Incremental Contour-Based Topological Segmentation for Robot Exploration

L. Fermin-Leon, J. Neira and J. A. Castellanos

***Abstract*—We propose an alternative to the common approaches to the topological segmentation in structured or unstructured environments, Contour-Based Segmentation. It is faster and equally accurate, without the need of fine tuning parameters or heuristics. During robotic exploration, we propose an incremental version that reduces the processing time by reusing the previous segmentation. Tests demonstrate the velocity and quality in room segmentation in both batch and incremental mode. Tests also demonstrate the incremental version outperforms the state of the art in incremental topological segmentation.**

***Keywords*—Topological Segmentation, Mapping, SLAM**

I. Introduction

State of the art SLAM algorithms provide accurate information on the metric representation of the environment. However many tasks like human-robot interaction, exploration, surveillance, among others, would benefit from a higher level forms of representation and reasoning.

Reasoning about the environment includes decomposing it into meaningful parts. In indoors environments this is known as "room segmentation". The result is commonly associated to rooms and corridors. In the general case this decomposition this representation is known as "topological maps".

In topological maps the environment is partitioned into regions, the only information being represented are their characteristics and its connectivity. This comes from mathematical topology, considering these connections invariant to deformation. The main difference with other mapping approaches is that regions are not restricted to be represented metrically or semantically, for example in [2] the topological map is constructed without any metric information or semantic labeling, only the appearance and connectivity are used.

The traditional approach for constructing a topological map with metric information in its regions consists in projecting the metric structure into an image and then applying image segmentation techniques. In 3D meshes there are two general types of segmentation: firstly the segmentation based on the surface characteristics, similar to 2D segmentation; and secondly segmentation known as "part segmentation" where the goal is to segment the object represented by the mesh into meaningful, mostly volumetric, parts [7]. Under the 2D constrains the mesh is equivalent to the contour of the image and the "part segmentation" techniques can be used for images.

*This work has been supported by the MINECO-FEDER project DPI2015-68905-P, Grupo DGA T04-FSE, and the research grant Fundación Carolina

Leonardo Fermín-León, José Neira and José Angel Castellanos are with the Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza 50018, Spain lfermin@unizar.es, jneira@unizar.es, jacaste@unizar.es

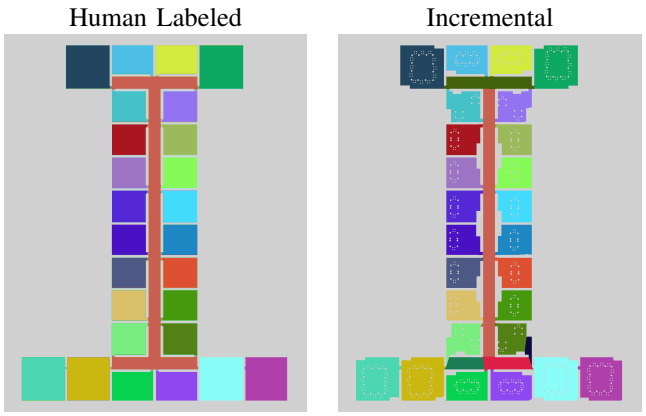

Fig. 1. Motivational example from a publicly available dataset [3]: human-labeled segmentation of an office environment (left, size 82×102 m^2) vs. the result of the incremental topological segmentation algorithm proposed in this paper (right). (Figures best viewed in color)

In this work we propose the use of part segmentation approaches for the construction of 2D topological maps, i.e. Contour Based Topological Segmentation.

Since there are not mathematical restrictions in the number of possible regions, the definition of the quality of a segmentation is usually related to the application. Some mathematical constraints are presented in [1] but those are specific to the path planning problem, and they are not easily extrapolable.

In order to test the quality of the Contour-based segmentation, we use the room segmentation scenario in which the "correct" segmentation is defined as the human labeling of the map, i.e. rooms and corridors. The similarity of our segmentation determines its quality [3].

In preparation for the autonomous exploration we also developed an incremental version of the segmentation algorithm. This reduces processing time and permits its use on-line, the robot can effectively explore the environment while reasoning about it.

Performance in datasets demonstrates that this new approach produces a high quality segmentation (Fig 1) with low processing time. Our incremental algorithm further reduces this processing time without sacrificing the quality.

II. Related Work

A. Voronoi-based Segmentation

The most popular approaches to segmenting floor plans are based on the Generalized Voronoi Graph. In general, these algorithms start by transforming the 2D grid based map to a binary image, where 0 is an occupied cell and 1 a free cell.

