Using GANs for Outpainting of Objects in a Scene

I. Abstract

Images can contain objects that compromise the image's quality, obstruct the central focus of the image, or reveal private or confidential information. Removal of specific objects from an image requires precise outlining and elimination of the selected object's image data and a convincing image pattern to conceal the deleted data. Manually achieving this via photo editing software can be imprecise, inefficient, and inaccurate mainly due to limitations in interpolating the blanked image data. In response to this, we propose a network pipeline that conjoins the object detection, categorization, and segmentation of Mask R-CNN (MRCNN) [1] and the automated pattern generation of the Generative Image Inpainting network (GIIN) [2]. We implement several modifications to both network aimed to allow the output of MRCNN to be usable as input into GIIN. Due to the lack of ground truth images that demonstrate a mapping from an image to its counterpart with specific objects removed, we contemplate the success of this network mostly qualitatively. This is due to the fact that GIIN is a Generative Adversarial Network (GAN), whose nature as a combination of a generator and discriminator network makes it hard to apply traditional measures like PSNR and RMSE [3]. We will quantify some of the limitations of the two networks occasionally resulting in outputs that contain residual information of removed objects.

II. Problem Statement

Scenes can contain objects that may lower the quality of the image or obstruct the purpose of the image. Some examples of this include nature photography containing manmade

objects (e.g. vehicles, power lines, people, etc.), people/objects in the foreground or background of an image which ultimately clutter the focus of the scene, or photos of people who wish to not be in the photo due to privacy and security reasons. Whatever the intention may be, the goal is to remove specific objects from a scene and cleverly fill in the missing image data with fabricated data that looks convincingly part of the original image.

As described, this is a two-stage problem. First, we must accurately and precisely identify the target object(s). The more accurate the mask, the more convincing the inpainted region will be due to both the increased context available for inpainting and the decreased chance of accidentally leaving part of the object in the final image. Secondly, we remove the segmented object and fill in the missing space with generated image data without introducing noticeable distortion or artifacts to the fabricated sections of the image. Because the proposed structure consists of two independent networks feeding into one another, it could easily be adapted to use other networks for either the image segmentation or generative inpainting tasks. Thus, the novel contribution of this paper is not training a more effective network for either half of the process, but proposing a framework whereby a task not previously accomplished by a neural network (outpainting of specific objects without modifying surrounding data) can be accomplished by two working in series.

The success of the solution will be qualitatively assessed since fabrication of image data is intended to convince humans that the edited image is authentic. This is a standard problem in testing Generative Adversarial Networks, since any easily defined function that could be used to test the realism of the generator's output would obviate the need for the discriminator. Thus, most proposed evaluation methods determined to be successful by Borji [3] involve either

implementing yet another network (FID) or have severe shortcomings based on the exact content of the generated data, and thus finding a suitable testing method proved more difficult than anticipated. Frameworks have recently been developed to detect if an image has been doctored or manipulated artificially [4]. For the sake of completeness, we have collected test images where we construct scenes with objects in them, and then manually remove some of these objects in order to obtain a ground truth. However, due to the lack of a large database of ground truth images (explained in more detail in Section V), our main focus will be on assessing success qualitatively.

III. Background

In order to understand our pipeline, we must first discuss the two distinct architectures that it comprises: MRCNN (Mask Region-Convolutional Neural Network) and GIIN (Generative Image Inpainting Network). MRCNN, based on Faster R-CNN (FRCNN), is an object detection, classification, and mask generation network. [1] MRCNN itself is a two stage process, which consists of the Region Proposal Network (RPN) found in FRCNN and a mask generation for each region of interest. An innovative feature of MRCNN is that the mask generation happens in parallel to the classification and bounding box generation. This allows a decoupling of mask generation and class prediction, which allows mask generation without competition between classes and leads to more accurate masks. The two stage MRCNN network consists of a convolutional “backbone” for its first stage of feature extraction and a “head” for its second. The backbone architecture uses a Feature Pyramid Network combined with a ResNet for a depth of 101 layers. The head portion is an extension of the FRCNN head network with an added fully

convolutional mask prediction branch. The MRCNN implementation we used was trained on the well known COCO dataset.

