Removal of Background People Using Object Detection and Inpainting

Yuanfang Li
Stanford University
yli03@stanford.edu

Xin Li
Stanford Univeristy
xinli16@stanford.edu

Abstract

We present an application of Viola-Jones face detection and exemplar-based inpainting to automatically select and remove unwanted background person(s) in photographs. We experiment with using various source and target region masks as well as patch and neighbourhood selections to produce realistic inpaintings. The final pipeline is able to generate results comparable to those achieved with manual target selection and can be used as a starting point for more complex target selection and inpainting algorithms.

1. Introduction

The primary motivation behind this project is to remove unwanted objects from photographs and fill in the gap in a visually plausible manner; we focus on removing people as unwanted background people is a frequent problem in photography, particularly when taking photos at popular tourist destinations. Common approaches require manual identification of the area to be removed, however we aim to be able to automatically detect the target area and create an end-to-end pipeline for background removal. Automation of this process can be extremely useful considering the prevalence of photo-taking nowadays, and could be incorporated into the image processing pipeline to produce better photographs without requiring user intervention. To this end, this project consists of three primary components:

1. Detect the location of both the person to keep and unwanted background people
2. Generate the target region to remove based on the location of the background people as well as the source region with which to fill in the gap
3. Fill in the identified target region in a realistic manner using information from the source region

Our final background removal pipeline takes as input a photograph where we wish to keep only the person in the foreground, and produces an image with the background people removed as the final output.

2. Related Work

2.1. Object Detection

Current state-of-the-art object detection algorithms focus largely on the use of regional convolutional neural networks (R-CNNs)[8]. These networks leverage CNNs, which are highly effective at the task of classification and apply them to object detection. The network first extracts thousands of region proposals using the selective search algorithm, which hierarchically joins segments of the image together into larger regions based on similarities between segments such as colour, shape and texture[13]. These region proposals are then fed into a standard CNN for classification in order to extract a fixed length feature vector for each proposed region. Finally, each vector is fed through a class-specific linear SVM for each object to determine which object is contained in the region.

The R-CNN architecture is further improved in [7], which uses region of interest pooling to extract the feature vectors of all the regions in a single pass through the CNN. Further, the linear SVM is folded into the CNN as a softmax layer so that the entire model can be jointly trained. [11] compresses the model even more by removing the selective search and feeding the CNN features directly into a region proposal network. In this way, the entire object detection network can be trained as one.

Although R-CNNs produce state-of-the art results and can generalize well to a variety of objects, they require an extensive amount of data and computation to train. More traditional image processing methods using simpler features can be used for the relatively easier task of only detecting faces/humans. One of the classic algorithms for face detection is the Viola-Jones face detection algorithm, which is able to provide fast, real-time detection [14]. This algorithm attempts to match a set of rectangular Haar features representative of the human face to the given image; classification is done using cascading layers of weak classifiers employing different features into a single strong classifier.

Because the Viola-Jones algorithm relies on the matching of regular facial features, it can generally only detect faces that are directly facing the camera. A similar al-

gorithm presented by Viola et al. in [15] also uses Haar features and a cascade classifier on consecutive frames of video in order to detect pedestrians. On a still image, the histogram of oriented gradients (HOG) method can also be used to detect humans in a variety of poses[5]. This algorithm uses a subsampling and a sliding window detector to calculated a HOG feature descriptor at different positions and scales of the image. The descriptor is then fed into an SVM that classifies the window as a person/non-person.

2.2. Image reconstruction

Traditionally, reconstruction of deteriorated or missing components of images can be separated into two classes. In texture synthesis methods, a repeating sample texture or texel is taken from elsewhere in the image and used to fill in large gaps. The algorithm presented in [10] describes patch-based sampling using an estimate of the local Markov Random Field density function to synthesize textures from a small input sample. Unfortunately, texture-based methods tend to perform poorly on images that have many different backgrounds of different textures. Conversely, structural inpainting methods are more suited for small gaps and focus on propagating edges and structures within the image. [2] proposes a structural inpainting algorithm that performs well on small regions crossing texture boundaries by promoting the propagation of isophotes. This is done by computing the normal to the gradient vector at each pixel on the fill front to preserve edges, then using a 2D Laplacian to obtain the colour variation at the boundary and propagating this information along the normal.

