\end{cases}, \quad (6)$$

where (b_x, b_y, b_z) correspond to the third column of the rotation matrix, and $c(t)$ the collective thrust. On the other hand, the rotational dynamics are defined by:

$$\dot{R}(t) = R(t)\begin{bmatrix}0 & -r(t) & q(t)\\ r(t) & 0 & -p(t)\\ -q(t) & p(t) & 0\end{bmatrix}, \quad (7)$$

where p, q, r are the angular velocities of the body. The physics and controller parameters are presented in table III and IV. More details about the controller are available at [29].

The trajectories obtained in simulation are evaluated using the metrics detailed in Fig. 7. Distance to the Closest Obstacle (DTCO) is the distance to the nearest occupied region. For Average Visibility (AV), eight rays are traced, and the distance to the nearest obstacle is calculated in each direction. The Characteristic Dimension (CD) is similar to Average Visibility but only considers the shortest distance. For Dispersion (Dsp), the number of rays is increased to sixteen, and the maximum range is set to a constant value. If a scan ray intersects an occupied region while the previous ray lies in a free region (or vice versa), a change is counted. For these four metrics, the final result is the average over the path. Tortuosity (Trts) is the ratio between the path length and the Euclidean distance between the start and target points. A complete description of these metrics can be found in [12].

D. Comparison with Medial-Axis Transform (MA)

The medial axis transform (MA) [30], [31] is a computational geometry technique used to extract the skeleton of the free workspace. We applied MA to a selection of maps from our dataset to highlight the advantages of our approach over classical methods. The MA transform is used to build a roadmap from a set of random samples $P \in \mathcal{X}_{free}$, such that each sample is retracted onto the medial axis of the free configuration space. Formally, the MA transform is defined as:

$$MA(P) = x \in P \mid \nexists; y \in P; \text{with}; B_P(x) \subseteq B_P(y) \quad (8)$$

In Equation (8), $B_P(\cdot)$ represents the largest closed disc centered at x (or y) that is contained in P. The results of the MA transform are very similar to the samples obtained from a skeletonization process. Our implementation follows the algorithm described in [30] with one exception: we always

TABLE V
METRIC FOR PATHS OBTAINED FROM SKELETONS USING SKELUNET, ZHANG-SUE'S METHOD AND A*

	SkelUnet				Zhang-Sue				A*			
	Min.	**Max.**	**Mean.**	**Std.**	**Min.**	**Max.**	**Mean.**	**Std.**	**Min.**	**Max.**	**Mean.**	**Std.**
DTCO.	2.71	18.80	8.32	2.49	2.81	19.75	**8.78**	2.80	1.00	22.65	6.06	3.30
AV.	8.64	30.81	18.17	3.91	8.64	31.45	**18.60**	4.08	8.64	31.34	17.54	4.02
Dsp.	0.00	1.89	**0.25**	0.30	0.00	2.22	0.36	0.47	0.00	4.00	1.17	0.78
CD.	5.00	48.00	22.70	7.10	5.00	53.00	**23.63**	7.71	5.00	53.00	22.01	7.42
Trts.	0.09	1.00	0.68	0.17	0.09	1.00	0.67	0.16	0.23	1.00	**0.78**	0.09

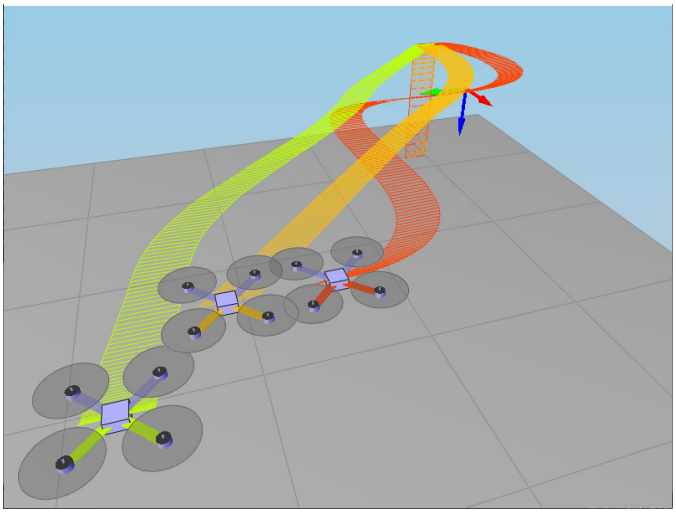


Fig. 6. Example of the environment for simulating drone dynamics. The quadrotor follows the paths obtained from SkelUnet in red, Zhang-Suen's method in green, and A* in yellow. This simulation validates dynamics; collision detection is performed using the dataset maps.