The Generalized Voronoi Diagram can be defined as the centers of the maximal disks inscribed in the free cells image. These points form a set of intersecting lines. The Generalized Voronoi Graph (GVG) is built defining the intersecting points as nodes and the lines connecting them as edges.

The original formulation for segmentation based on Generalized Voronoi Graph comes from [8]. In this work Thrun defined the critical points as the points on the Voronoi diagram that minimize clearance locally, i.e. doors. The regions are defined by the nodes in the GVG and extends until the critical points are reached.

Several heuristics can be introduced to improve GVG based segmentation. In [3] Bormann *et al.* prune the extracted GVG to a mayor skeleton by collapsing leave edges into the node points of their origins. Next, they define a set of critical point candidates, and then some heuristics in point selection, as well as merging, are applied. They classify the traditional approaches according to the criteria each follows: Morphological, Distance Transform, GVG and Features. An implementation of each approach is tested in 20 images, corresponding to indoors environments in different configurations (furnished and without furniture) and compared with human label segmentation. In this dataset, the Voronoi-based approach scores the highest quality in terms of precision and recall. However it exhibits one of the highest processing times compared with the other approaches.

Incremental versions of GVG improves its velocity, increasing the possibility of being used on line in exploration tasks. The most recent algorithm is presented in [6]. They incrementally construct a Voronoi graph-based topological segmentation for semi-structured environments (unlike room segmentation). They report a high dependence of processing time with respect to cell size: the change of the cell size from 0.1m to 0.25m reduces the maximum processing time from 50 to 2.5 seconds[1].

B. Contour-Based Segmentation

Contour-based Segmentation is associated with "Part Segmentation", a type of segmentation commonly associated to 3D meshes [7]. Its goal is the segmentation of the object represented by the mesh into meaningful parts. When applied to 2D, the image is represented as a set of closed contours and processed to generate sub-contours. In the contour representation lies the most significant difference between this approach and the current approaches used for topological segmentation.

Unlike the pixel based segmentation, contour-based complexity doesn't depend on the image size (a rectangular shape can be represented by only 4 contour points), but rather in the characteristic of the contour.

One common criteria for part segmentation is convexity, finding the exact convex decomposition of the contour. According to [4], this decomposition can be costly to construct and can result in representations with an unmanageable number of components. Furthermore, if the polygon has holes the problem is NP-hard. The alternative they propose is the Approximate Convex Decomposition (ACD). In this way they can produce a hierarchical representation of convex decompositions of different levels of approximation. They claim the complexity of this decomposition to be $\mathcal{O}(nr)$ with n the number of vertexes and r the number of notchs (the most significant non-convex feature). This decomposition provides an insight of the characteristics of the shape.

[1]Processing time extracted from the figure 13 in [6]

III. Dual- Space Decomposition

Other recent works in part segmentation point to decompose two-dimensional shapes into functional and visually meaningful parts include the Dual-Space Decomposition [5]. In this work the authors demonstrate that the resulting decomposition is statistically similar to the one produced by a human in a dataset composed by complex shapes from the MPEG7 database with added noise and the contour of some images, such as trees, human crowd, and bikes.

In dual space decomposition (DuDe) the input is a polygon P, possibly with holes in it. The polygon and the holes are represented as a sequence of 2D points whose ends are assumed to be connected (same as a closed contour).The algorithm decomposes the polygon P based on the decomposition of the complement $\overline{P}$ (dual-space).

The following definitions are useful to understand the algorithm.

A. Definitions

1) Convex Hull: The smallest convex set which includes the polygon, CH.

2) Pockets: (p_i) The regions inside the convex hull but not inside the contour. Basically the regions the polygon P needs to add to become CH.

$$\bigcup p_i = \overline{P} \tag{1}$$

$$\overline{P} \bigcup P = CH \tag{2}$$

3) Bridge: The segment of the convex hull that limits each $pocket$.

4) Convexity: A polygon is considered convex if for any pair of points inside the polygon, every point on the straight line segment that joins them is also inside the polygon. The smallest convex set which includes a polygon P is called the convex hull. Based on these definitions, several methods to measure the convexity of a polygon [9] based on either the probability of finding a straight line segment inside the polygon or in the similarity to its convex hull.