The second portion of the pipeline is the GIIN, which takes as input an image and a mask, and outputs the image with the masked area inpainted from surrounding context [2]. The GIIN network also consists of two stages: a coarse network and a refinement network. The coarse network is a dilated convolutional neural network trained with a spatially discounted reconstruction loss, and its purpose is to rough out the missing contents of the input image. The refinement network is where the actual inpainting takes place, which is done using a contextual attention module in parallel with a convolutional pathway. The convolutional path is used to generate new content, while the contextual attention module uses features of known patches as convolutional filters in order to process the generated inpainted patches. The contextual attention module has a spatial propagation layer that enforces spatial coherence, avoiding mismatched edges that sometimes occur in poorly inpainted images. The outputs of the pathways are fed into a single decoder that outputs the final image. The entire network is then trained with reconstruction losses as well as two Wasserstein GAN losses, one global and one local to the patch. The GIIN implementation we used was trained on the Places2 dataset. Figure 1 depicts the GIIN architecture, taken from the original paper and modified for clarity.

The main task of our pipeline is to seamlessly integrate the two networks by modifying the input/output structure of Mask R-CNN and GIIN so images or image batches could be inpainted with no guidance required from the user. The out-of-box implementation of both networks was designed to be run from the command line, so the first thing we had to do was change the GIIN so that we could use it in code. Next, the MRCNN would only accept an image

filepath as an input, and output a set of bounding boxes, classes, and masks. We modified this so that we can specify an input image as well as an object class we wish to remove and the MRCNN would then output only the masks of the class we specified (above some class probability threshold). Our pipeline prompts the user to select from identified classes, which removes the burden of manually sorting through the classes of MRCNN. We then took the relevant masks, and enlarged them by convolving with a 21x21 kernel of ones. We did this because we found that the masks tend to underestimate the region of the object, and that the GIIN works better with overestimate-sized patches. We then use the enlarged mask(s) to remove the desired object from the image, and feed this as the input to the GIIN. The GIIN outputs the inpainted image, thus completing our pipeline (see Figure 2).

IV. Framework and Architecture

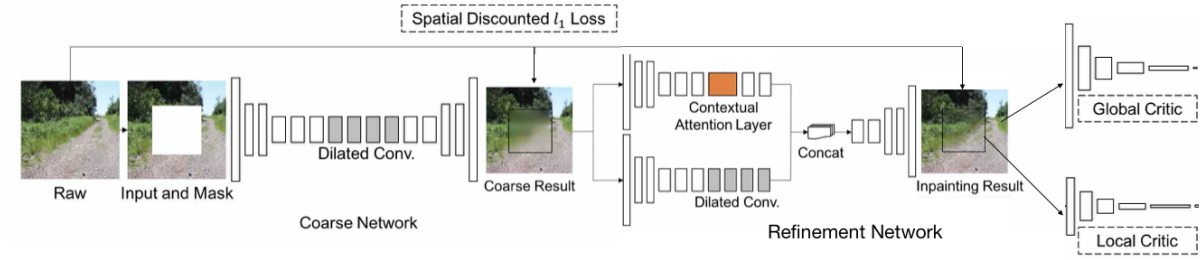

Figure 1: Generative Image Inpainting Network Architecture.