initialize the random samples in the free space to reduce processing time.

Fig. 8 shows how the medial axis, in the context of our dataset, has the disadvantage of retracting samples onto the same specific regions even as we increase the number of samples. Therefore, the resulting roadmap does not cover the entire workspace (see Fig. 9). To connect nodes, we use the k-nearest neighbors algorithm with $k = 6$. Considering these results, we can identify the need for a more accurate method to obtain samples in safe regions while simultaneously covering the entire free workspace.

E. Analysis

From a computer vision perspective, SkelUnet outperforms similar work in skeletonization tasks, demonstrating its effectiveness in processing two-dimensional maps. Based on the experiments, we believe that skeletonized roadmaps have some advantages over Zhang-Suen. Namely, SkelUnet efficiently covers free regions of the workspace without including hard-to-transit areas, showcasing the neural network's efficiency in computing complex path planning tasks.

Analyzing the results from Table V regarding average values, we observe that DTCO has the best performance in paths from Zhang-Suen and SkelUnet, which exhibit very close mean values (a difference of only about 5.2%). For A*, the mean is 6.06, which means that its paths are closer to the obstacles. For the AV index, a small value means the computed path goes through narrow passages, implying greater difficulty for the robot to navigate. The best performance in AV is achieved by Zhang-Suen, followed by SkelUnet with a 2.3% difference.

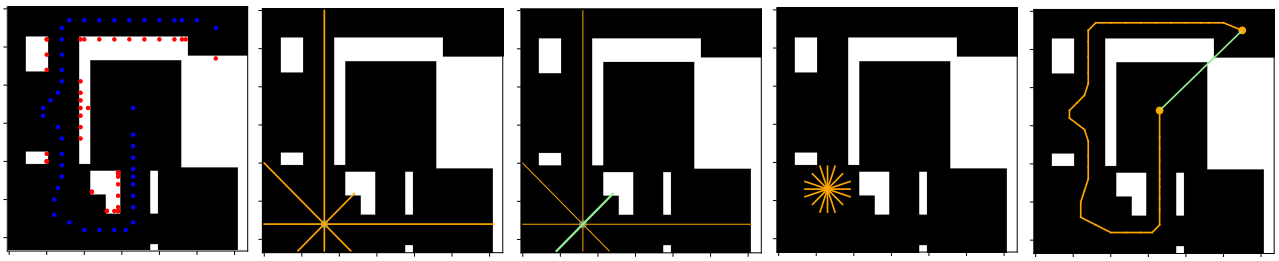
For Dsp, a low value means that the path follows a clear region. SkelUnet has the best performance, with a difference of 44.0% compared to Zhang-Suen and 368.0% lower than A*, indicating a remarkable difference between the methods. The CD metric is very similar across the three methods because it is a more unspecific metric derived from the AV metric. Finally, regarding tortuosity, a value close to one means short paths. A* has the shortest paths, while Zhang-Suen and SkelUnet only differ by 1.47%. The results indicate that SkelUnet and Zhang-Suen generate longer paths than A* but are more feasible for robot navigation. SkelUnet and Zhang-Suen's results are very similar except for Dsp. We can assume that Dsp reflects the diagonal parts of the skeleton removed by SkelUnet. Please note that unlike works such as [17] and [19], we do not explicitly consider the robot's geometry in the planning scheme because the robot is modeled as a point in the configuration space.