According to [4], these global measurements of convexity are not useful to identify where and how the polygon should be decomposed. They instead use the measure of concavity (complementary to convexity)

$$concavity(P) = \max_{x \in \partial P} \{dist(x, CH)\} \tag{3}$$

with P the polygon, ∂P its contour and $dist(x, CH)$ is the shortest distance (or path) from the point $x \in \partial P$ to the convex hull CH.

B. Decomposition

This decomposition works iteratively. Firstly the convex hull of the polygon is found, this defines the $pockets$ and its corresponding bridges. The contour of these $pockets$ is iteratively decomposed: every $pocket$ is treated as a polygon with a new convex hull and new $pockets$ that will be decomposed once again till no more pockets are found.

Because of the hierarchical relation of the DuDe decomposition (every polygon is the $pocket$ of a parent polygon), it is possible to measure the distance from every point to its closest bridge, and next the distance from this bridge to the bridge in the parent polygon, and so on till the convex hull is reached. This path defines the distance to the convex hull and thus the concavity.

The set of points with concavity bigger than the concavity threshold are the pivot points of the cut. Finally the set of cuts, using these points, that maximizes the convexity (minimize the concavity) in the the final decomposition, is chosen.

C. Implementation

In order to produce a topological segmentation of the 2D grid based map we transform the occupancy grid into a set of polygons, then we use the function `DuDe_Segment`[2] from [5] to segment them. This function decompose a polygon P represented by a set of n boundaries b_0, b_1, b_2, b_{n-1}, where b_0 is the external boundary and $b_{k>0}$ are boundaries of the holes, until a maximum allowed concavity is reached.

The first step consist in transforming the occupancy grid into a binary image, this is a common step for every topological map segmentation based on occupancy grids. This consist in applying thresholds to the image to obtain the obstacles and the free space. In the case a real robot is used the obstacles and free space are modified according to the robot's characteristics (Algorithm 1).

Algorithm 1: Occupancy Grid to Binary_Image

Input : Occupancy_Grid
Output : Binary_Image

Occupancy_Image = Occ2Image(Occupancy_Grid)

Open_Space = Threshold1(Occupancy_Image)
Obstacle =Threshold2(Occupancy_Image)

`/* In Navigation with Real Robots */`
Obstacle = Inflate_Obstacle(Obstacle)
Open_Space = Filter(Open_Space)

Binary_Image = Open_Space $\cap$ $\overline{\text{Obstacle}}$

return Binary_Image

We transform the set of contours into a polygon representation using the $OpenCV$ ® function `findContours`. Using the retrieval tree mode in this function, we can find all the external contours (Parents) and the set of contours inside them (CHILDREN) in the binary image. Consequently, every set formed by a parent contour and the set of its child contours represents a polygon.

Looping through every external contour and its children, we obtain every decomposition and build the $Decomposed$ set aggregating every decomposition.

Algorithm 2: Dual Space Decomposition of Binary_Image (`DuDe_Binary`)

Input : Binary_Image, Max_Concavity
Output : Decomposed = $\{c_0, c_1, \cdots\}$
$Contour$ $c_i = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots\}$
Variables: $Contour\_Sets$:
Parents = $\{P_0, P_1, \cdots P_k\}$
Child$_i$ = $\{\}$
Dude_Contours = $\{\}$
CHILDREN = $\{\text{Child}_0, \text{Child}_1, \cdots \text{Child}_k\}$

(Parents, CHILDREN) =
`findContours`$_{tree}$(Binary_Image)

for *every contour* $P_i \in$ *Parents* **do**
Dude_Contours =
`DuDe_Segment`($\{P_i$, Child$_i\}$, Max_Concavity)
Decomposed = Decomposed $\cup\{$Dude_Contours$\}$

return Decomposed

IV. Incremental Decomposition

During exploration, whenever a new a scan is processed, some new information is aggregated to the map. This information usually only modifies the contours connected to the new scan, leaving the rest unchanged (except during looop closings). This observation permits the implementation of the incremental version of the decomposition.

In the SLAM context a loop closure can potentially modify the whole map, in this scenario the whole map is decomposed. In this worst case scenario the processing time is the same as the batch implementation, in the regular exploration scenario (no loop closure) the expected processing time is lower.

The first step consist in obtaining the difference of the binary image between time step k and $k-1$, the received occupancy grid is transformed into a binary image using Algorithm 1, it is then compared to the previous map image using algorithm 3 to produce the difference image.