Many algorithms have focused on a combination of the texture and structural methods. [3] simultaneously fills in texture and structure by decomposing the image into two images representing characteristically different functions of texture and structure. These decomposed images are then filled in using synthesis and inpainting respectively, and the results are combined back into the final image. Criminisi et al. present an alternative method of incorporating both texture and structure in [4]. This method uses patch samples as in texture synthesis, but instead of reconstructing patches of the image from the outside in, it uses a order that propagates existing edges/structures via isophotes.

Like object detection, more recent work on image reconstruction has also focused on using CNNs. [16] adapts denoising auto-encoders to the task of blind image inpainting. Generative adversarial networks (GANs) have also been used to realistically colour and alter the style of images[9][6]. These networks are composed of a generator and discriminator network. The generator attempts to generate a realistic image that will fool the discriminator while the discriminator attempts to differentiate between true and false (generated) images. However, GANs are not only data and compute intensive but also difficult to train.

Our work applies the Viola-Jones face detection models in MATLAB vision toolbox to detect the location of people within the input image and uses exemplar-based inpainting based on the implementation in [1] to fill in the removed target regions of the image. As our input images are relatively simple, these fast and traditional methods should be able to work comparably well. Given more complex background structures or more people, it may be necessary to move to a neural network implementation to achieve good results.

3. Methods

Figure 1 below shows the end-to-end pipeline of object detection and removal. In the following section we describe each component in further detail.

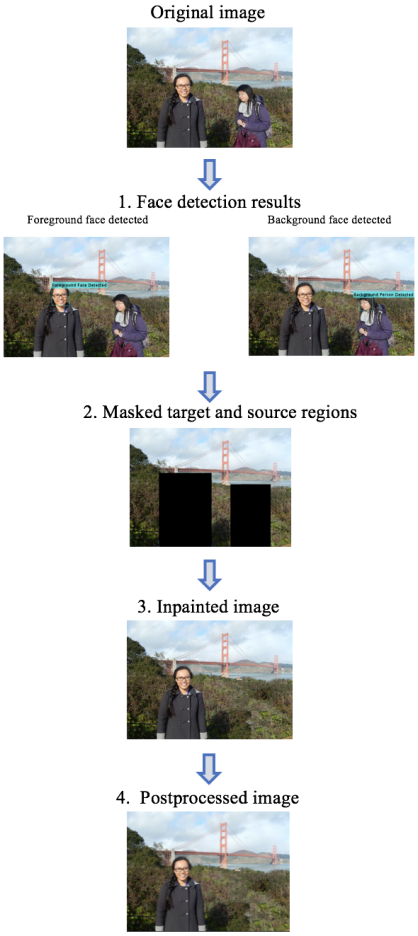


Figure 1: Detection and Inpainting Pipeline

3.1. Viola-Jones Face Detection

The first step in our pipeline is to detect the location of people in the input photograph. The Viola-Jones face detection algorithm[14] is a robust method for quickly distinguishing faces from non-faces. It operates on the integral image of the input photograph and works by combining a series of simple binary classifiers into a single cascade structure able to detect more complex features.

3.1.1 Integral Image

In order to perform face detection, we must first generate features that can be used for classification from the input image. The features used in Viola-Jones face detection algorithm are reminiscent of Haar basis functions. They can be categorized into three types: two-rectangle features, three-rectangle features and four-rectangle features.

