An Adaptive Row Crops Path Generator with Deep Learning Synergy

An Adaptive Row Crops Path Generator with Deep Learning Synergy

1st Simone Cerrato
*Politecnico di Torino**
Turin, Italy
simone.cerrato@polito.it

2nd Diego Aghi
*Politecnico di Torino**
Turin, Italy
diego.aghi@polito.it

3rd Vittorio Mazzia
*Politecnico di Torino**
Turin, Italy
vittorio.mazzia@polito.it

4th Francesco Salvetti
*Politecnico di Torino**
Turin, Italy
francesco.salvetti@polito.it

5th Marcello Chiaberge
*Politecnico di Torino**
Turin, Italy
marcello.chiaberge@polito.it

***Abstract*—The autonomous navigation of agricultural field machines strongly depends on the global path generation system. Indeed, a correct and effective path construction heavily influences the overall navigation stack compromising the successfulness of the robot mission. However, the most commonly used search algorithms struggle to adapt to environments where a significant prior knowledge of the domain is not negligible. Despite this crucial factor, path generation for row-based crops has received little attention from the research community so far. The proposed research introduces a novel global path planning system that works in synergy with a deep learning model to provide an accurate and centered path with respect to the rows of the analyzed crop. It guarantees the full coverage of the given occupancy grid with less processing time compared to other available literature solutions. Moreover, the presented methodology can detect an anomaly in the path generation and provide the hypothetical user feedback of the missing full coverage of the given crop. Indeed, especially in a practical application, the correct coverage and centrality of the path are essential for effective autonomous navigation. Experimentation with synthetic and real-world satellite occupancy grid maps clearly show the advantages of the proposed methodology and its intrinsic robustness.**

***Index Terms*—Autonomous Navigation, Global Path Planning, Precision Agriculture, Deep Learning, Service Robotics.**

I. Introduction

With the continuous growth of the world population, the agriculture industries must focus on developing new technologies aimed at maximizing efficiency and product quality in a sustainable way. Indeed, over the past years, precision agriculture and smart farming have gained significant attention from the research community [1] introducing robotics and artificial intelligence into the agricultural processes as a means to cut production costs, reduce the required resources, and face labor shortage by also optimizing production quality and quantity. In this context, autonomous navigation plays a crucial role; in fact, agricultural machinery endowed with a self-driving system and the proper set of accessories can be used to replace labors on high-intensive tasks such as harvest [2], seed [3], spray, and irrigate [4]. Moreover, when configured as a platform, it can carry workers to prune and thin trees, thus eliminating the inefficiencies and injuries related to ladders [5]. In addition, these autonomous vehicles can provide real-time crop inventory management [6], as well as a disease monitoring system for plants and fruits [7]–[9].

Thanks to this variety of possible applications, many efforts have been made to come out with innovative autonomous guidance systems for row crops. Indeed, the literature presents different approaches based on 2D-Lidar [10], GPS [11], deep learning [12], visual odometry [13], and computer vision [14].

Nevertheless, a reliable autonomous navigation system strongly depends on the planned trajectory; therefore, a precise and effective path generator is necessary to ensure high performance and safety. Despite so, the path generation problem for row-crops environments has been a bit neglected. Indeed, only a few studies are available; the most common solutions apply clustering to satellite images or aerial drone footage to extract information related to the position and the orientation of the rows. For instance, in [15], Zoto et al. identify the vineyard rows from UAV imagery and exploit the gathered information to automatically perform the path planning. However, as already demonstrated [16], this type of tasks is computationally heavy and complex. On the other hand, alternative solutions such as [17] require a significant amount of information regarding the location and orientation of the crops that make the whole pipeline very time-consuming.

In [18], the authors present DeepWay, a deep learning waypoints estimator for row-based crops global path planning. It simply requires an occupancy grid input map of the analyzed crop in order to estimate all the waypoints of the given map. Nevertheless, a path search algorithm is still required to provide the final global path. The most commonly used search algorithms, including A* and RRT*, can be computationally demanding and, above all, far from the optimal requested path. Indeed, row crops, in addition to connect all the waypoints and cover the full extension of the crop, require paths evenly

This work was supported by PIC4SeR (https://pic4ser.polito.it) and SmartData (https://smartdata.polito.it) interdepartmental centers.