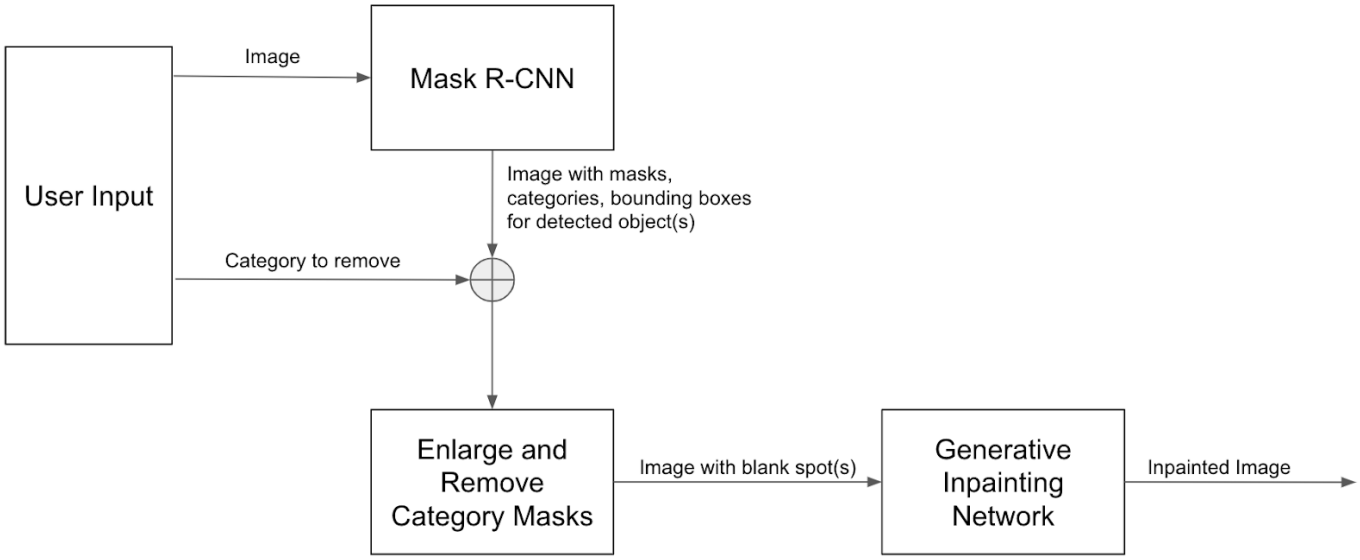

Figure 2: General network flow for the framework we propose.

V. Experiments and Results

Since the desired output of our pipeline is an image with certain objects convincingly removed, success directly depends on how accurately and reliably MRCNN can identify objects on the pixel level, as well as how well GIIN can fill in identified areas. To probe the efficacy of our pipeline, we tried a host of varied types of input images and identified several failure points. Results of our tests can be seen in the Appendix, in Figures 3-8. In some cases, we were able to manually place and then remove objects from a scene, allowing us to construct a Ground Truth image and compute PSNR. We test a variety of lighting conditions and camera settings in order to well-characterize the performance of the pipeline. Test images in Figures 5-8 were taken with a Sony A7iii in manual mode on a tripod to ensure consistent exposure, focus, framing, and depth-of-field. Images in Figures 3-4 were taken on authors' iPhones, and serve as representative

examples of what the average use case might look like. Some tests are presented alongside a ground truth, which is noted in each description. Full quality images are available here.

V(a). General Success Cases

Qualitatively, our proposed pipeline produces acceptable results in many cases. Namely, when scenes have clearly identifiable objects that do not occupy very large areas, they are inpainted with convincing results. Figure 6 illustrates how performance correlates with object area. One crucial part of our pipeline is expanding the masks generated by MRCNN. When segmenting objects on the pixel level, MRCNN sometimes misses boundaries, which can have disastrous results for reconstruction with GIIN. We pad each mask by convolving it with a 21x21 array of ones (see Figure 3). We found that overestimating object boundaries is preferable to underestimating, because GIIN can erroneously incorporate areas of the object as context if they are excluded from the mask.

We note a few conditions that seem to correlate with better results during the inpainting step. First, it helps if the object to be removed takes up a relatively small portion of the scene; we think that a smaller unknown area allows for better incorporation of context, and a smaller area is less noticeable to a human observer. This is demonstrated with several examples: First, we show a scene with the removed object getting progressively smaller, from a human in the foreground to a car in the background. Then, we show a direct comparison between removing a large human compared to removing a small frisbee (see Figures 5 and 6).

