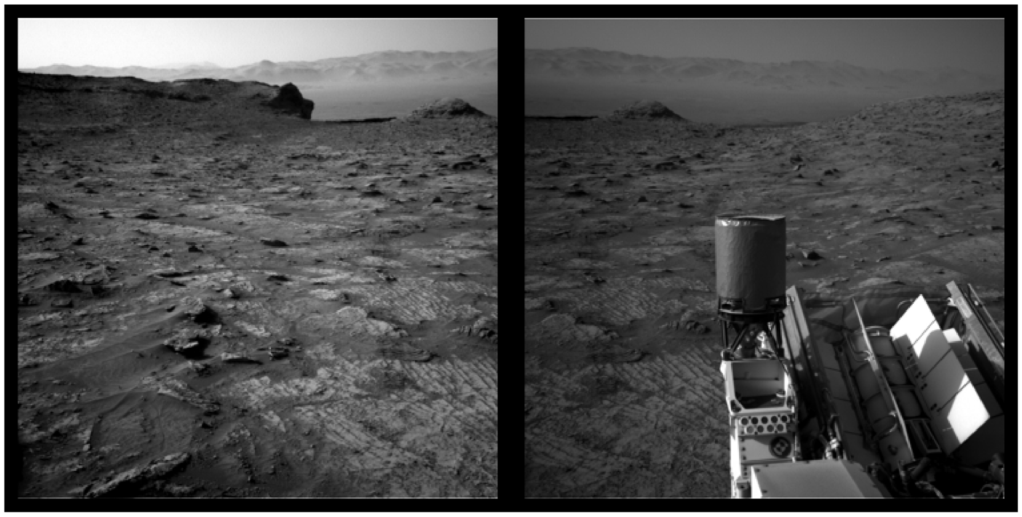
Mars Rover: Final Project

Follow the instructions provided in the course to complete your final project.

Load the images

```
fixedImg = imread("sol_03333_opgs_edr_ncam_NLB_693387385EDR_F0921230NCAM00259M_.JPG");
movingImg = imread("sol_03333_opgs_edr_ncam_NLB_693387301EDR_F0921230NCAM00259M_.JPG");

montage({movingImg, fixedImg},"BorderSize",[10 10])
```



Register the Images

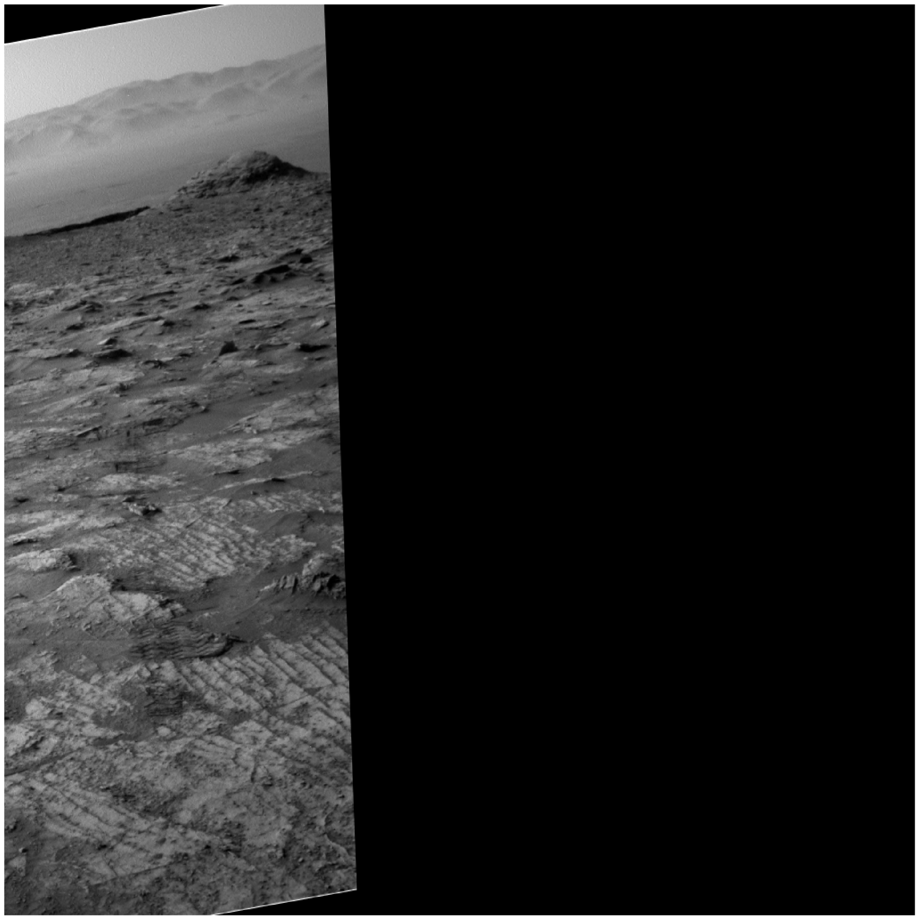
Use the Registration Estimator app to create a function that registers the images. Then, use that function to return the registration structure.