*Department of Electronics and Telecommunications

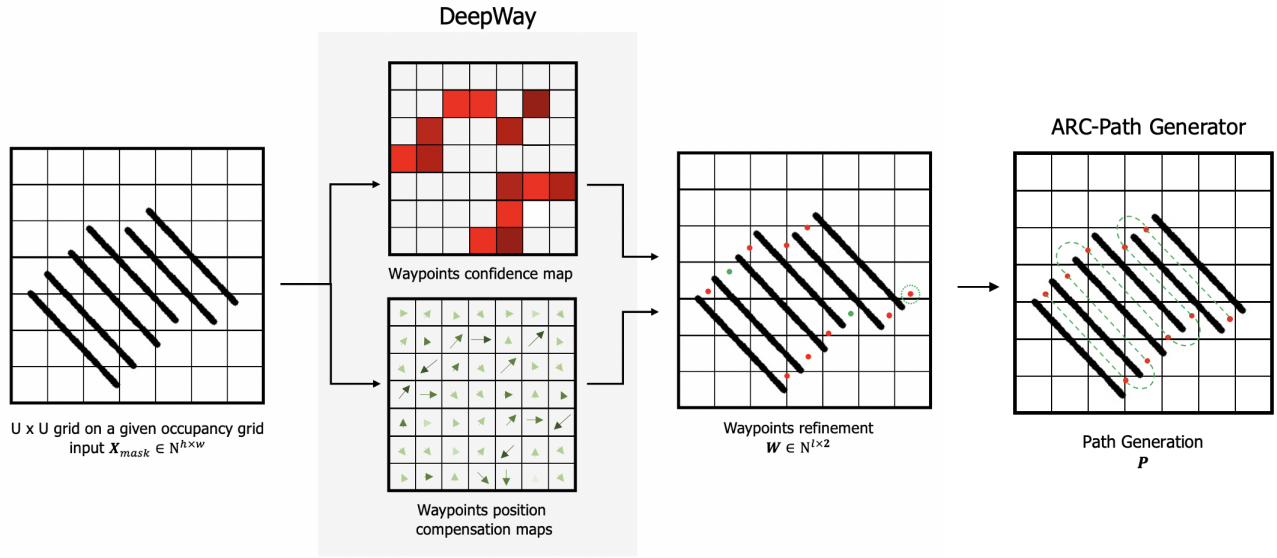


Fig. 1: Given an occupancy grid of the analyzed crop $\boldsymbol{X}_{mask}$, DeepWay estimates the global waypoints, $\hat{\boldsymbol{Y}}^{(i)}$, directly from the full input image. Subsequently, a waypoint refinement algorithm post-processes the prediction of the network, taking care of possible missing and misplaced waypoints (green dots and dashed circle). Finally, the adaptive row crops path generator produces a global path plan $\boldsymbol{P}$ ensuring the full coverage of the crop and the centrality with respect the to rows.

spaced and centered from the border of the rows.

So, building on top of the DeepWay framework, a fast and accurate adaptive row crops global path planner is proposed that exploits the prior knowledge of the environment to produce a centered and smooth path with on average less processing time compared with available literature solutions. Moreover, such novel solution is capable of detecting an anomaly in the pipeline generation and providing as output feedback of the missing full coverage of the given crop. That, used in conjunction with active user intervention, would provide an assurance of the correctness and full coverage of the path. Overall, the proposed algorithm is capable of providing an accurate and reliable global path at a low computational cost, given an occupancy grid map.

The remainder of the article is organized as follows. Section 2 covers the proposed methodology with an explanation of the DeepWay architecture, the waypoint refinement, and the fast adaptive row crops path generation process. Section 3 presents the experimental results and discussion. Finally, section 4 draws some conclusions.

II. Methodology

Given an occupancy grid of the analyzed crop $\boldsymbol{X}_{mask} \in \mathbf{N}^{h \times w}$, where h and w are the height and width, respectively, DeepWay predicts the global path waypoints directly from the full input image, straight from image pixels to points in one evaluation. The end rows waypoint detection is framed as a regression problem, estimating positions of the different points with a fully convolutional deep neural network. So, as shown in the following equation,

$$\hat{\boldsymbol{Y}}^{(i)} = H_{DeepWay}(\boldsymbol{X}^{(i)}_{mask}, \Theta) \tag{1}$$

DeepWay, parametrized with Θ, takes a i-th $\boldsymbol{X}^{(i)}_{mask}$ occupancy grid map and produces the final waypoints estimation in the original input space $\hat{\boldsymbol{Y}}^{(i)} \in \mathbf{N}^{h \times w}$. Subsequently, a waypoint refinement algorithm post-processes the prediction of the network, taking care of possible missing and misplaced waypoints. Moreover, it further elaborates $\hat{\boldsymbol{Y}}^{(i)}$, clustering all the predictions and ordering them. The output of this post-processing phase is an ordered list of waypoints, $\boldsymbol{W} \in \mathbf{N}^{l \times 2}$, ready to be processed with a search algorithm. Finally, the proposed fast adaptive row crops path generator produces a global path plan, efficiently connecting all predicted waypoints. The final result is a global path that covers the crop in its full extension, strictly maintaining the centrality with respect to the crops. The full pipeline is summarized in Fig. 1.