We also note that performance seems better when the area to be filled in is blurred by bokeh effects from the camera. We conducted an experiment of inpainting a human figure in front of both a sharp and a blurry background, and note that the results from the blurred image

are superior (see Figure 8). We also see this result when removing the tennis racket from different scenes. It looks more convincing when held in front of a blurry background of trees rather than in front of a detailed brick background.

V(b). Shortcomings

Although our pipeline works well in many cases, it can run into limitations depending on a host of factors. Primarily, results subjectively degrade in quality as any of the following occur: the size of objects to remove dominates the image, scenes are excessively cluttered, objects cover complicated patterns, objects have large shadows, or the scene is very dissimilar to images in the Places2 dataset. However, it is important to note that these conditions can also be very difficult for a human attempting to remove an object from a scene using Photoshop. In some of our more extreme failure modes, the objects we were attempting to remove took up much larger portions of the image than a human could reasonably expect to remove simply by incorporating context from the image (eg with Photoshop's Clone brush). These failure conditions can be separated into those related to object detection and those related to inpainting. When object detection fails, such as cases where objects are obscured or deformed, there is no way to automatically generate a masked area for the GAN to fill in. In cases where objects have been correctly classified, these objects may cast shadows that are not identified. Not only will these shadows not be identified as areas to fill in, they can also falsely be incorporated into inpainted areas.

In our experiments, we noticed how shadows seem to creep into the inpainted area when they should not be there. For example, when removing the human figure from the brick background, the person casts a shadow and intersects with the shadow from the trees in the background. Although the brick behind the person should be sunny, the inpainted area is a

mixture of bright and dark brick texture and jumps out to a human observer as out-of-place (see Figure 5).

Another area where the pipeline can fall short is objects that are deformed or in odd positions. This was a recurring issue with our experiments using a tennis racket. In some cases, the end of the handle of the racket was not included in the mask, and the error was so large that it could not be corrected by our mask expansion step. This caused the end of the handle to remain in the scene (see Figures 5f and 8f). We also tested MRCNN's detection abilities when an object is placed in an odd orientation. In Figure 7c, we placed a tennis racket such that only its side was showing, and it was not detected, and therefore could not be removed from the scene. We note that MRCNN is still robust, though, as demonstrated by its ability to recognize a small frisbee in many orientations (see Figures 5h, 6d, 7b).

V(c). Speed

Before fully implementing our pipeline, we expected it to run on the order of seconds. Now we realize that it is unrealistic on our consumer-grade hardware. To save computation, we downsize each input image such that its short side is 600px long and its long side is no more than 1000px while maintaining the aspect ratio. After downscaling the image, our pipeline runs on the order of one minute per image. This is almost entirely due to the time required for object detection, since the implementation of Mask R-CNN we are using is not optimized for speed. Possible improvements are discussed in the *Future Work* section.

VII. Conclusion

VII(a). Future Work

We think there are several natural extensions of our proposed pipeline. First, we would like to adapt the mask generation step to identify all faces in an image, and then use a GAN to paint a procedurally generated face on top of each real face. This could be accomplished using a network trained on the Celeb-A dataset. This could further be extended by adding a facial recognition component, which would then enable us to select which faces to modify based on each person's identity.

Second, it would be possible to implement a context-aware layer that identifies which pre-trained inpainting model would best suit the scene; this could potentially get better results by ensuring that the inpainting model used was trained on images similar to the input.

