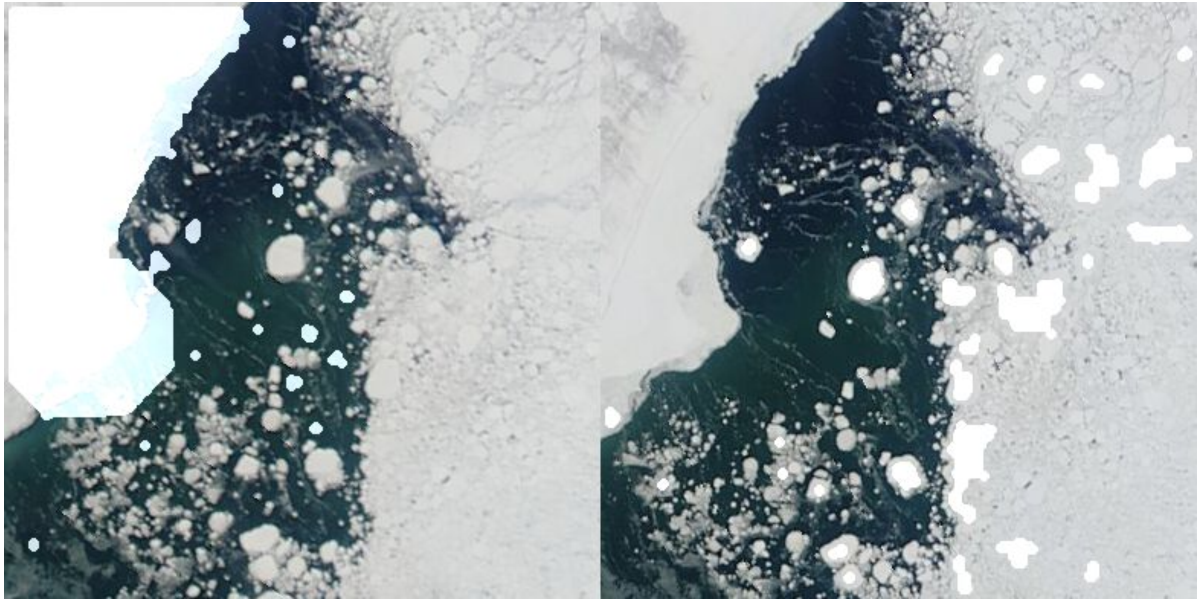
Identifying causes of error in Segmentation F

Daniel Watkins, Oct. 9, 2024

Problem: A persistent error in which the large portions of land-masked regions are labeled as sea ice floes in the final segmentation

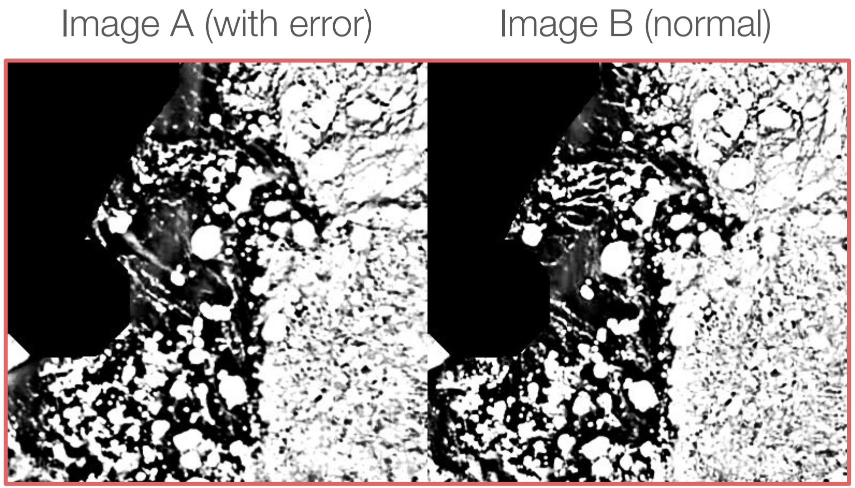


Example of error: test_images/298-hudson_bay. Two images from the same day with similar properties; in one, the land mask is labeled as a floe, and no true floes are detected.

