############# PERFECT MATCH

\* Why did you choose that given metric?

- We choose normalized metrics to make thresholding easier since the output image values is in between 0 to 1.

- Tweaking the parameters all the normalized metrics works well for detecting the matching objects.

\* How robust to false positives/negatives is your selected metric.

- To prevent multiple detection for the same object, we have implemented some features in the ‘min\_max’ function that checks that each detection is far enough to each other.

\* Is the number of output locations the same as the matching objects?

- Yes, with all normalized metrics, it counts the exactly the same number of matching objects.

############# NOT SO PERFECT MATCH

\* How did you select the number of maximums/minimus?

We choose the lower threshold values than the previous exercise to detect deformed objects.

\* How robust to false positives/negatives is your selected metric.

We have chosen correlated method to make the detection robust. Our method only detected the right object especially for the template 1, we only found the hands up aliens. If we use the MSE method, the difference between the hands up aliens and hands down aliens would not be big enough and thus, leading the false negatives.

\* Is the number of output locations the same as the matching objects?

Yes it is. We found 5 green objects and 12 hands up aliens.

\* Could you use any of the features from the last chapter to improve the matching?

We may use the histogram equalization to detect the value of the big object and then filter the object that has the same color. It could make detecting easier and robust allowing us to reduce the threshold more.

############# NOT SO PERFECT MATCH

\* What metric seemed to work better this time?

Normalized cross correlation works better than the other metrics since we have exactly the same template that matches from the original image.

\* Was it different from the previous exercise?

Thresholding values were different from the previous exercise but other part of the algorithm stays the same

\* Could you use any of the features from the last chapter to improve the matching?

\* How could you reduce the false positives?

You can detect t