Third, we could improve the speed of our pipeline. This might be accomplished by implementing MRCNN on our own (or using a version optimized for speed) instead of using the model provided by gluoncv. We could also experiment with techniques like truncating SVD representations of matrices during the inpainting step. This could also be used to expand the pipeline to inpainting of objects across multiple frames of a video. In principle, this is no more difficult than processing a single frame, and time should scale linearly with the number of frames in the video. A naive approach would be to treat each image independently, then string together the processed results. A more involved approach could leverage the spatial correlation of objects between frames to more accurately and reliably detect objects and construct masks. It would also be aware of past and future frames when inpainting the current frame. We tried implementing the

naive approach for this paper, but had technical difficulties handling different file formats, and even small videos would take a prohibitively long time to process on our hardware.

Fourth, the mask expansion step could be improved significantly. We could increase the specificity of the masks by using more advanced morphological techniques rather than convolution with a square filter; we simply used the convolution because it was easy to implement. Secondly, we could use a shadow detection algorithm such as the one proposed in [5] to remove the shadows of objects that are no longer present in the scene. One of the major failure modes of our pipeline was when shadows were inpainted back into the region of the removed object, creating prominent visible artefacts; this algorithm could help remedy this.

Since our pipeline is very modular, the door is open for others to add components they think might be useful, such as the facial recognition tool discussed earlier. This is the major contribution of our pipeline, since object recognition and inpainting algorithms exist already, but using them together in a modular approach like this opens the door for many new applications.

VIII. References

[1] He K., Gkioxari G., Dollár P., & Girshick R. (2017, October). Mask R-CNN. In Computer Vision (ICCV), 2017 IEEE International Conference on (pp. 2980-2988). IEEE.
[2] Yu J., Lin Z., Yang, J., Shen X., Lu X., Huang, T. S., (2018, January) Generative Image Inpainting with Contextual Attention
[3] Borji, Ali. (2018). Pros and Cons of GAN Evaluation Measures. 10.13140/RG.2.2.16789.42720.
[4] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman and Alexei A. Efros. "Generative Visual Manipulation on the Natural Image Manifold", in European Conference on Computer Vision (ECCV). 2016.
[5] Prati, Andrea et al. “Shadow detection algorithms for traffic flow analysis: a comparative study.” (2001).

Appendix

Figure 3. a) Mask from MRCNN. b) Expanded mask after convolution. c) Input image. d) Results using mask from (a). e) Results using mask from (b).

a

b

c

d

e

f

Figure 4. a) Input image. b) Result from removing ‘person’ object from (a). c) Input image. d) Result from removing ‘vase’ objects from (c). e) Input image. f) Result from removing ‘person’ from (e).