[14] introduces a method of representing image features efficiently using integral images. All three types of rectangle features mentioned above can be computed using the integral image, which is defined as follows: each value at position (x, y) is the sum of all the pixel values to the left of and above this position, i.e.

$$ii(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x', y')$$

in which $ii(x, y)$ is defined as the pixel value of position (x, y) in the integral image, and $i(x, y)$ is the pixel value in the original image. We can then efficiently compute the integral image using the recurrence of the cumulative row sum and the image pixel value as follows:

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

3.1.2 AdaBoost Algorithm

A common difficulty faced when training classifiers is that adding more features to the classifier may help to improve performance but will cause the model to beomce more computationally expensive. Further, too many features may result in the classifier overfitting on the training set samples and generalizing poorly to new test samples it has not seen before. In order to both reduce the risk of overfitting and simplify the model, we aim to find a method of selecting a small fraction of features to train the classifier. There are several effective feature selection methods such as PCA or decision tree; the AdaBoost algorithm is based on the Winnow exponential perceptron learning algorithm[12], which produces a result where most of the weights in the classifier are zero. Thus while we provide a large set of features, only a small fraction corresponding to the non-zero weights are actually used, and we allow the learning algorithm to select which features to keep and which to discard.

The algorithm is as follows[14]:

Given images $x_1, \cdots, x_n$ and their corresponding labels $y_1, \cdots, y_n$.
Define m as the number of negative examples, define l as the number of positive examples. Initialize weights:

$$w_{1,i} = \begin{cases} \frac{1}{2m}, & \text{if } y_i = 0 \\ \frac{1}{2l}, & \text{if } y_i = 1 \end{cases}$$

For $t = 1, \cdots, T$:

1. Normalize the weights
$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
2. For each feature j, train a classifier h_j which is restricted to using a single feature. The error is:
$$\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$$
3. Choose the classifier h_t with the lowest error ϵ_t.
$$t = \arg\min_j \epsilon_j = \arg\min_j \sum_i w_i |h_j(x_i) - y_i|$$
4. Update weights
$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
in which
$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$$
$$e_i = \begin{cases} 0, \text{ if } x_i \text{ is classified correctly} \\ 1, \text{ otherwise} \end{cases}$$

The strong classifier we get

$$h(x) = \begin{cases} 1, \text{ if } \sum_{t=1}^{T} \alpha_t h_t(x) \ge \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, \text{ otherwise} \end{cases}$$

in which

$$\alpha_t = \log \frac{1}{\beta_t}$$

3.1.3 Classifier Cascade

The Viola-Jones algorithm further improves detection accuracy by using a degenerate decision tree process to combine a series of simple classifiers into a detection cascade. In each stage, if the outcome is negative (i.e. not a face), then the corresponding sub-window is rejected. Otherwise we evaluate the classifier in the next stage. This classifier cascade tends to reject classifiers with negative outcomes in the first stages and thus has better performance and is less computationally expensive.

3.2. Generating Source and Target Region

We use the Viola-Jones face detection algorithm to detect both the person in the foreground we wish to keep and the background person(s) we wish to remove. We then use the detected faces to generate source and target region masks as follows:

1. Use the full-frontal model to detect forward facing faces in the image. Return a square bounding box of size $n \times n$ around the largest face detected.

2. We assume that the image of the person to keep is not full-body and mask out the region by extending the bounding box to the left and right by n pixels, up by $n/3$ pixels and down to the bottom of the image.

3. Use the profile model to detect profile faces in the image. Return a square bounding box of size $m \times m$ around each face detected.

4. For each face that does not overlap with the face detected in Step 1, mask out the region by extending the bounding box to the left and right by $n/2$ pixels, up by $n/3$ pixels and down by $8 \times (4n/3)$ pixels. Identify this as part of the target region we wish to remove.

5. If no faces are detected using the profile method, use the whole body model to detect any remaining people. Return a square bounding box of size $h \times w$.

6. For each person that does not overlap with the face detected in Step 1, mask out the region by extending the bounding box to the left and right by $w/3$. Identify this as part of the target region we wish to remove.

7. Finally identify the source region as all unmasked regions of the image.

The algorithm above was obtained through experimentation with various photos and detection models as well as human body proportions which are further explained in Section 4.

