

New Message Type for Instance Segmentation Results #5047

 Closed

Unanswered

StepTurtle asked this question in Q&A






StepTurtle on Jul 30

Collaborator

Hi,

We want to add a new 2D instance segmentation model to Autoware perception pipeline and I have a question related this topic. We couldn't decide on the message type.

Check following PRs and issue to see current works:

- Issue:  [Implement RTMDet to Perception Pipeline](#) [autoware.universe#7235](#)
- Model PR: [WIP]  [feat\(autoware_tensorrt_rtmddet\): add tensorrt rtmddet model](#) [autoware.universe#8165](#)
- instance segmentation fusion PR: [WIP]  [feat\(image_projection_based_fusion\): add instance segmentation fusion](#) [autoware.universe#8167](#)

In current Autoware design we are using following message types for 2D detection:

For only bounding box detection:

- `tier4_perception_msgs::msg::DetectedObjectWithFeature` ([under tier4_autoware_msgs repository](#))

For bounding box and semantic segmentation result:

- `tier4_perception_msgs::msg::DetectedObjectWithFeature` ([under tier4_autoware_msgs repository](#)) + `sensor_msgs::msg::Image`

For semantic segmentation models, we have a single image representing all objects in the scene, so we can publish the segmentation mask as a single image. However, for instance segmentation models, we have a separate segmentation mask for each detected object, and we need to determine how to store this mask information.

Possible Approaches:

We can publish the instance information in two different ways.

- 1) A `sensor_msgs::msg::Image` for each detected object.

Category



Q&A

Labels

[component:percept...](#)

5 participants



We can add a new message type something similar with

`tier4_perception_msgs::msg::DetectedObjectWithFeature` and it contains
`sensor_msgs::Image`

Current: `tier4_perception_msgs::msg::DetectedObjectWithFeature` :

```
autoware_perception_msgs/DetectedObject object
Feature feature
```



Possible Approach:

```
autoware_perception_msgs/DetectedObject object
Feature feature
+ sensor_msgs/Image
```



- It can be named `tier4_perception_msgs::msg::DetectedObjectWithMask` .
- Should we store under [autoware_msgs](#) or [tier4_autoware_msgs](#)
- It wastes a little bit extra time to copy images and create a `sensor_msgs::Image` for each detected object.

2) A Border Arrays

We can add a new fields to

`tier4_perception_msgs::msg::DetectedObjectWithFeature` which contains
borders of the detected object

Current: `tier4_perception_msgs::msg::DetectedObjectWithFeature` :

```
autoware_perception_msgs/DetectedObject object
Feature feature
```



Possible Approach:

`tier4_perception_msgs::msg::DetectedObjectWithBorder`

```
autoware_perception_msgs/DetectedObject object
Feature feature
+ Border[] borders
```



`tier4_perception_msgs::msg::Border`

```
+ uint8 EXCLUDED=0
+ uint8 CONTAINED=1

+ bool boundary_status
+ geometry_msgs/Point[] border
```

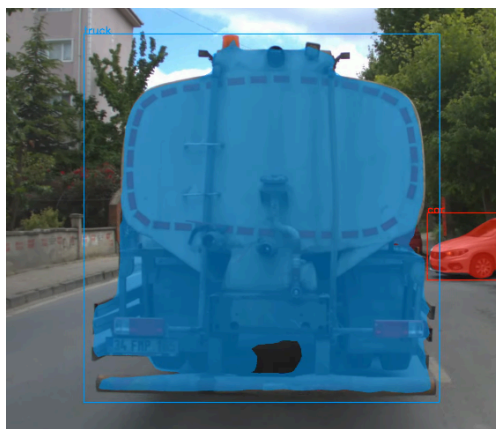


- We need to add two message type and they can be named `tier4_perception_msgs::msg::DetectedObjectWithBorder` and `tier4_perception_msgs::msg::Border` .

- Should we store under `autoware_msgs` or `tier4_autoware_msgs`
- It will probably be faster to use, but we may need to perform more complex operations while finding borders and decoding borders.
- Why we need border array for single detected object? Because sometimes we might encounter objects made up of multiple parts.



- Why we need `boundary_status` ? Because sometimes we might encounter objects with gaps inside them.



If you have any opinions or another approach similar to or completely different from the information I shared above on this topic, I would be glad to hear them.

↑ 1

2 comments · 8 replies

Oldest

Newest

Top



mitsudome-r on Aug 13

Maintainer

@YoshiRi @kminoda Do you have any comments about the message type?