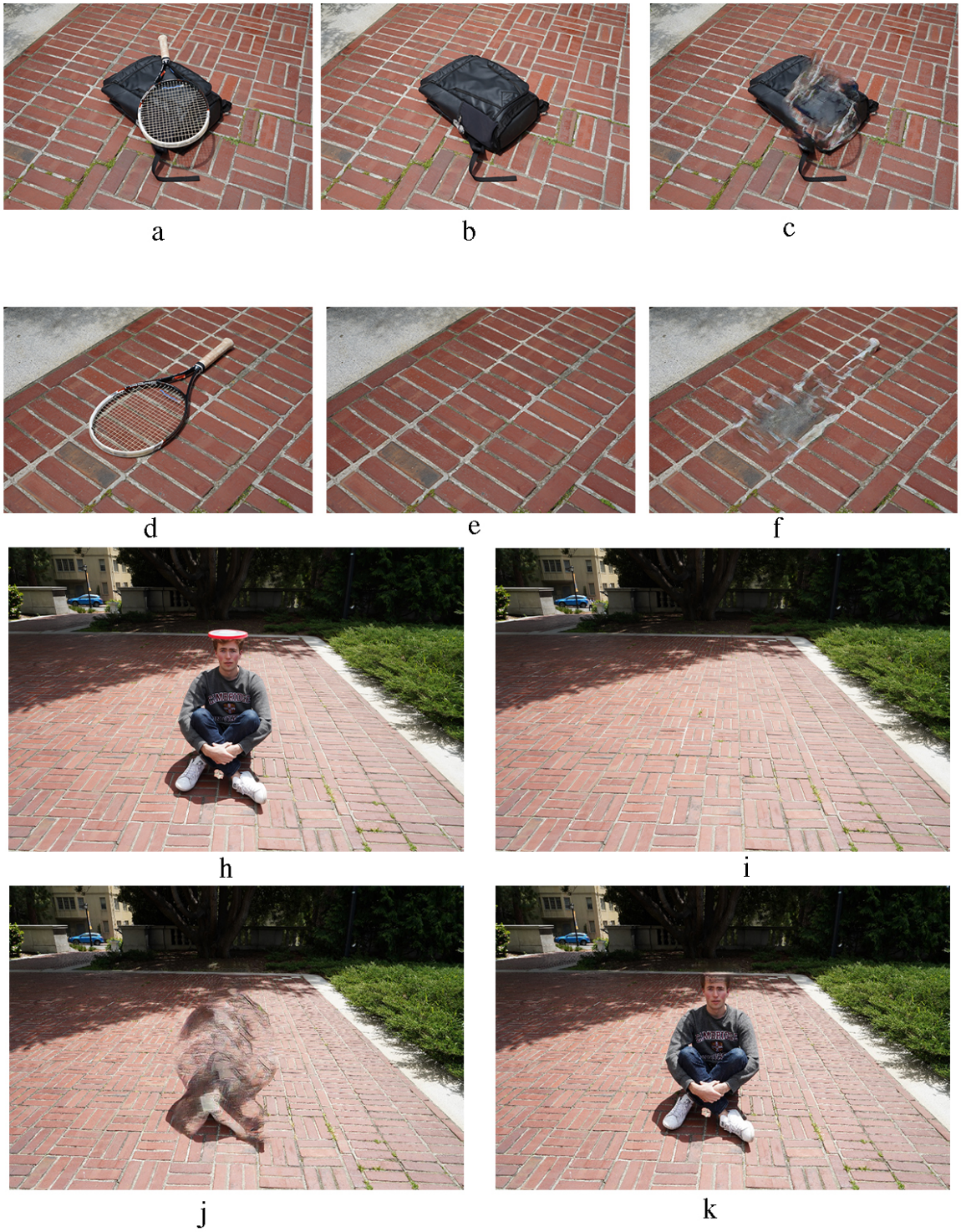

Figure 5. a) Input of objects stacked on each other. Both the bag and racket are recognized. b) Ground truth of manually picking up the racket and removing it from the picture. c) Removing the racket from (a) using our pipeline (PSNR 28.5). d) Input image. e) Ground truth from picking up the racket. f) Removing the racket using the pipeline (PSNR 29.8). h) Input. i) Ground truth for (h). j) 'person' removed from (h) (PSNR 30.0). k) 'frisbee' removed from (h) (PSNR 30.0).