Tracing the error to the *k*-means step in Segmentation F

After running the preprocessing and segmentation up to segmentation B, I stepped through the processing steps in segmentation F one-by-one.

1. Input from segmentation B



Tracing the error to the *k*-means step in Segmentation F

After running the preprocessing and segmentation up to segmentation B, I stepped through the processing steps in segmentation F one-by-one.

1. Input from segmentation B
2. Identification of leads (intersection of watershed output and application of area opening)

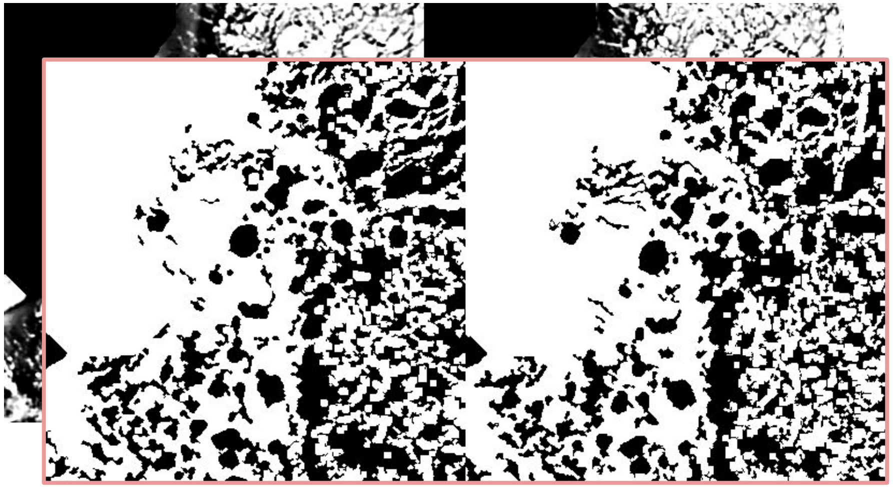


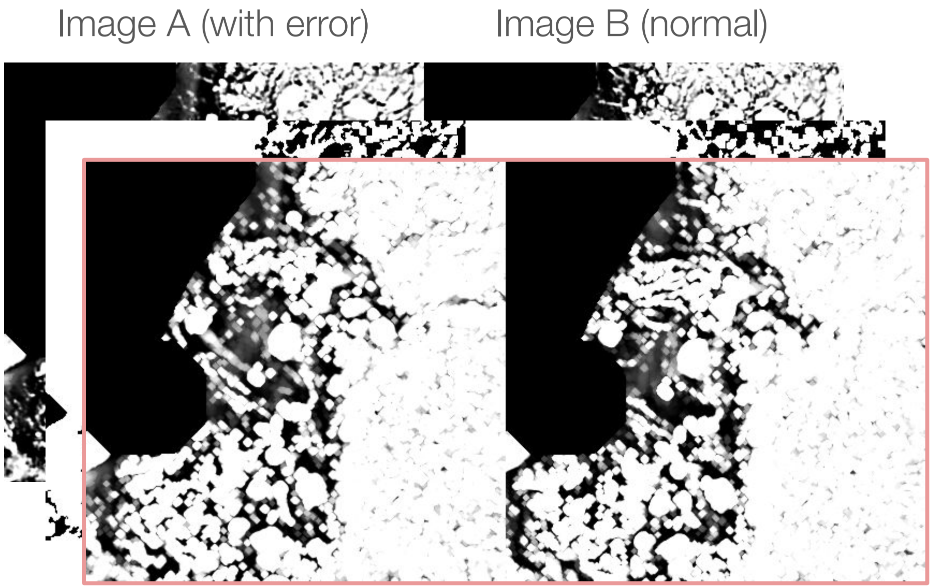
Image A (with error)

Image B (normal)

Tracing the error to the *k*-means step in Segmentation F

After running the preprocessing and segmentation up to segmentation B, I stepped through the processing steps in segmentation F one-by-one.