↑ 1

👁 1

7 replies



Show 2 previous replies



badai-nguyen on Aug 13

Collaborator

@StepTurtle Thank you for your explanation.

- I think current `camera_lidar_fusion` pipeline is designed to enhance the detection of known objects (by using `roi_cluster_fusion` node) alongside with DNN 3D detection model (such as `centerpoint`) while ensuring the safety by retaining `unknown_obstacle_objects` in output.
- Recently, the segmentation based filter is added, mainly aimed to ignore some low-risk obstacle unknown objects (such as unknown caused by bushes)
- From my understanding of your proposed pipeline and result of instance segmentation model, it primarily focuses on improving the known objects detection. Is my understanding correct? If `yes` , do you have any strategy to retain the unknown obstacle objects?



StepTurtle on Aug 13 Collaborator Author

@badai-nguyen

Yes, you are right. We believe that we can improve the detection of known objects. However, I don't have a solution to help retain the unknown obstacle objects.

In addition to these, if we don't find the use of instance segmentation in this way beneficial or if we can't use it this way, do you have any ideas on other areas where we could use it? Or do you think adding the instance segmentation model would be entirely useless?



badai-nguyen on Aug 13 Collaborator

@StepTurtle Thank you for your confirmation. I understand your idea now.

- I believe that even if this only improves known detection, it would still be valuable to integrate it with Autoware and make it work. It might useful for some usecases. From my experience, this can help avoid incorrect fusions on objects with large occlusions, which we sometimes encounter by using `roi_cluster_fusion` method.
- Of course, it is important to design the pipeline carefully to avoid degrading the performance of unknown detection.

Please allow us some time for internal discussion.

By the way, as it is summer holiday in Japan currently, so it might take time to get response from others.



StepTurtle on Aug 13 Collaborator Author

@badai-nguyen Thanks your response.

Let me take a closer look at the points you mentioned in the first article (`roi_cluster_fusion`). I'll let you know if I come up with an idea.

btw I am sorry, last week, Yamato Ando-san mentioned they were on holiday, but I just remembered when you brought it up. Feel free to get back to me whenever you want. Have a nice holiday!



armaganarsIn on Aug 19 Collaborator

@StepTurtle Thank you for your explanation.

- I think current `camera_lidar_fusion` pipeline is designed to enhance the detection of known objects (by using `roi_cluster_fusion_node`) alongside with DNN 3D detection model (such as `centerpoint`) while ensuring the safety by retaining `unknown_obstacle_objects` in output.
- Recently, the segmentation based filter is added, mainly aimed to ignore some low-risk obstacle unknown objects (such as unknown caused by bushes)
- From my understanding of your proposed pipeline and result of instance segmentation model, it primarily focuses on improving the known objects detection. Is my understanding correct? If `yes`, do you have any strategy to retain the unknown obstacle objects?

Why you are suggesting to use `roi_cluster_fusion` instead of https://autowarefoundation.github.io/autoware.universe/pr-6707/perception/autoware_image_projection_based_fusion/docs/segmentation-pointcloud-fusion/? isn't this node created for these purposes anyway? I didn't understand your comment on this topic?



YoshiRi on Aug 14 Collaborator

@StepTurtle

Thank you for the intriguing proposal. As Dai-san mentioned, the use of segmentation is also being considered at TIER IV.

comment and suggestions

Regarding the message type, I'd like to comment that in the proposed approach, a mask of the same size as the image would be sent as a topic for each class or object. This could lead to very inefficient communication. To minimize the number of topics related to images, consider issuing pairs of images and configuration (config) data.

For instance, you could assign scalar values to each class based on the image intensity. As for the correspondence between these scalar values and classes, you might use a field or a separate latch topic.

Here's an example of how you could define such messages. However, for the config part, you could flow it in a different format, such as a string in a separate latch topic, while still achieving the desired functionality:

```
Config config # Config should represent a map between class and
image intensity
```



```
Image image # Mask image
```

Concerns

- For semantic segmentation, handling classes alone is sufficient, and grayscale might be enough. However, for instance segmentation, where you need to associate UUID text with pixels, using RGB seems necessary to handle more than 256 objects.
- Additionally, considering tasks like panoptic segmentation, where both class and UUID need to be handled simultaneously, or dealing with more complex information than probabilities, it might be better to avoid hastily defining a new type and instead leverage existing types for effective information exchange.