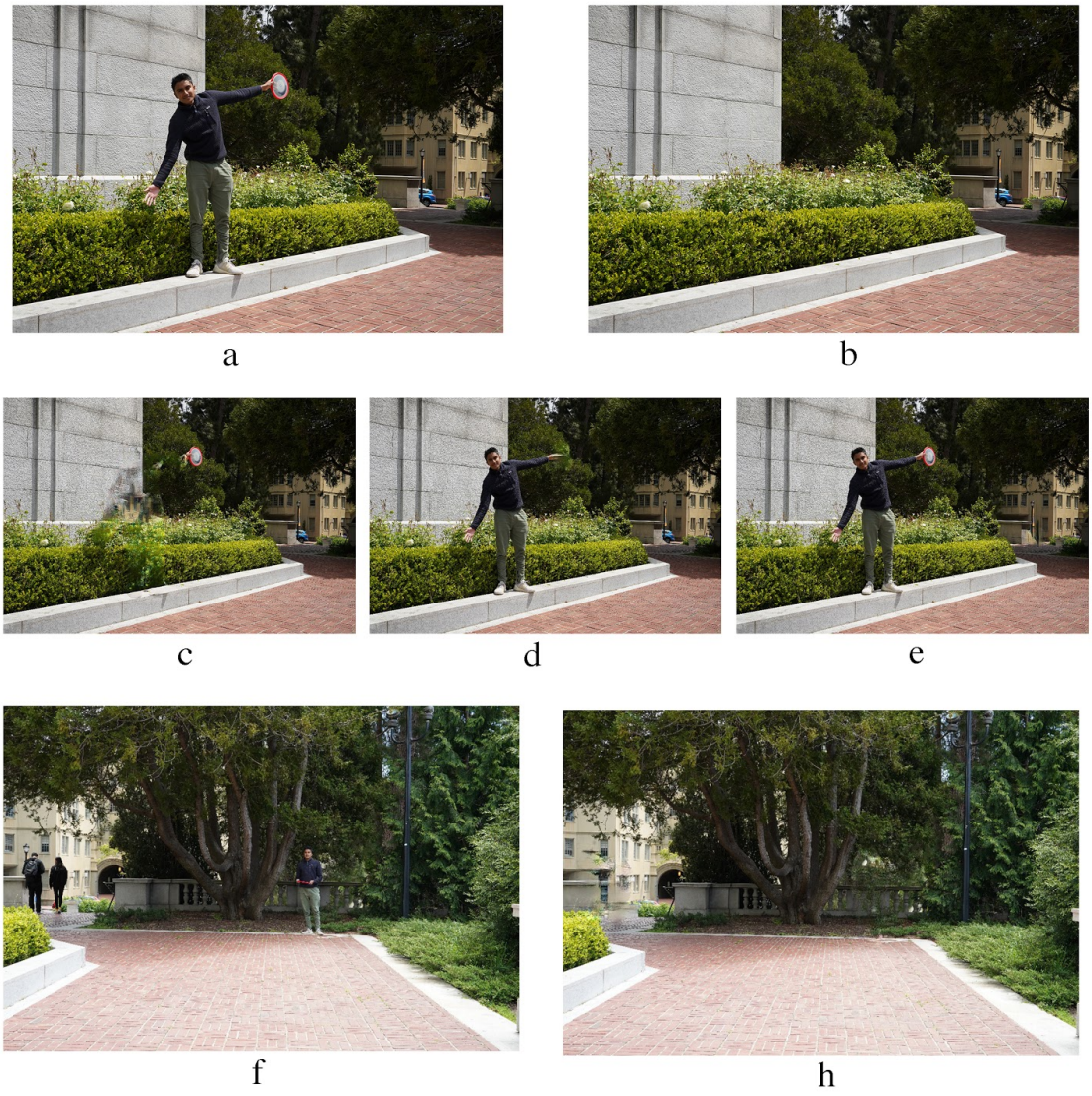

Figure 6. a) Input with many objects. b) Ground truth with person and frisbee gone. c) ‘person’ object removed. Note how the boundaries of the clocktower and hedges are partially reconstructed (PSNR 29.0). d) ‘frisbee’ object removed. e) ‘car’ object removed from background. f) Input with small areas covered by objects. h) ‘person’ objects procedurally removed from (f).

a

b

c

Figure 7. a) Input with a person's face obscured by another object. b) 'frisbee' removed from (a). Note how a face is not reconstructed, this is a limitation of the dataset the GAN was trained on. c) MRCNN failed to recognize the tennis racket. This illustrates how objects in unusual orientations or positions can be difficult to detect.

a b

c d

e f

Figure 8. a) A person in front of a sharp background. b) ‘person’ removed from (a). c) A person in front of a blurred background. Blurring was achieved with a larger aperture, it is not computer-generated. d) ‘person’ removed from (c). e) A tennis racket in front of a slightly blurred background. f) ‘tennis racket’ removed from (e). Note how not all of the handle is detected by Mask-RCNN as belonging to the racket object.