When you're happy with your registration function, return to the course and use the online grader to check if it gives the correct result.

```
reg = registerMarsImages(movingImg, fixedImg)
```

```
reg = struct with fields:
     FixedMatchedFeatures: [244×1 SURFPoints]
    MovingMatchedFeatures: [244×1 SURFPoints]
           Transformation: [1×1 affine2d]
          RegisteredImage: [1024×1024 uint8]
            SpatialRefObj: [1×1 imref2d]
```

```
figure; imshow(reg.RegisteredImage)
```



Initialize the Panorama

Use your geometric transformation to determine the world coordinates and initialize an empty panorama image. Here, recall that you're working with **grayscale** images, unlike the color concrete images.

```
n = 2;
tforms(2) = projective2d(eye(3));
ImageSize = zeros(n,2);
```

asdf

```
tforms(n) = estimateGeometricTransform2D(reg.MovingMatchedFeatures, reg.FixedMatchedFeatures,..
    "projective",'Confidence',99.9,'MaxNumTrials',100)
```

```
tforms = 1×2 projective2d
```

	1	2
1	1×1 projective2d	1×1 projective2d

```
tforms(n).T = tforms(n).T * tforms(n-1).T;
```

When you're done, r

```
sizeX = width(fixedImg)
```

```
sizeX = 1024
```

```
sizeY = height(fixedImg)
```

```
sizeY = 1024
```

```
for i = 1:numel(tforms)
    [xlim(i,:),ylim(i,:)] = outputLimits(tforms(i), [1 sizeX], [1 sizeY]);
end

%find max and min
xMin = min(xlim(:));
xMax = max(xlim(:));

yMin = min(ylim(:));
yMax = max(ylim(:));
```

```
%width and height of panaroma
imgWidth = round(xMax - xMin);
imgHeight = round(yMax - yMin);
```

asdf

```
% Initialize the "empty" panorama.
panorama = zeros([imgHeight, imgWidth, 1],"uint8");
```