3.3. Exemplar-Based Inpainting

Once the source and target regions of the image are identified, we use exemplar-based inpainting as presented in [4] to fill in the removed person(s) in a visually plausible manner. This algorithm combines both texture synthesis (used to propagate two-dimensional repeating textures) and inpainting methods (used to propagate one-dimensional linear structure) and is comprised of two key steps: selecting the next patch of the target region to fill in and selecting the patch of the source region with which to fill it.

3.3.1 Selection of target patch

Unlike in traditional texture-synthesis methods that fill in the target region in concentric layers from the outside in, in exemplar-based inpainting, patches of the target region are filled in by priority value. The priority value of a patch along the fill-front, $\Psi_{\mathbf{p}}$ centered at pixel $\mathbf{p}$ is computed as the product of its confidence $C(\mathbf{p})$ and data $D(\mathbf{p})$ terms.

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in \Psi_{\mathbf{p}} \cap \bar{\Omega}} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}$$

Confidence $C(\mathbf{p})$ of a pixel is defined as the average of the confidence of the pixels in patch $\Psi_{\mathbf{p}}$ that have been filled in and measures the reliability of the pixels surrounding p. By initializing the pixels in the source and target regions with confidence of 1 and 0 respectively, $C(\mathbf{p})$ will decrease as we move towards the center of the target region and can thus be thought of as approximating the concentric fill order used in most texture synthesis algorithms.

$$D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{p}}^{\perp} \cdot \mathbf{n}_{\mathbf{p}}|}{\alpha}$$

Conversely, the data term $D(\mathbf{p})$ tends to direct the fill order in tendrils towards the centre of the target region. The dot product of the isophote at pixel $\mathbf{p}$, $\nabla I_{\mathbf{p}}^{\perp}$, and the normal to the fill front, $\mathbf{n}_{\mathbf{p}}$ is largest when the two vectors are parallel to each other. The isophote is the curve on an illuminated surface that connects points of equal brightness and can be used to represent linear structures. Thus the data term is higher for pixels where the isophote has a large magnitude and is perpendicular to the fill front, promoting propagation of strong linear structures into the target region.

By using a product of the confidence and data terms, the exemplar-based inpainting algorithm attempts to find a balance between filling in textures from the outside in and propagating edges.

3.3.2 Selection of source patch

Once the target patch to be filled in is determined, we need to find an exemplar source patch with which the algorithm will fill in the pixels of the target patch. This is achieved through calculating the similarity between the target patch $\Psi_{\mathbf{p}}$ and each patch $\Psi_{\mathbf{q}}$ in the source region, where similarity is defined as the sum of squared distances between filled in pixels in the two patches.

$$\Psi_{\hat{q}} = \text{argmin}_{\Psi_{\mathbf{q}}} d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\mathbf{q}})$$

The source patch with the highest similarity is selected as the best exemplar, $\Psi_{\hat{\mathbf{q}}}$ and each pixel in $\Psi_{\mathbf{p}}$ is filled in with the value of the corresponding pixel in $\Psi_{\hat{\mathbf{q}}}$.

3.4. NLM Filtering

Non-local Means is an image denoising technique which makes use of patches with similar neighbourhoods to denoise/smooth the image. As in other denoising methods, pixels that are in similar windows are averaged together, however in NLM, the pixels do not need to be close to each other and can be selected from anywhere in the image as follows:

$$w(x, x') = \exp\Big(-\frac{||W(i_{\text{noisy},x'}) - W(i_{\text{noisy}}, x)||^2}{2\sigma^2}\Big)$$

$$i_{\text{denoised}}(x) = \frac{\sum_{x'} i_{\text{noisy}}(x')w(x, x')}{\sum_{x'} w(x, x')}$$

where $w(x, x')$ is the weight on how similar pixel x is to all other pixels x' and i_{noisy} and $i_{denoised}$ are the noisy and smoothed images respectively.