A. Waypoints Estimation

DeepWay is a fully convolutional neural network that is directly fed with an occupancy grid map of a row-based crop $\boldsymbol{X}^{(i)}_{mask}$, and predicts waypoints for the successive global path generation. The input spatial dimension is reduced with a stack of N residual reduction modules. The synergy of the channel and spatial attention layers lets the network focusing on more promising and relevant features. Finally, the neural network outputs a tensor $\boldsymbol{Y}^{(i)}$ of dimension $U \times U \times 3$, where the first channel features a probability $P(u)$ for each cell u and the second and third channels contain the position compensation couple Δ_x, Δ_y, respectively. Indeed, the considered methodology divides the input image of dimension $h \times w$ into a $U_h \times U_w$ grid and, if the centre of an end row waypoint falls into a grid cell u, that cell is responsible for detecting that point. More specifically, the two values (Δ_x, Δ_y) predicted for each cell u let displace a possible prediction respect to a

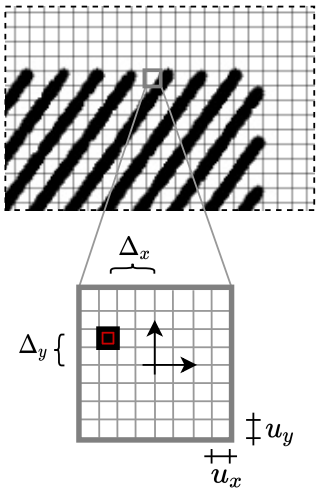


Fig. 2: DeepWay estimates for each cell u of a grid $U_h \times U_w$ a certain probability $P(u)$ and position compensations Δ_x, Δ_y to better adjust detected waypoints on the original occupancy map with dimension $h \times w$. The magnification shows with a red square the actual position of the ground truth taken into account and the need to displace the prediction from the centre of the cell.

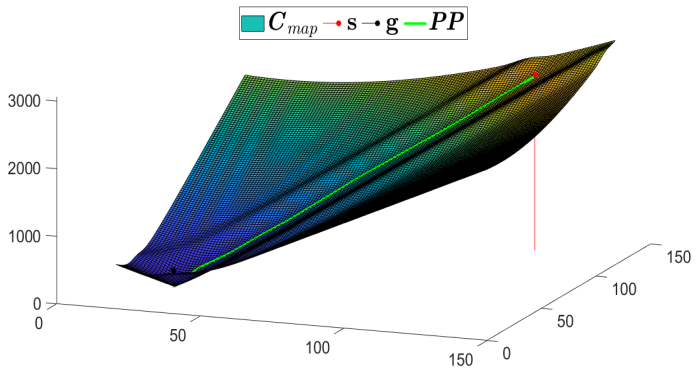


Fig. 3: A representation of the costmap used by the gradient planner function to find the inter-row partial path. The axes scale is represented in terms of pixels.

reference $\mathcal{R}_U$, placed in the centre of the cell. Consequently, the coordinates of a certainly detected waypoint in the original input dimension $h \times w$ can be found using the following equation:

$$\hat{\mathbf{y}}_O = k(\hat{\mathbf{y}}_U + \frac{\boldsymbol{\Delta} + \mathbf{1}}{2}) \tag{2}$$

where $\hat{\mathbf{y}}_O$ and $\hat{\mathbf{y}}_U$ are the two vectors containing the coordinates x and y in the input $\mathcal{R}_O$ and output $\mathcal{R}_U$ reference frames, respectively. Position compensations are normalized, and the reference frame, $\mathcal{R}_u$, of the cell u is centered with respect to the cell itself, as shown in Fig. 2.

So, in order to obtain the waypoints estimation in the original input space, $h \times w$, a certain confidence threshold t_c is applied to the first channel in order to filter all detected waypoints with a probability $P(u) > t_c$. Subsequently, Eq. 2 is exploited together with the position compensations Δ_x, Δ_y maps in order to obtain the respective coordinates on the original reference frame of the input $\mathcal{R}_O$. Finally, a simple waypoint suppression process is applied to remove all predicted points with a reciprocal Euclidean distance inferior to a certain threshold d_c. $P(u)$ values are used to discriminate points to be maintained. The final output is a binary matrix, $\hat{\boldsymbol{Y}}^{(i)}$, with dimension, $h \times w$, that embeds in its representation if a waypoint is present or absent from every i, j cell, $\mathbb{1}_{i,j}^{wp}$.

B. Waypoints Refinement

The waypoints predicted by the neural network are then post-processed in order to reach the following goals:

- the points should be organized into two main clusters that contain the starting and ending point of each row
- in each cluster the point should be ordered so that it is possible to identify the right connections between them
- duplicated points should be removed and missing points should be added whenever possible

An initial clustering is obtained using the density-based clustering algorithm DBSCAN [19]. This approach clusters together points depending on their density and creates a variable number of groups, depending on the specific geometry of the field. The groups computed with this first clustering are merged with a heuristic approach based on the clusters size and position until two main clusters representing the starting and ending points of the rows are obtained. The ordering of the points in each cluster is performed by sorting their projections along the perpendicular to the rows. This direction is estimated with the progressive probabilistic Hough transform technique [20].

To remove duplicated points and add missing ones, the number of rows crossed by the line are computed connecting each subsequent couple of points in a cluster. The correct condition is when a single row is found; when no rows are detected, the two points are redundant, so they are replaced with their average; when more than one crosses occur, rows with no points are present, thus the missing points are added in the middle between the crossing points. Eventually, the final order is obtained by selecting the points from the two final clusters following an A-B-B-A scheme to get the ordered list of waypoints $\boldsymbol{W} \in \mathbf{N}^{l \times 2}$. Each couple of points of this list is used as start and goal for the path generator algorithm.

Code and data related to DeepWay are available on Github[1].