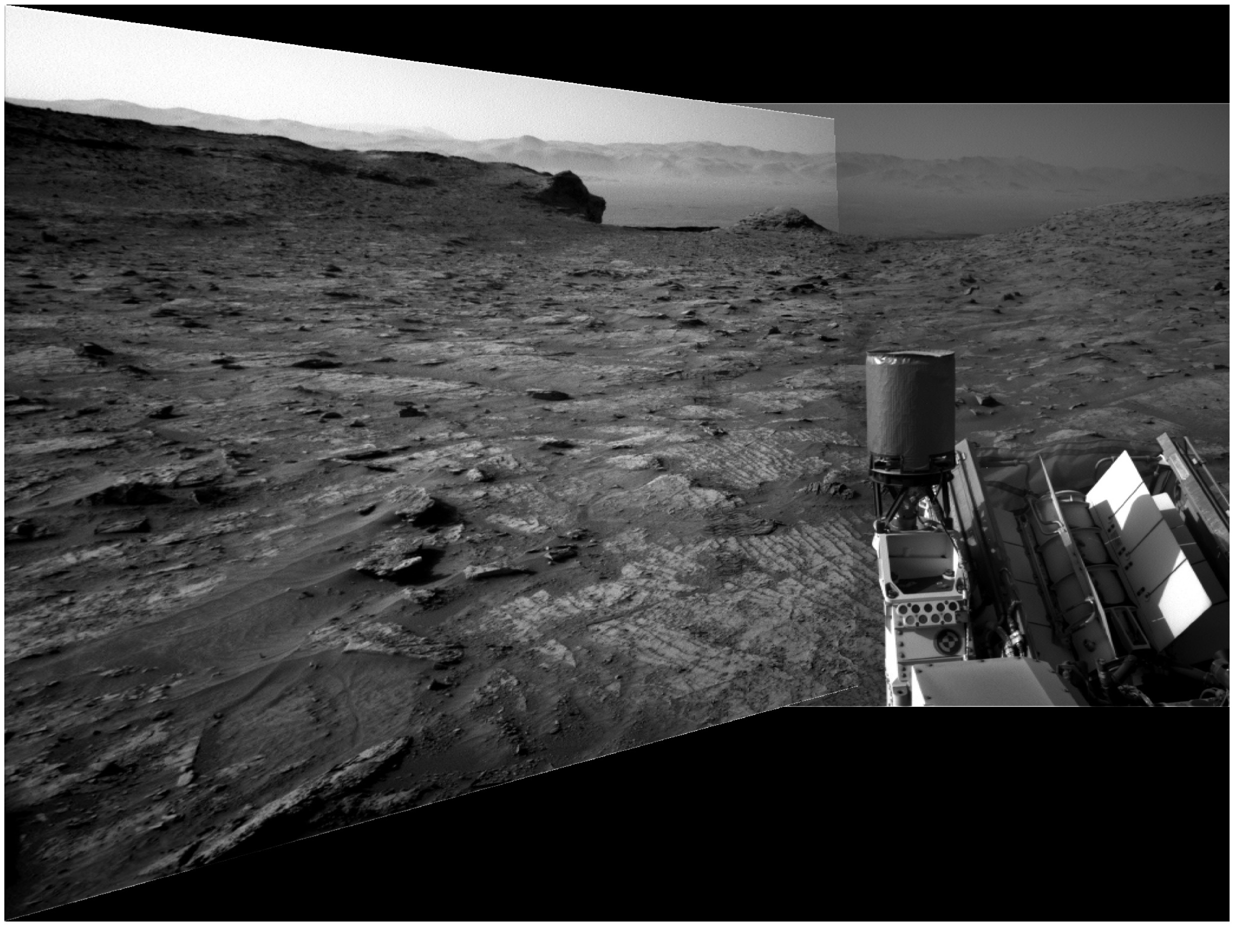
(eturn to t

```
blender = vision.AlphaBlender("Operation","Binary mask","MaskSource","Input port");

% Create a 2-D spatial reference object defining the size of the panorama
panoramaView = imref2d([imgHeight imgWidth],[xMin xMax],[yMin yMax]);
```

he course and

```
i =1;
img = fixedImg;

warpedImage = imwarp(img, tforms(1),"OutputView",panoramaView);
mask = imwarp(true(size(img,1)),tforms(1),"OutputView",panoramaView);
panorama = step(blender,panorama,warpedImage, mask);
i =2;
img = movingImg;
```

```
warpedImage = imwarp(img, tforms(2),"OutputView",panoramaView);
mask = imwarp(true(size(img,2)),tforms(2),"OutputView",panoramaView);
panorama = step(blender,panorama,warpedImage, mask);
imshow(panorama)
```



take the quiz to check if your panorama dimensions are as expected.

Create the Panorama

Use the `vision.AlphaBlender` object to create the final panorama. When you're done, pass your final image to the `testMarsImage` function to see if it's correct.

```
testMarsImage(panorama)
```

If you're correct, go back to the quiz and add the numeric code to complete the course!

If you get a dimension error while using `AlphaBlender`, double check that you are initializing the empty panorama correctly for a grayscale image. Remember that the third dimension of a color image has size 3, but the third dimension of a grayscale image only has size 1.

```
testMarsImage(panorama)
```

```
Success!
Enter the number: 1984 into the final quiz question.
```