Algorithm 3: Difference Image Map

Input : Binary_Image
Output : Difference_Image
Variables: Previous_Map_Image

Negated_image = invert_image(Previous_Map_Image)
Difference_Image = Binary_Image $\cap$ Negated_image
Previous_Map_Image = Binary_Image

return Difference_Image

[2]Public code of Dual Space Decomposition can be found in http://masc.cs.gmu.edu/wiki/Dude2D

The second step consist in processing the difference image to update the decomposition. The difference image is transformed into the set of contours "$Difference$" using `findContours`, with simple contour approximation and retrieval mode set as external (no hierarchy need). Next, these contours are compared with the previous decomposition to obtain two sets of contours

1) $Unchanged$, Contours in previous decomposition with no contact with the "$Difference$" set.
2) $Modifiable$, It consist in expanding the set "$Difference$" with the contours in the previous decomposition in contact with it

The set "$Modifiable$" is transformed into a reduced binary image and processed using the Dual-Space Decomposition of binary images `DuDe_Binary`. Finally the result of this decomposition is aggregated to the set of unchanged contours to produce the new decomposition. The new decomposition is stored for the next iteration.

Algorithm 4: Incremental Decomposition

Input : Difference_Image, Max_Concavity

Output : Decomposed = $\{c_0, c_1, \cdots\}$

Variables: $Contour\_Sets$:
Previous = $\{p_0, p_1, \cdots\}$
Difference = $\{d_0, d_1, \cdots\}$
Modifiable = $\{\}$
Unchanged = $\{\}$
Modified_Contours = $\{\}$

DIFF = $\texttt{findContours}_{Ext}$(Difference_Image)

for *every* $p_i \in$ *Previous* **do**
  for *every* $d_j \in$ *DIFF* **do**
    if *contours_connected*(p_i, d_j) **then**
      Modifiable = Modifiable $\cup \{p_i, d_j\}$
    else
      Unchanged = Unchanged $\cup \{p_i\}$

Expanded_Image = draw_contours(Modifiable)
Modified_Contours = `DuDe_Binary`(Expanded_Image, Max_Concavity)

Decomposed = Disconnected $\cup$ Modified_Contours
Current_Decomp = Decomposed

return Decomposed

V. Experiments

The first experiment evaluates the performance of the algorithm in the structured environment scenario (room segmentation). We evaluate our approach in both of its versions, batch (`DuDe_Binary`) and incremental, compared to the traditional approaches in the room segmentation dataset of [3] in terms of time, precision and recall.

The second experiment evaluates the performance in poorly structured environments. We use the three exploration sequences described in [6], namely a Parking Lot, a Tunnel and a Building. The lack of ground truth implies the quality of the decomposition cannot be determined using a precision and recall, thus we evaluate the algorithm in terms of the average processing time between the batch and the incremental version.

The algorithms run in a Laptop Intel® Core™i7-2620M CPU @ 2.70GHz 4, under the robot operating system framework (ROS). Our ROS package is based on the function $IncrementalDecomposition$ [3] (Algorithm 4) that takes the occupancy grid ROS message and the previous decomposition, processes it and then decomposes it with the public implementation of [5].

The optimal value for the parameter "$Max\_Concavity$" was found to be $2.5meters$ for every scenario in both experiments, independent of the size, resolution and type of environment.

A. Environment with High Structure (Room Segmentation)

We use the maps proposed in [3], consisting in 20 scenarios. Every scenario contains images of the rooms with and without furniture and the human segmentation of the image. We choose the evaluation in terms of time, precision and recall because the time can be compared (the characteristics of our computer are roughly the same) and precision and recall allows the comparison with a human labeled image.

In this context the precision and recall is evaluated between the human labeled regions R_{human} and the segmented R_{seg}, for every correspondence between regions the true positive tp represent the pixels in both regions, the false positives fp are the pixels in the segmented region but not in the human labeled and the false negatives fn are the pixels in the human labeled region but not in the segmented. Consequently $tp + fp$ is the region R_{human} and $tp + fn$ is R_{seg}. This can be summarized in the following equations.

$$Precision = \frac{tp}{tp + fp} = \frac{R_{human} \cap R_{seg}}{R_{human}} \quad (4)$$

$$Recall = \frac{tp}{tp + fn} = \frac{R_{human} \cap R_{seg}}{R_{seg}} \quad (5)$$

In order to evaluate the incremental version of the algorithm we simulated the exploration with a virtual robot with a radius of laser range of 5 meters. In this situation the map is constructed incrementally and then processed with our algorithm. The precision and recall are calculated in the final map and the time reported is the average processing time per simulated scan.