C. Adaptive Row Crops Path Generator

A fast, adaptive, and innovative global path planner is designed that exploits both the general-purpose A* algorithm and the fast computing gradient descent principle, in order to speed up the path computation and maintain the row crop centrality. The planner is able to find a suitable solution in a straightforward manner. First of all, it takes as input: the occupancy grid image $\boldsymbol{X}_{mask} \in \mathbf{N}^{h \times w}$ (where obstacles and free spaces are represented with 0 and 255, respectively), the

[1]https://github.com/fsalv/DeepWay

Algorithm 1 Adaptive Row Crops Path Generator

Input: $\boldsymbol{X}_{mask}$, $\boldsymbol{W}$, $\boldsymbol{M}$
Output: $\boldsymbol{P}$

function GRADIENT_PLANNER(**s**,**g**,$\boldsymbol{O}$,$\boldsymbol{C}_{map}$)
  $c_{min} \leftarrow \infty$
  current_node $\leftarrow$ **s**
  $d \leftarrow Euclide\_distance($**current_node**$, \mathbf{g})$
  while $d > th$ **do**
    for $j = 1, \cdots, q$ **do**
      $x \leftarrow current\_node_1 + M_{j,1}$
      $y \leftarrow current\_node_2 + M_{j,2}$
      if $\boldsymbol{C}_{map_{(y,x)}} < c_{min} \wedge (x, y) \notin \boldsymbol{V}_n \wedge (x, y) \notin \boldsymbol{O}$ **then**
        $\mathbf{n}_{min} \leftarrow$ **current_node** $+ \boldsymbol{M}_{j,:}$
        $cost_{min} \leftarrow C_{map_{(y,x)}}$
      end if
      $\boldsymbol{V}_n \leftarrow \boldsymbol{V}_n \cup (x, y)$
    end for
    current_node $\leftarrow \mathbf{n}_{min}$
    $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup \mathbf{n}_{min}$
    $d \leftarrow Euclide\_distance($**current_node**$, \mathbf{g})$
  end while
  return $\boldsymbol{R}$
end function

$\boldsymbol{O} \leftarrow$ obstacle_extraction($\boldsymbol{X}_{mask}$)
for $i = 1, \cdots, l-1$ **do**
  s $\leftarrow \boldsymbol{W}_{i,:}$
  g $\leftarrow \boldsymbol{W}_{i+1,:}$
  $k \leftarrow$compute_kernel_size(**s**,**g**,$\boldsymbol{O}$)
  if $i\%2 == 0$ **then**
    if $k == 1$ **then**
      $k = k_{min}$
    end if
    $\boldsymbol{C}_{map} \leftarrow$compute_costmap_gradient(k,$\boldsymbol{X}_{mask}$)
    $\boldsymbol{PP} \leftarrow$ gradient_planner(**s**,**g**,$\boldsymbol{O}$,$\boldsymbol{C}_{map}$)
  else
    $\boldsymbol{H}_{map} \leftarrow$compute_occupancy_map($k$,$\boldsymbol{X}_{mask}$)
    $\boldsymbol{PP} \leftarrow$ a_star_planner(**s**,**g**,$\boldsymbol{H}_{map}$)
  end if
  $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \boldsymbol{PP}$
end for

waypoints computed by the deep neural network and successive refinement $\boldsymbol{W} \in \mathbf{N}^{l \times 2}$ and the admissible movements among cells $\boldsymbol{M} \in \mathbf{N}^{q \times 2}$. Second, the obstacles positions are extracted from the mask image, checking whose pixels are set to 0 and storing the row-column pairs in the 2D arrays $\boldsymbol{O} \in \mathbf{N}^{m \times 2}$. Then, the procedure exploits the gradient descent principle in case the partial path to be computed should lay inside the row crop, while it uses the well-known A* algorithm to switch between different rows crop.

In the first case, a costmap $\boldsymbol{C}_{map} \in \mathbf{R}^{h \times w}$ is obtained, shown in Fig. 3, overlapping two custom 3D functions:

(a) (b)

Fig. 4: Two examples of synthetic occupancy grids. The black pixels are a representation of the rows crop.

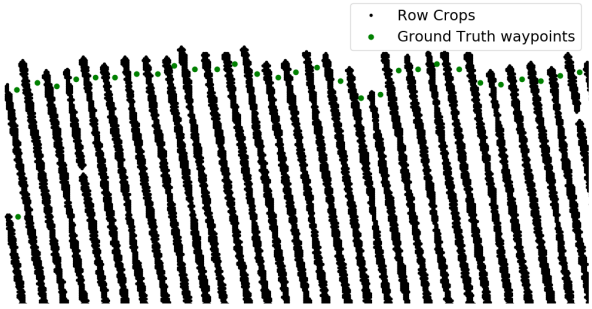


Fig. 5: An example of ground truth waypoints placement.