Supplement: Proposed improvement

If the Image topic's encoding allowed specifying boolean values, it could significantly mitigate the data size issue. However, based on my review of the [ROS source code](#), it appears that boolean masks are not supported. In that case, consider creating a specialized type capable of storing boolean mask images and including it in `DetectedObjectsWithFeature`.

↑ 3

👁 1

1 reply



StepTurtle on Aug 22

Collaborator

Author

edited ▾

Really thanks for your suggestions. [@YoshiRi](#)

As your proposal, we can create something like that:

Config.msg

```
uint8 UNKNOWN=0
uint8 CAR=1
uint8 TRUCK=2
uint8 BUS=3
uint8 BICYCLE=4
uint8 MOTORBIKE=5
uint8 PEDESTRIAN=6
uint8 ANIMAL=7
```



```
# It represent map between instance segmentation mask and
classes
# The pixel intensities in mask's starts with 1 (pixel
value 0 represent pixel is not belong any object)
#   `class_array[0]` returns the `class_id` for pixels
with an intensity value of `1` in the mask.
#   `class_array[1]` returns the `class_id` for pixels
with an intensity value of `2` in the mask.
#   ...
# The relationship between pixels and class IDs continues
```

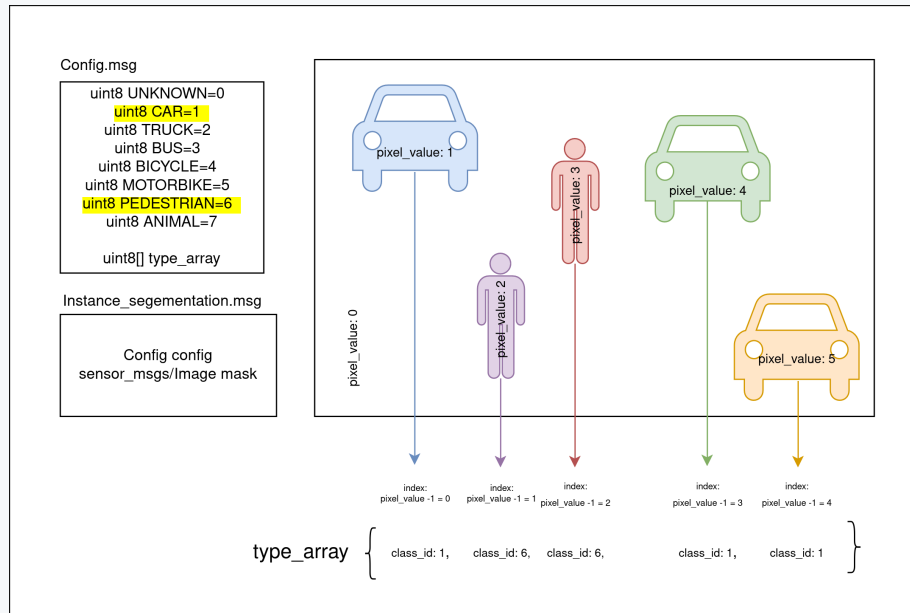
in this way.
uint8[] class_array

Segmentation.msg

Config config # Stores relation between class and image intensity



Image image # Mask image



If pixel value is 0, the pixel is not belong to any object. If pixel value is X, the pixel is belong an object and object class is (X-1). index of `class_array`.

Additionally, considering tasks like panoptic segmentation, where both class and UUID need to be handled simultaneously, or dealing with more complex information than probabilities, it might be better to avoid hastily defining a new type and instead leverage existing types for effective information exchange.

On your concern, what do you think merging the all segmentation methods under the message type you suggest. Think we have two message as above and one of them is `Segmentation.msg` and other one is `SegmentationConfig.msg`. I think these messages fulfill semantic, panoptic and instance segmentation needs. Perhaps, if it's normally used to define a class for semantic segmentation, we can't use a pixel intensity of 0 because other ones needs pixel intensity for background.

For semantic segmentation, handling classes alone is sufficient, and grayscale might be enough. However, for instance segmentation, where you need to associate UUID text with pixels, using RGB seems necessary to handle more than 256 objects.

Also, the pre-trained model of RTMDet which we use have a 100 detection limitation for each frame. So, it looks, for now grayscale should be enough when object number is less than 256.

In the end, we have a single image and it could be even gray scale. So, I guess it could avoid inefficient communication.

Regarding where to place the message type, I don't have a great idea, but I guess I'll put it inside `autoware_internal_messages`.

I'd love to hear any other suggestions or ideas you may have in the future.