Because the maps presented have different sizes, number of rooms and complexity, we present the results for every map using histograms for time (Fig 3), precision and recall (Fig 2).

The best result in precision and recall reported in this dataset correspond to the Voronoi-based segmentation, in table I the average values are compared. The contour-based approach has a performance similar to the best evaluated in this dataset, in particular our incremental version does not sacrifice any quality in pursue of increased velocity.

[3]Public Implementation can be found in https://github.com/lfermin77/Incremental_DuDe_ROS

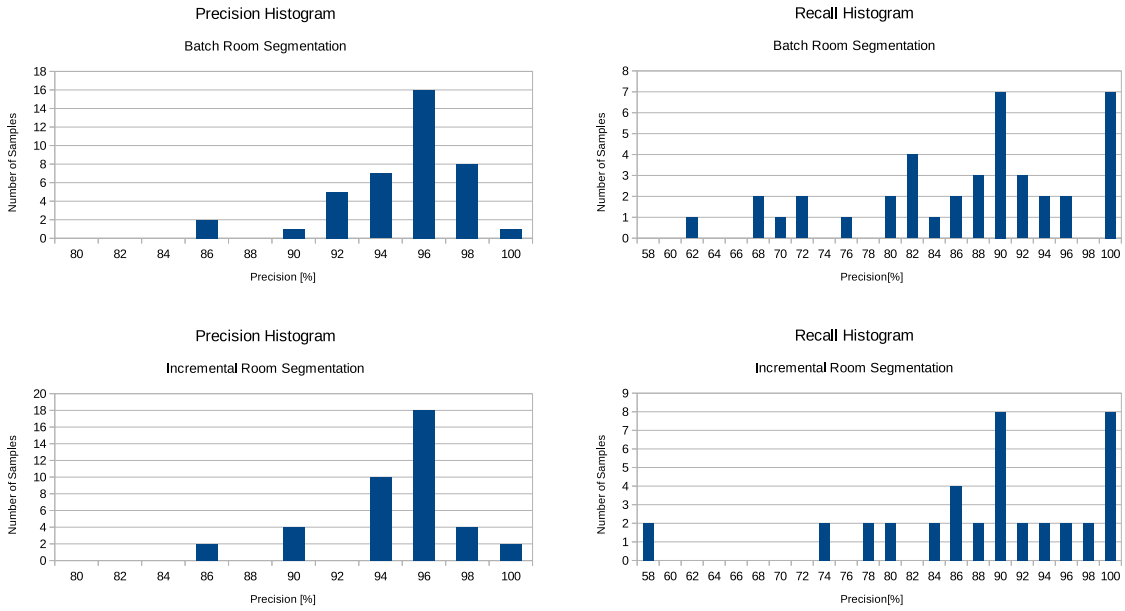

Fig. 2. Precision and Recall Histograms, In spite the diversity of the maps the precision is very localized, in both cases 95% of the maps scores precisions above 94%. The effect of the heterogeneity of the maps in terms of complexity and size affects importantly the recall, we score recalls above 84% in the batch mode and 86% in the incremental mode for the 85% of the maps

		Voronoi	Batch	Incremental
No Furniture	recall	95.0 ± 2.3	86.3 ± 9.7	87.5 ± 10.7
	precision	94.8 ± 5.0	94.1 ± 3.1	93.8 ± 3.1
Furnished	recall	86.6 ± 5.2	85.8 ± 10.3	87.5 ± 10.7
	precision	94.5 ± 5.1	94.0 ± 3.2	93.8 ± 3.1

TABLE I. AVERAGED PRECISION AND RECALL

The diversity on the size of the images in the dataset implies the time needed to process them is not consistent (the time reported for the Voronoi-based segmentation is $12.0 \pm 14.2s$). To evaluate the processing time we use the histograms in fig.3. The batch version of the contour-based segmentation is one order of magnitude faster than the Voronoi-based. Our incremental approach further reduces these processing time to tens of miliseconds.

Summing up, the proposed Contour-Based Segmentation algorithm obtains similar quality to the state of the art approaches, one order of magnitude faster, without any application oriented heuristics. Our incremental algorithm further reduces these times, without sacrificing quality.

Qualitatively, the decomposition is similar to the human labeled, with the difference that corridors are usually over segmented (Fig. 4), this is because this algorithm is based on the convexity of the regions, one "L" shaped corridor is always segmented (as a minimum) in two regions.