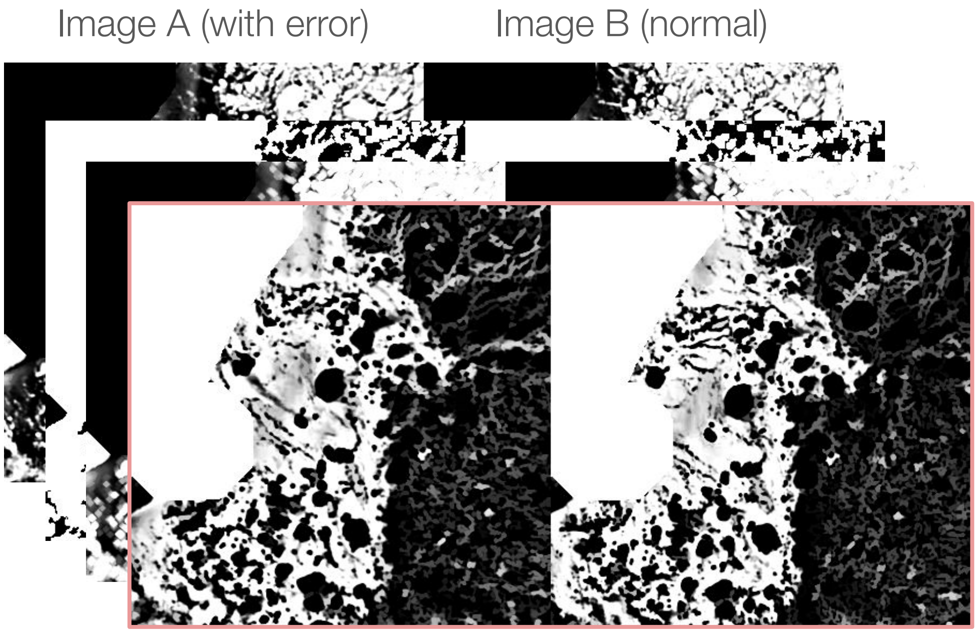
1. Input from segmentation B
2. Identification of leads (intersection of watershed output and application of area opening)
3. Dilation of "not ice" object



Tracing the error to the *k*-means step in Segmentation F

After running the preprocessing and segmentation up to segmentation B, I stepped through the processing steps in segmentation F one-by-one.

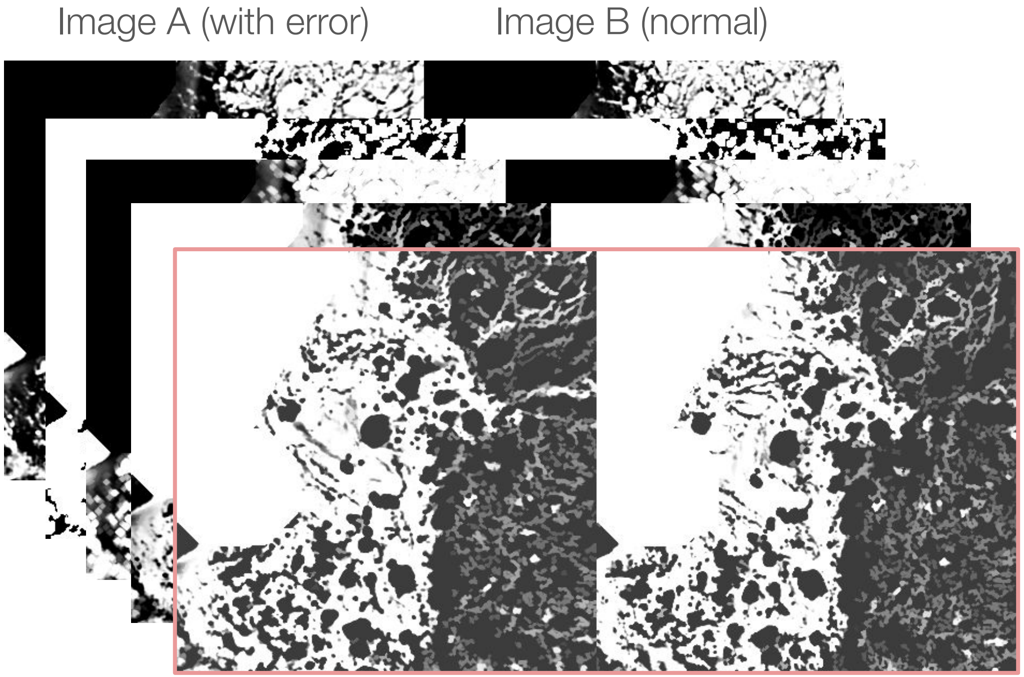
1. Input from segmentation B
2. Identification of leads (intersection of watershed output and application of area opening)
3. Dilation of “not ice” object
4. Reconstruction of “not ice”