$$D_{goal_{(i,j)}} = \left(\sqrt{(i - g_i)^2 + (j - g_j)^2} \right)_{\substack{i=1,\ldots,h \\ j=1,\ldots,w}} \tag{3}$$

$$D_{goal_{(i,j)}} = \left(k_{goal} \cdot \frac{D_{goal_{(i,j)}}}{max(\boldsymbol{D}_{goal})} \cdot 255 \right)_{\substack{i=1,\ldots,h \\ j=1,\ldots,w}} \tag{4}$$

$$\boldsymbol{X}_{blur} = k_{blur} \cdot (\boldsymbol{K} * \boldsymbol{X}_{mask}) \tag{5}$$

$$C_{map_{(i,j)}} = \left(D_{goal_{(i,j)}} + X_{blur_{(i,j)}} \right)_{\substack{i=1,\ldots,h \\ j=1,\ldots,w}} \tag{6}$$

where $\boldsymbol{D}_{goal} \in \mathbf{R}^{h \times w}$ takes into account the distance from the considered goal $\mathbf{g} \in \mathbf{N}^2$, $\boldsymbol{X}_{blur} \in \mathbf{N}^{h \times w}$ is obtained through a convolution operation with a Gaussian kernel $\boldsymbol{K} \in \mathbf{N}^{k \times k}$ over the mask image $\boldsymbol{X}_{mask}$, k_{goal} and k_{blur} are two gains to weight the distance from the goal and the blurring effect, respectively. Finally, a portion of the total path $\boldsymbol{PP} \in \mathbf{N}^{u \times 2}$ is computed exploiting the gradient planner function, well described in Algorithm 1, where $\boldsymbol{R} \in \mathbf{N}^{u \times 2}$ is the computed portion of total path, while $\mathbf{s} \in \mathbf{N}^2$ and $\mathbf{g} \in \mathbf{N}^2$ are 1D arrays that contain the row and column indexes of the start and goal points, respectively.

In case of different rows crop, an occupancy map $\boldsymbol{H}_{map} \in \mathbf{N}^{h \times w}$ with enlarged obstacles is built using,

$$\boldsymbol{T}_{conv} = \boldsymbol{K} * \boldsymbol{X}_{mask} \tag{7}$$

$$H_{map}{}_{\substack{i=0,\ldots,h \\ j=0,\ldots,w}}(i,j) = \begin{cases} 1, & \text{if } T_{conv_{(i,j)}} \leq th_{dynamic} \\ 0, & \text{if } T_{conv_{(i,j)}} > th_{dynamic} \end{cases} \tag{8}$$

In these equations, $\boldsymbol{T}_{conv} \in \mathbf{N}^{h \times w}$ stores the result of the convolution operation and $th_{dynamic}$ is a dynamic threshold,

established to avoid issues in case of close row crops as follows:

$$th_{dynamic} = th_{start} + (\frac{k-1}{2}) \cdot step \tag{9}$$

When the dynamic threshold $th_{dynamic}$ is greater than a predefined maximum value th_{max}, it will be set equal to such value. Moreover, in Eq. 9, th_{start} refers to a minimum value of the threshold and $step$ is a scalar. Eventually, the partial path between different rows crop is obtained by means of the A* algorithm, using the occupancy map with enlarged obstacles.

The overall algorithm exploits the dynamic computation of the kernel size k,

$$k = (d_{min} - (d_{min} \mod 2)) \cdot 2 + 1 \tag{10}$$

to overcome issues of both different distances between rows crop and waypoints near obstacles. In order to obtain a suitable value of k, the minimum Euclidean distance d_{min} is computed between obstacles $\boldsymbol{O} \in \mathbf{N}^{m \times 2}$, start $\mathbf{s} \in \mathbf{N}^2$ and goal $\mathbf{g} \in \mathbf{N}^2$ nodes, then the obtained number is transformed in an effective kernel size, that means an odd scalar, as shown in Eq. 10. The proposed fast and adaptive path generator does not guarantee to find the global path with full coverage in case of missing waypoints, as may happen. However, it is able to find anomalies in the path generation task and notify a human operator that something is going wrong.

Code and data related to ARC-PG are open source and available on Github[2].

III. EXPERIMENTAL RESULTS AND DISCUSSION

A. Dataset Creation

Since there is not any ready-to-use dataset of row crop occupancy grids available online and creating a real one is a really hard task due to its complexity, a custom synthetic dataset has been generated. Since most of the crop scenarios in the real world present straight rows, an algorithm that exploits geometry properties to produce linear row crops occupancy grids is developed. More specifically, each occupancy grid of dimensions $h \times w$ contains a random number of lines from 20 to 50. All the lines share the same angle $\alpha \in [-\pi/2, \pi/2]$ in order to simulate the real-world texture, as represented in Fig. 4. In order to make the occupancy grids more realistic, diverse inter-row distances are set on the same image, the rows with circles of radius 1 or 2 are thickened for each 1-pixel , and finally the parcels edges are made irregular. Furthermore, the images are randomly rescaled, and holes along the lines are added to simulate errors in the grid generation as a means to increase the robustness of the network.

Regarding the ground truth waypoints, one point is placed between each of the two extremities of every pair of rows, and then they are moved inward the row as shown in Fig. 5. In this way, the path generator task is simplified, since waypoints outside the rows can lead to a wrong choice of the next row to go through.