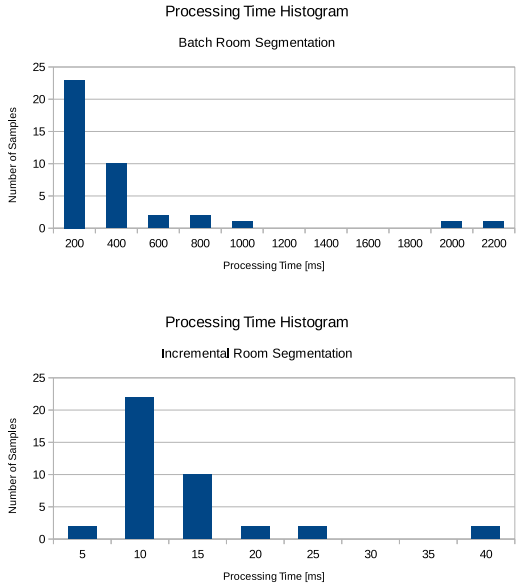

Fig. 3. Processing Times, Using contour based segmentation 95% of the maps in the dataset are processed under 1s, with our incremental version the processing time is below 25 ms for the 95% of the maps

B. Environment with Low Structure

The lack of ground truth for segmentation in poorly structured scenarios implies the evaluation of the quality can only be made using parameters being affected by it. The number of regions can indicate if a region is over segmented (value too high) or under segmented (value too low).

As expected, the average time for the incremental version is lower than the batch version (Table II). On the other hand, the histogram of the processing time shows an interesting