1) Scalability and Limitations: Currently, the method is not suitable for three-dimensional maps, such as voxel maps. The main restriction is that the SkelUnet architecture operates on two-dimensional inputs; therefore, for three-dimensional maps a new architecture is required. Nevertheless, the proposed method was analyzed for two-dimensional maps and tested on a robot with a workspace in $\mathbb{R}^3$. For this purpose, we considered the path as a set of waypoints with constant elevation. This approach demonstrates that the method is suitable for mobile robots. The path computed by our method is used within a global planner, where each vertex represents a position for the robot and the transitions are handled by a local planner.

V. CONCLUSIONS

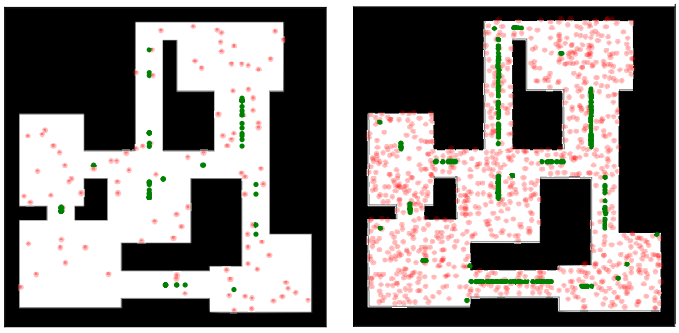
We presented a method for path planning that constructs a safe roadmap to perform the sampling stage in a single cycle using a neural network called SkelUnet.

According to navigation metrics, SkelUnet-OSS generates more traversable paths than the A* algorithm and demonstrates slight improvements in the dispersion metric. This represents a significant advantage, as it indirectly accounts for uncertainties encountered in real-world robotics applications.



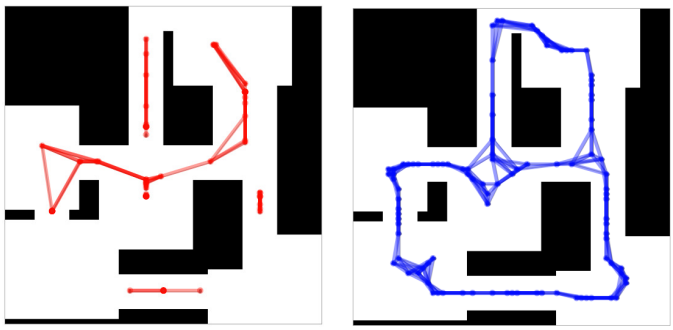
(a) DTCO (b) AV (c) CD (d) Dsp (e) Trts

Fig. 7. From (a) to (e) the five benchmark navigation ground metrics to evaluating the resulting paths. In this Figure the free workspace is highlighted in black.



(a) MA low-sampling (b) MA high-sampling

Fig. 8. Medial-axis transform over an example map with two different rate of samples, (a) low-sampling and (b) high.-sampling. In red random samples in free workspace (x). In green the result of MA transform (MA(x))



(a) MA roadmap (b) SkelUnet roadmap

Fig. 9. Roadmap comparative with 100 samples. (a) a roadmap applying MA, (b) roadmap applying SkelUnet.

Compared to traditional roadmap methods like the Medial Axis Transform, SkelUnet achieves comprehensive workspace coverage with fewer samples, highlighting its superiority in this particular task. Furthermore, although we employ an edge connection module, if the number of neighbors is low, a graph can be built from SkelUnet without requiring a collision detection stage. The underlying idea of this approach is based on the following assumption: Given a path planning problem, we assume the existence of a path in state space that resolves the query. This route can be expressed in the workspace and therefore serve as a guide for the remaining configuration variables.

In future work, we aim to explore the replacement of collision detection with a learning approach. Another aspect to improve is extending the skeleton version to 3D environments and applying it in more complex tasks.

ACKNOWLEDGMENTS

This research was funded by Secretaria de Investigación y Posgrado-IPN grant number 20240705.

REFERENCES

[1] P. I. Corke, W. Jachimczyk, and R. Pillat, *Robotics, vision and control: fundamental algorithms in MATLAB*. Springer, 2011, vol. 73, dOI: https://doi.org/10.1007/978-3-031-07262-8.