Tracing the error to the *k*-means step in Segmentation F

After running the preprocessing and segmentation up to segmentation B, I stepped through the processing steps in segmentation F one-by-one.

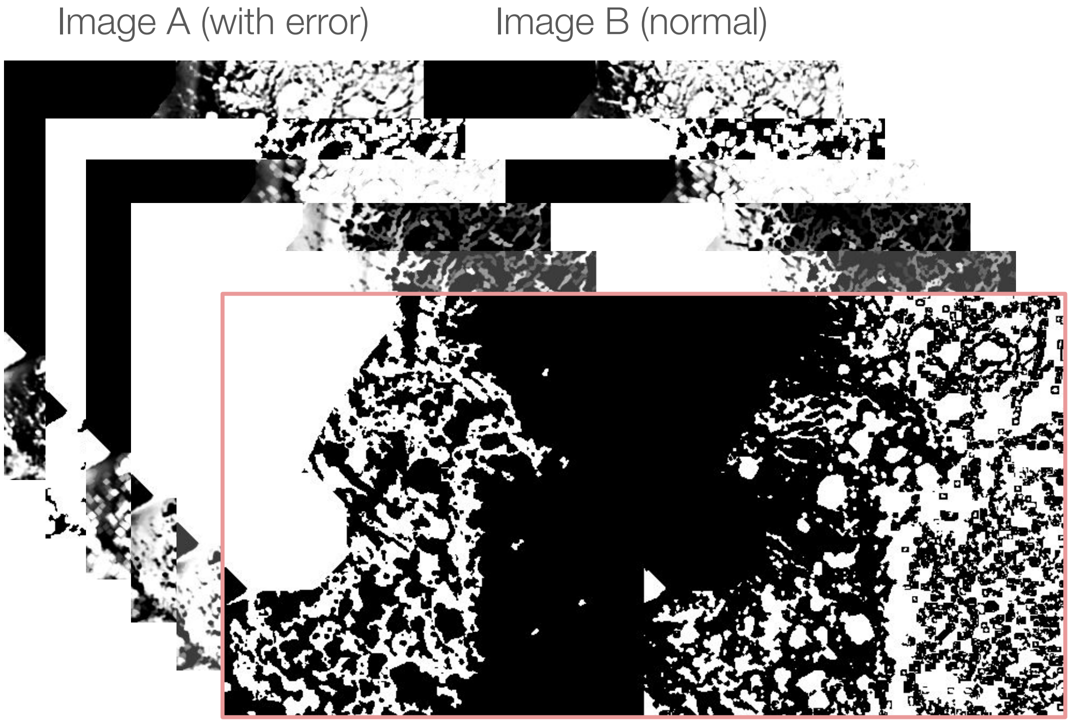
1. Input from segmentation B
2. Identification of leads (intersection of watershed output and application of area opening)
3. Dilation of "not ice" object
4. Reconstruction of "not ice"
5. Product of "not ice", "ice lead"



Tracing the error to the *k*-means step in Segmentation F

After running the preprocessing and segmentation up to segmentation B, I stepped through the processing steps in segmentation F one-by-one.