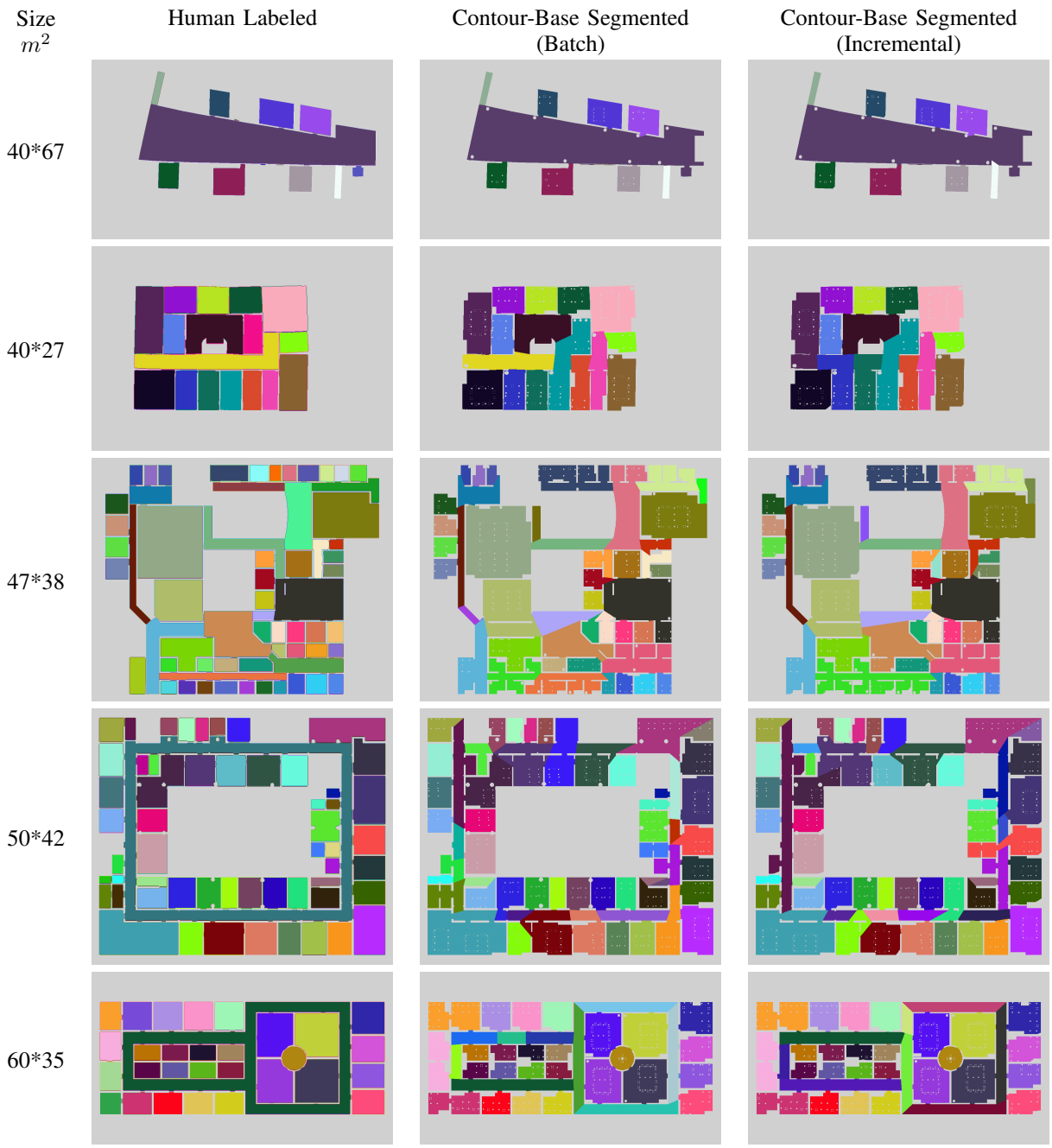

Fig. 4. Some results on the datasets in [3], Although the corridors are over segmented, the rooms are usually well differentiated