[2]https://github.com/SimoCerra/FARCPathGenerator

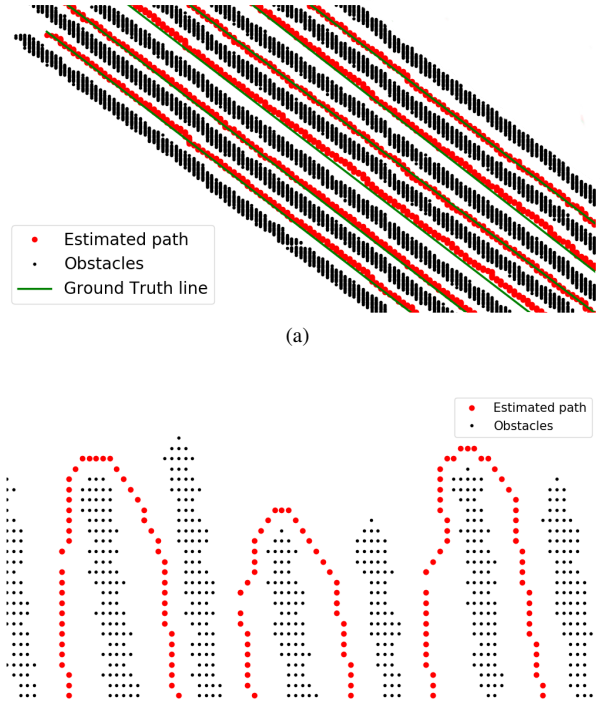


Fig. 6: An example of path solution found by the proposed algorithm ARC-PG. (a) shows the partial path inside the row crops, while (b) represents the portion of the path between two different rows.

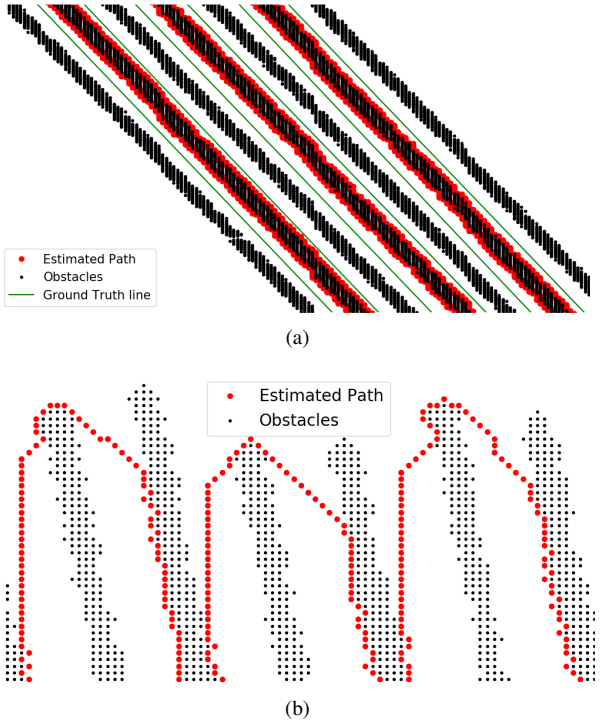


Fig. 7: An example of path solution found by the standard A* algorithm. (a) shows the partial path inside the row crops, while (b) represents the portion of the path between two different rows.

TABLE I: Comparison between different path planning algorithms on the artificial dataset. t and $\#c$ stand for time and number of visited cells, respectively.

Algorithm	$t_{min}[s]$	$t_{max}[s]$	$t_{\mu}[s]$	$\#c_{min}$	$\#c_{max}$	$\#c_{\mu}$
A*	4.89	45.50	10.61	53536	173557	92522
RRT*	56.52	659.08	216.21	39	97	67
ARC-PG	1.44	8.67	4.60	49054	122906	83194

TABLE II: Comparison between A*, RRT* and ARC-PG algorithms on the dataset composed of satellite images. t and $\#c$ stand for time and number of visited cells, respectively.

Algorithm	$t_{min}[s]$	$t_{max}[s]$	$t_{\mu}[s]$	$\#c_{min}$	$\#c_{max}$	$\#c_{\mu}$
A*	1.19	92.78	4.37	26736	436913	70604
RRT*	18.63	475.65	129.38	17	119	55
ARC-PG	0.64	62.96	3.69	23135	355093	63817

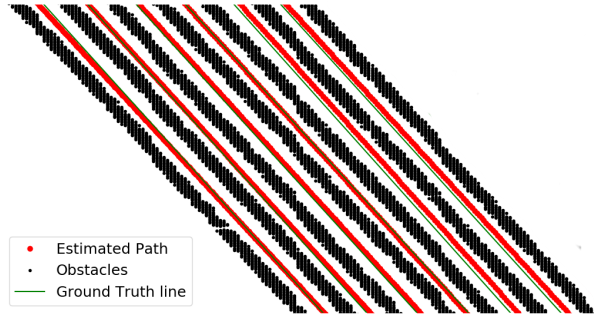
The solely synthetic dataset is not enough to validate the deep neural network, so 100 satellite images of row-based crops are collected from the Google Maps database, then they are manually labelled to extract the corresponding occupancy grids and the ground-truth waypoints. Eventually, both artificial and real-world binary masks are used in order to evaluate the novel approach.