[2] T. Huang, K. Fan, and W. Sun, "Density gradient-rrt: An improved rapidly exploring random tree algorithm for uav path planning," *Expert Systems with Applications*, vol. 252, p. 124121, 2024, dOI: https://doi.org/10.1016/j.eswa.2024.124121. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424009874

[3] J.-C. Latombe, *Robot Motion Planning*. New York: Springer, 1991, dOI: https://doi.org/10.1007/978-1-4615-4022-9.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion*. London, England: MIT Press, 2005, dOI: https://doi.org/10.1017/S0263574706212803.

[5] S. Song, H. Bae, and J. Park, "Disco - u-net based autoencoder architecture with dual input streams for skeleton image drawing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 2128–2135, dOI: https://doi.org/10.1109/ICCVW54120.2021.00241.

[6] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022, dOI: https://doi.org/10.1007/978-3-030-34372-9.

[7] W. Wathen-Dunn, "A transformation for extracting new descriptors of shape, in models for the perception of speech and visual form," *MIT Press, Cambridge: MA*, 1967.

[8] P. K. Saha, G. Borgefors, and G. S. di Baja, "Skeletonization and its applications–a review," *Skeletonization*, pp. 3–42, 2017, dOI: https://doi.org/10.1016/B978-0-08-101291-8.00002-X.

[9] P. Maragos and R. Schafer, "Morphological skeleton representation and coding of binary images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1228–1244, 1986, dOI: https://doi.org/10.1109/TASSP.1986.1164959.

[10] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984, dOI: https://doi.org/10.1145/357994.358023.

[11] I. Demir, C. Hahn, K. Leonard, G. Morin, D. Rahbani, A. Panotopoulou, A. Fondevilla, E. Balashova, B. Durix, and A. Kortylewski, "Skelneton 2019: Dataset and challenge on deep learning for geometric shape understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019, dOI: https://doi.org/10.48550/arXiv.1903.09233.

[12] D. Perille, A. Truong, X. Xiao, and P. Stone, "Benchmarking metric ground navigation," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2020, pp. 116–121, dOI: 10.1109/SSRR50563.2020.9292572.

[13] D.-H. Yang and S.-K. Hong, "A roadmap construction algorithm for mobile robot path planning using skeleton maps," *Advanced Robotics*, vol. 21, no. 1-2, pp. 51–63, 2007, dOI: https://doi.org/10.1163/156855307779293724.