1. Input from segmentation B
2. Identification of leads (intersection of watershed output and application of area opening)
3. Dilation of “not ice” object
4. Reconstruction of “not ice”
5. Product of “not ice”, “ice lead”
6. Application of k-means



Diagnosing issue in the *k*-means step

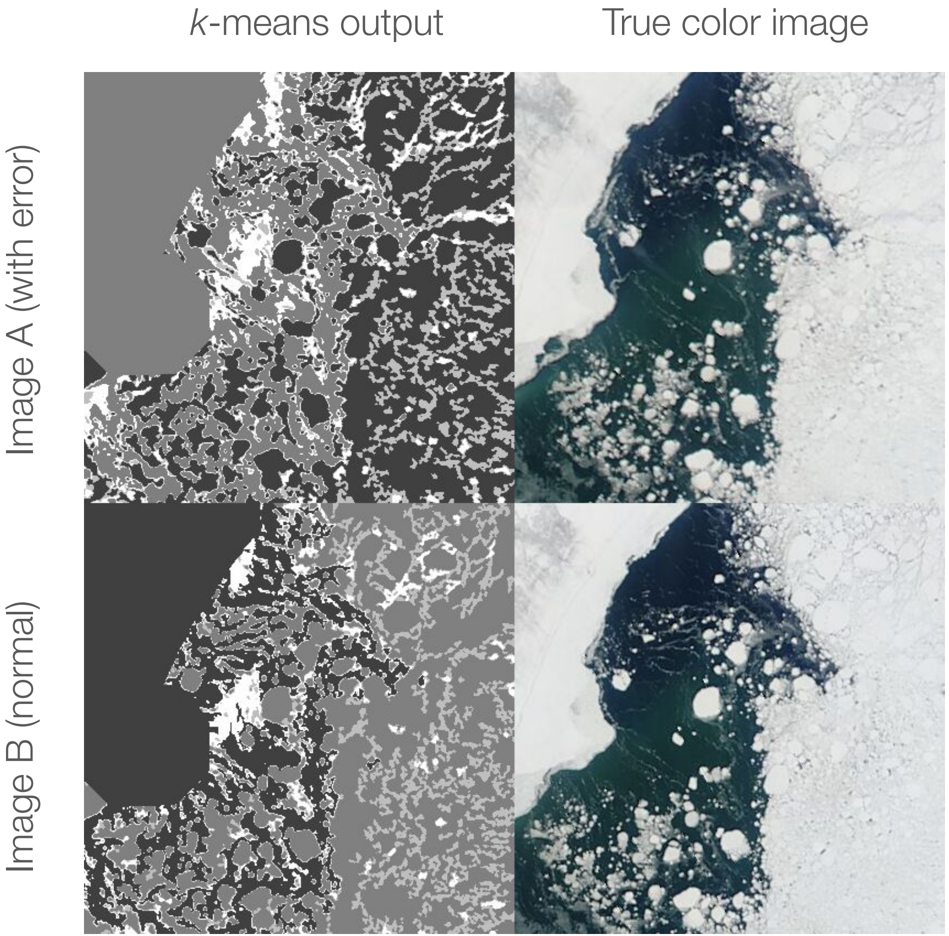
The *k*-means algorithm divides a grayscale image into *k* clusters. Currently *k* is set to 4. The function does the following:

```
gray_image ← reconstructed_leads
classes ← kmeans(gray_image, k=4)
ice_class ← mode( classes[ice_labels] )
segmented_image ← reshape(classes)
return segmented_image == ice_class
```

Diagnosing issue in the *k*-means step

The *k*-means algorithm divides a grayscale image into *k* clusters. Currently *k* is set to 4. The function does the following:

```
gray_image ← reconstructed_leads
classes ← kmeans(gray_image, k=4)
ice_class ← mode( classes[ice_labels] )
segmented_image ← reshape(classes)
return segmented_image == ice_class
```



Diagnosing issue in the *k*-means step

The *k*-means algorithm divides a grayscale image into *k* clusters. Currently *k* is set to 4. The function does the following:

```
gray_image ← reconstructed_leads
classes ← kmeans(gray_image, k=4)
ice_class ← mode( classes[ice_labels] )
segmented_image ← reshape(classes)
return segmented_image == ice_class
```

The result of the *k*-means algorithm looks perfectly reasonable in both cases. My suspicion: we need to improve the accuracy of the ice label step. From what I can see, the error is actually in ice_labels – it labels some sections of the landmask as ice there.

Next steps

- Validate the find_ice_labels function
- Verify that framework is robust to chance that no ice_labels are found