B. Algorithm Comparison

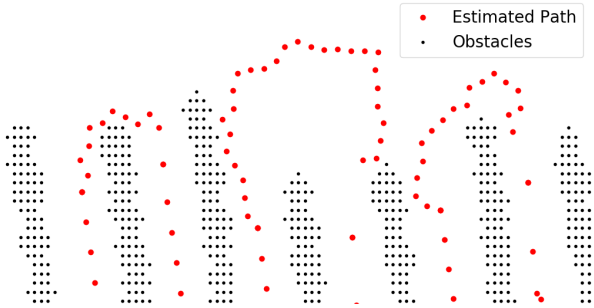
The proposed Adaptive Row Crops Path Generator (ARC-PG) is compared to some other popular path planning algorithms, as A* and RRT*, in order to show how the designed custom solution is able to outperform standard path planning algorithms in terms of time consumption and the number of visited cells, as shown in Tables I and II. During the experimentation, some gains and variables, described in the previous part, are set to a fixed value found with a trial and error procedure over the entire dataset. In particular, $k_{min} = 9$, $k_{goal} = 12$, $k_{blur} = 0.7$, $th_{start} = 180$, $step = 10$, $th_{max} = 240$, and $th = 2$. All the algorithms have been tested in the same working condition on an Intel Core i7-9750H CPU @ 2.60 GHz and 16 Gb of RAM, using 100 synthetic and 100 real-world binary images, in order to check the accuracy of the proposed approach on both artificial and real images.

One of the main advantages of ARC-PG is the ability to detect anomalies during the path computation task, that may be caused by missing or wrong ordered waypoints and very close rows crop. This additional feature comes from the intrinsic structure of the algorithm, indeed it exploits the gradient descent principle inside rows crop and A* algorithm to switch between different rows. As a consequence, in case of the gradient planner is not able to find a valid path between two consecutive waypoints, it means either there are issues with the waypoints (ordering or absence) or the row crops are too close and the planner fails. Such intrinsic feature allows to require the attention of a hypothetical user in order to check what is going wrong in the path computation.

Furthermore, the proposed adaptive path planner shows very promising results in terms of row crop centrality, number



(a)



(b)

Fig. 8: An example of path solution found by the standard RRT* algorithm. (a) shows the partial path inside the row crops, while (b) represents the portion of the path between two different rows.

TABLE III: Comparison of different path planning algorithm in terms of Mean Absolute Error(MAE) and Fault Rate (FR).

Dataset Type	Algorithm	MAE	FR[%]
Synthetic	A*	7.88	34
	RRT*	1.08	34
	ARC-PG	1.60	5
Satellite	A*	8.10	29
	RRT*	1.07	43
	ARC-PG	1.57	31

of visited cells and processing time using both the artificial and the real images dataset. Indeed, it visits a lower number of cells with respect to A* and is faster of both A* and RRT* algorithms, as shown in Tab. I and Tab. II. Moreover, it outperforms the A* regarding the row crop centrality as shown in Tab. III, where the Mean Absolute Error (MAE) has been computed inside the rows crop with respect to the ground truth reference obtained by connecting the ground truth waypoints with a line. The row crop centrality can be visually checked observing Fig. 6 and Fig. 7. Finally, the RRT* is able to maintain a better central path with respect to the novel designed algorithm, as can be visually observed comparing Fig. 6 and Fig. 8 and numerically in Tab. III, however it has very high processing times relative to both dataset type as shown in Tab. I and Tab. II.

For what concern the Fault Rate (FR) on the synthetic dataset, both the A* and the RRT* algorithms have higher values than the proposed solution, as shown in Tab. III, because the main fault conditions are free spaces on the same

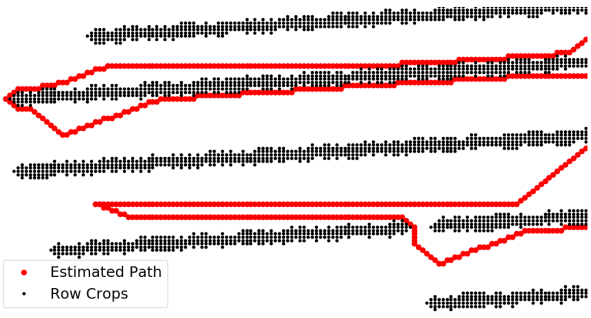


Fig. 9: An example of path fault due to a free space in the row crop.

row crop as shown in Fig. 9 that leads standard path planning algorithms to fail easily. It is worth highlighting that the ARC-PG algorithm does not fail in such particular condition thank to the usage of a custom computed costmap.

For what concerns the real-world dataset, all the considered algorithms have similar values of FR since the main issue is a lack or wrong ordering of estimated waypoints. In such conditions, any of the used planners are not able to find a valid path; however, the innovative proposed ARC-PG succeeds in notifying the unreliability of the current waypoints array.

All considered, the provided results show clear advantages in using the fast row crops path planner with respect to general-purpose and standard global path planning algorithms.