[14] Y. Dong, E. Camci, and E. Kayacan, "Faster rrt-based nonholonomic path planning in 2d building environments using skeleton-constrained path biasing," *Journal of Intelligent & Robotic Systems*, vol. 89, pp. 387–401, 2018, dOI: https://doi.org/10.1007/s10846-017-0567-9.
[15] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001, dOI: https://doi.org/10.1177/02783640122067453.
[16] N. Bourbakis, D. Goldman, R. Fematt, I. Vlachavas, and L. Tsoukalas, "Path planning in a 2-d known space using neural networks and skeletonization," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 3. IEEE, 1997, pp. 2001–2005, dOI: https://doi.org/10.1109/ICSMC.1997.635147.
[17] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020, dOI: https://doi.org/10.1109/TASE.2020.2976560.
[18] H. Ryu and Y. Park, "Improved informed rrt* using gridmap skeletonization for mobile robot path planning," *International Journal of Precision Engineering and Manufacturing*, vol. 20, pp. 2033–2039, 2019, dOI: https://doi.org/10.1007/s40684-019-00075-8.
[19] F. Zou, S. Jia, T. Liu, C. Wang, and Y. Niu, "Generating critical nodes for sub-path planning with a deep neural network based on resnet50," in *Proceedings of 2022 International Conference on Autonomous Unmanned Systems (ICAUS 2022)*, W. Fu, M. Gu, and Y. Niu, Eds. Singapore: Springer Nature Singapore, 2023, pp. 811–822, https://doi.org/10.1007/978-981-99-0479-2_74.
[20] M. Ramana, S. A. Varma, and M. Kothari, "Motion planning for a fixed-wing uav in urban environments," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 419–424, 2016, 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2016; DOI: https://doi.org/10.1016/j.ifacol.2016.03.090. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896316300908
[21] V. Zinage and S. Ghosh, "An efficient motion planning algorithm for uavs in obstacle-cluttered environment," in *2019 American Control Conference (ACC)*, 2019, pp. 2271–2276, dOI: https://doi.org/10.23919/ACC.2019.8814609.
[22] M. C. G. Pawan Kumar, Kunwar Pal and A. Choudhary, "Rapid a*: a robust path planning scheme for uavs," *Int J Intell Robot Appl*, 2023, dOI: https://doi.org/10.1007/s41315-023-00294-y.
[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241, dOI: https://doi.org/10.1007/978-3-319-24574-4_28.
[24] Y. B. Ian Goodfellow and A. Courville, *Deep Learning*. London, England: MIT Press, 2017, dOI: https://doi.org/10.1007/s10710-017-9314-z.
[25] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013, dOI: https://doi.org/10.1109/TPAMI.2013.50.
[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015, dOI: https://doi.org/10.1038/nature14539.
[27] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016, dOI: https://doi.org/10.48550/arXiv.1603.07285.
[28] R. Y. A. A. Gabriel O. Flores-Aquino, Jheison Duvier Diaz Ortega, O. O. G.-F. Raul Lopez Munoz, and J. I. Vasquez-Gomez, "2d grid map generation for deep-learning-based navigation approaches," in *2021 IEEE International Conference on Mechatronics, Electronics and Automotive Engineering*, 2021, dOI: https://doi.org/10.48550/arXiv.2110.13242.
[29] A. P. Schoellig, C. Wiltsche, and R. D'Andrea, "Feed-forward parameter identification for precise periodic quadrocopter motions," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 4313–4318, dOI: https://doi.org/10.1109/ACC.2012.6315248.
[30] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1024–1031, dOI: https://doi.org/10.1109/ROBOT.1999.772448.
[31] H. I. Choi, S. W. Choi, and H. P. Moon, "Mathematical theory of medial axis transform," *pacific journal of mathematics*, vol. 181, no. 1, pp. 57–88, 1997, dOI: http://doi.org/10.2140/pjm.1997.181.57.

Gabriel O. Flores-Aquino received his B.Sc. degree in Mechatronics Engineering in 2017 from the Professional School of Engineering and Advanced Technologies of the National Polytechnic Institute of Mexico (UPIITA-IPN). He received the M.Sc. degree in 2019 and the Ph.D. degree in 2023, both in Advanced Technology from the National Polytechnic Institute in Mexico. He is currently a postdoctoral researcher at the Mathematical Research Center (CIMAT), Guanajuato, Mexico. His research interests include mobile robotics, motion planning, pursuit-evasion problems, and artificial intelligence.

Octavio Gutierrez-Frias received his B.S. degree in Mechatronics from the Professional School of Engineering and Advanced Technologies of the National Polytechnic Institute of Mexico (UPIITA-IPN) in 2003. He received an M.Sc. degree in Computing Engineering from the Computing Research Center of the National Polytechnic Institute in 2006. In 2009, he received a Ph.D. in Computer Sciences at the CIC-IPN. Since 2012, he has been with the Graduate Section at UPIITA-IPN. His research focuses on nonlinear systems control, underactuated systems, robotics, and automation.

Juan Irving Vasquez received his M.Sc. and Ph.D. degrees from the National Institute for Astrophysics, Optics, and Electronics (INAOE), Mexico, in 2009 and 2014, respectively. He earned his B.S. degree in Computer Sciences from the Tehuacan Institute of Technology, Mexico, in 2006. From 2016 to 2021, he served as a researcher at the National Council of Science and Technology (CONACYT) in Mexico. Since 2021, he has been a full-time professor at the National Polytechnic Institute (IPN). His research interests include robotics, motion planning, view planning, and their applications to object reconstruction, inspection, and surveillance.