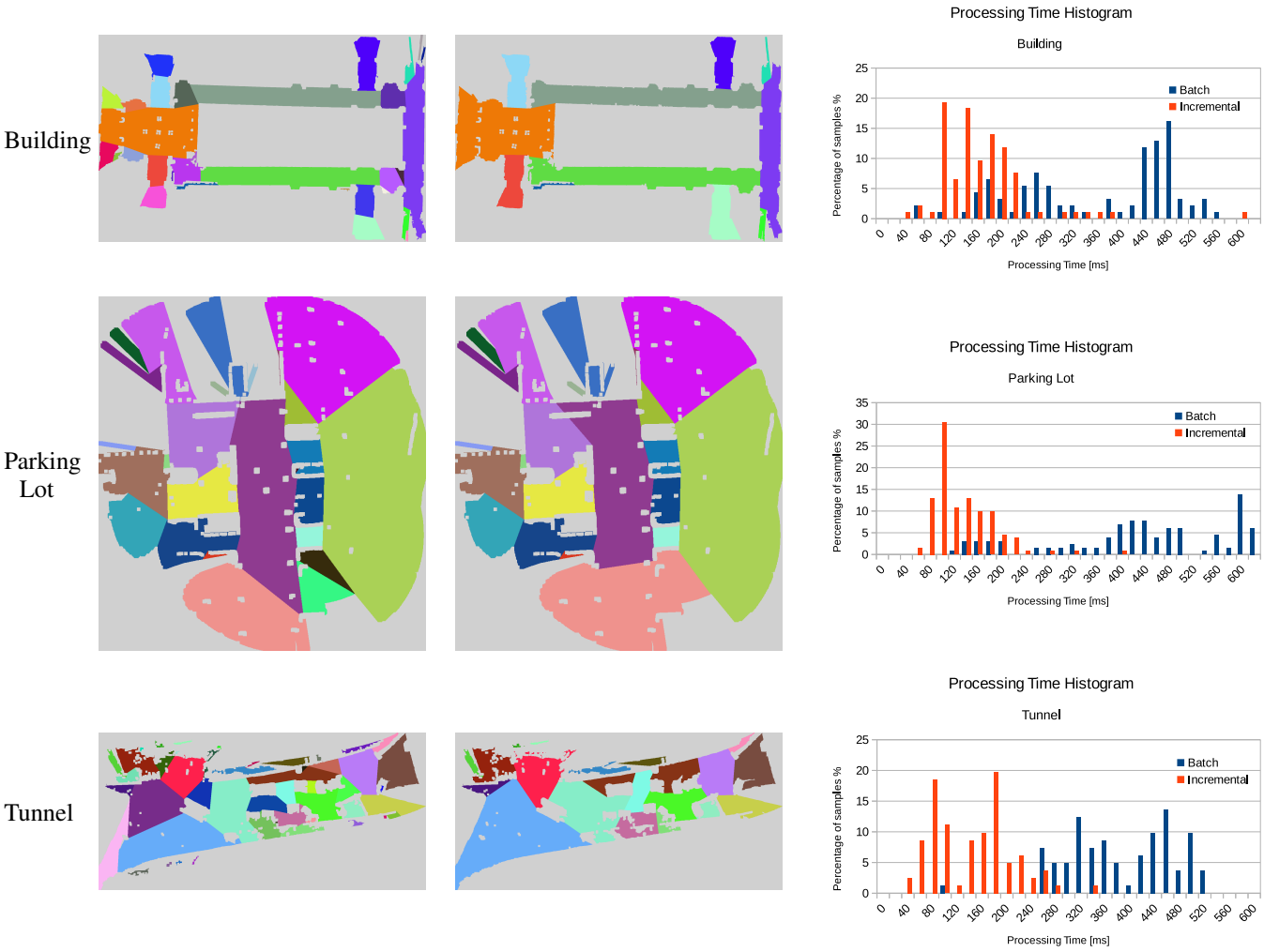

Fig. 5. Exploration sequences in [6] are processed with the incremental and the batch version of the algorithm. Both methods produce visually appealing segmentations. The incremental version tends to produce a number of regions slightly lower than the incremental counterpart. The histograms show the processing time of our incremental algorithm is under 200ms for the 95% of the frames processed, on the other hand in the batch version the time to process increases with every frame producing and spread distribution

behavior (fig 5): in the case of the incremental version, the distribution is almost completely localized independent on the map resolution, but the batch version grows monotonically spreading through all X axis.

Figure 5 shows that when the number of regions are similar (table II), both decompositions are similar. Additionally it can be observed that the number of regions produced by the incremental algorithm is consistently smaller in this dataset. This seems to indicate a tendency to under segment environments with low structure, compared to the batch version.

Scenario	Resolution [m/pixel]	Processing Time [ms] Batch	Processing Time [ms] Incremental	Number of Regions Batch	Number of Regions Incremental
Parking Lot	0.050	427 ± 156	104 ± 51	16	12
Building	0.025	331 ± 131	137 ± 76	13	11
Tunnel	0.025	376 ± 85	135 ± 63	21	20

TABLE II. SEGMENTATION IN LOW STRUCTURE ENVIRONMENTS

VI. CONCLUSION

We introduce a new contour-based approach for topological segmentation. This segmentation has the following advantages over traditional approaches:

1) Velocity: batch times are comparable to the simplest segmentation cases, with quality comparable to the most complicated cases; an incremental version allows on-line use.
2) Grid Size Independence: contour based decomposition depends only on the contour characteristics, not in the image size.

3) Quality of segmentation: the resulting segmentation scores high in precision and recall in the comparison with human labeling.
4) Flexibility: it works in environments ranging from highly to poorly structured, with comparable results to human labeling.
5) One parameter decomposition: the only parameter in the algorithm is the maximum concavity ($Max\_Concavity$) allowed, it is measured in meters. Empirically, the same value ($2.5m$) is use for highly and poorly structured environments, this hints this parameter could be use as a constant for map segmentation.

Many 2D mapping algorithms provide an occupancy grid. In these cases, this grid has to be transformed into an image and then into contours.

VII. Future Work

This segmentation approach is inspired by the "Part Segmentation" of the 3D meshes. Thus, aapplying this segmentation to maps resulting of 3D SLAM systems is a natural extension of this work.

The topological segmentation is commonly the first step to attempt higher level tasks. In particular, the incremental formulation is built as one part of the autonomous robot exploration pipeline, currently under development.

References

[1] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, 2012.

[2] Olaf Booij, Bas Terwijn, Zoran Zivkovic, and B Krose. Navigation using an appearance based topological map. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3927–3932. IEEE, 2007.

[3] Richard Bormann, Florian Jordan, Wenzhe Li, Joshua Hampp, et al. Room segmentation: Survey, implementation, and analysis. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1019–1026. IEEE, 2016.

[4] Jyh-Ming Lien and Nancy M. Amato. Approximate convex decomposition of polygons. *Computational Geometry*, 35(12):100 – 123, 2006. Special Issue on the 20th {ACM} Symposium on Computational Geometry20th {ACM} Symposium on Computational Geometry.

[5] Guilin Liu, Zhonghua Xi, and Jyh-Ming Lien. Dual-space decomposition of 2d complex shapes. In *27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, Jun. 2014. IEEE.

[6] Ming Liu, Francis Colas, Luc Oth, and Roland Siegwart. Incremental topological segmentation for semi-structured environments using discretized gvg. *Autonomous Robots*, 38(2):143–160, 2015.

[7] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[8] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[9] Jovisa Zunic and Paul L Rosin. A new convexity measure for polygons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7):923–934, 2004.