IV. CONLCUSIONS

In the presented work, an innovative and fast computing approach has been proposed to solve the global path planning problem in row crops. As proved by the provided results, the novel solution outperforms the most used global path planning algorithms in terms of computing time and path accuracy with respect to row centrality. Moreover, the designed path generator algorithm is able to detect anomalies in the waypoints array provided by the deep neural network and successive refinement. As a consequence, a hypothetical user may be informed about the missing full coverage of the row crops field. Future works are intended for designing a complete autonomous framework, that takes as input a single remote sensed image of a row-based crop field and provides as output the complete global path.

ACKNOWLEDGMENT

This work has been developed with the contribution of the Politecnico di Torino Interdepartmental Centre for Service Robotics PIC4SeR[3] and SmartData@Polito[4].

REFERENCES

[1] M. Bacco, P. Barsocchi, E. Ferro, A. Gotta, and M. Ruggeri, "The digitisation of agriculture: a survey of research activities on smart farming," *Array*, vol. 3, p. 100009, 2019.

[2] L. Luo, Y. Tang, Q. Lu, X. Chen, P. Zhang, and X. Zou, "A vision methodology for harvesting robot to detect cutting points on peduncles of double overlapping grape clusters in a vineyard," *Computers in industry*, vol. 99, pp. 130–139, 2018.

[3]https://pic4ser.polito.it

[4]https://smartdata.polito.it

[3] A. Keerthana, P. Kirubaharan, S. Krishnamoorthy, K. Rajeswari, and G. S. Zabiyullah, "Agriculture robot for seeding and forming," *International Research Journal of Engineering and Technology*, vol. 5, no. 5, 2018.

[4] I. Adabara, N. Hiriji, O. Emmanuel, S. M. Alkasim, K. Zaina, and M. M. Mustafa, "Intelligent embedded agricultural robotic system," 2019.

[5] B. Davis, "Cmu-led automation program puts robots in the field," *Mission Critical*, pp. 38–40, 2012.

[6] N. Virlet, K. Sabermanesh, P. Sadeghi-Tehran, and M. J. Hawkesford, "Field scanalyzer: an automated robotic field phenotyping platform for detailed crop monitoring," *Functional Plant Biology*, vol. 44, no. 1, pp. 143–153, 2017.

[7] S. Ramesh, R. Hebbar, M. Niveditha, R. Pooja, N. Shashank, P. Vinod *et al.*, "Plant disease detection using machine learning," in *2018 International conference on design innovations for 3Cs compute communicate control (ICDI3C)*. IEEE, 2018, pp. 41–45.

[8] X. E. Pantazi, D. Moshou, and A. A. Tamouridou, "Automated leaf disease detection in different crop species through image features analysis and one class classifiers," *Computers and electronics in agriculture*, vol. 156, pp. 96–104, 2019.

[9] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: an edge ai application," *IEEE Access*, vol. 8, pp. 9102–9114, 2020.

[10] V. A. Higuti, A. E. Velasquez, D. V. Magalhaes, M. Becker, and G. Chowdhary, "Under canopy light detection and ranging-based autonomous navigation," *Journal of Field Robotics*, vol. 36, no. 3, pp. 547–567, 2019.

[11] N. Khan, G. Medlock, S. Graves, and S. Anwar, "Gps guided autonomous navigation of a small agricultural robot with automated fertilizing system," SAE Technical Paper, Tech. Rep., 2018.

[12] D. Aghi, V. Mazzia, and M. Chiaberge, "Autonomous navigation in vineyards with deep learning at the edge," in *International Conference on Robotics in Alpe-Adria Danube Region*. Springer, 2020, pp. 479–486.

[13] S. Zaman, L. Comba, A. Biglia, D. R. Aimonino, P. Barge, and P. Gay, "Cost-effective visual odometry system for vehicle motion control in agricultural environments," *Computers and Electronics in Agriculture*, vol. 162, pp. 82–94, 2019.

[14] D. Aghi, V. Mazzia, and M. Chiaberge, "Local motion planner for autonomous navigation in vineyards with a rgb-d camera based algorithm and deep learning synergy," *Machines*, vol. 8, no. 2, p. 27, 2020.

[15] J. Zoto, M. A. Musci, A. Khaliq, M. Chiaberge, and I. Aicardi, "Automatic path planning for unmanned ground vehicle using uav imagery," in *International Conference on Robotics in Alpe-Adria Danube Region*. Springer, 2019, pp. 223–230.

[16] I. Vidović and R. Scitovski, "Center-based clustering for line detection and application to crop rows detection," *Computers and electronics in agriculture*, vol. 109, pp. 212–220, 2014.

[17] L. Comba, A. Biglia, D. R. Aimonino, and P. Gay, "Unsupervised detection of vineyards by 3d point-cloud uav photogrammetry for precision agriculture," *Computers and Electronics in Agriculture*, vol. 155, pp. 84–95, 2018.

[18] V. Mazzia, F. Salvetti, D. Aghi, and M. Chiaberge, "Deepway: A deep learning waypoint estimator for global path generation," *Computers and Electronics in Agriculture*, vol. 184, p. 106091, 2021.

[19] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[20] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer vision and image understanding*, vol. 78, no. 1, pp. 119–137, 2000.