4. Experiments and Results

Figure 2 shows the final results of our model with the input image, automatically generated masks and inpainted and smoothed result. Compared to manual target selection, we are able to achieve an average PSNR of 26.18dB.

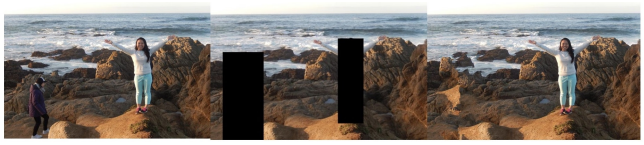
In general our method is able to produce visually plausible results, particularly for images 2a to 2d. In 2b, we can see that the background people at the right edge of the image are not removed as they are too small to be detected by the Viola-Jones algorithm. 2c produces an acceptable inpainting but closer inspection reveals that it has inpainted the sunlit rocks from the foreground into the target region, which was originally in shadow. This error is likely due to the generated target mask covering too large of a region as it does not occur when using a manual mask that covers only the person to be removed.

In image 2e, the more complex background structures lead to a very patchy and unrealistic inpainting of the removed region whereas image 2a results in the most realistic inpainting as the target region contains only a single texture. This suggests that isophote propagation in the inpainting algorithm does not work well when there are too many structures as multiple patches on the fill front may have large data terms. This would cause the confidence term to dominate in priority calculations, promoting the concentric fill order used in texture synthesis techniques and leading to poor results on texture boundaries.

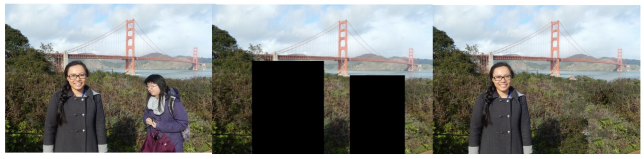
We experimented with various different parameters to create a better mask generation process and obtain more realistic inpainted images. The results are discussed below.

(a) Patch size = 9, Target region = 4.72 % of source region

(b) Patch size = 7, Target region = 2.77 % of source region



(c) Patch size = 11, Target region = 11.91 % of source region



(d) Patch size = 17, Target region = 16.32 % of source region

(e) Patch size = 9, Target region = 6.58 % of source region

Figure 2: Background people removal results

4.1. Source and target selection

As shown in figure 3, we first compare the performance of using an automatically generated rectangular mask against a manually selected mask as presented in the original inpainting algorithm. The resulting inpainted target region using the two different masks can be seen in figure 5c and 5d. These two methods lead to similar performance. Manual target selection produces slightly better results with less patches, but both are visually plausible.

When using rectangular masks, it is important to remove the foreground person from the source region. As shown in figure 5a and 5c, if we keep the person of interest in the source region, then it is possible that our model will use patches from the foreground person to inpaint the target region. This problem is not as prevalent when using manual target selection.

We also use different detection models to detect the

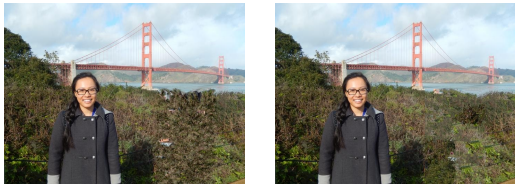
background people to remove. While the 'Full Face' model worked consistently for the foreground person facing the camera, the background people were usually in different orientations and poses. Through experimentation, we find that the 'Profile Face' model results in the fewest false positives but is unable to detect people that are too small. The 'Upper Body' model is able to detect smaller people but also frequently classifies other objects as people (i.e. the Golden Gate Bridge). Thus in our model, we first apply the 'Profile Face' detector and only use the 'Upper Body' detector if this results in no matches.

(a) Generated rectangular target region mask (b) Manually selected target region mask

Figure 3: Comparison of different masks used

4.2. Patch size

Exemplar-based inpainting uses a default patch size of 9 and recommends that the patch size be selected to be slightly larger than the smallest texel in the image[4]. Through experimentation, we find that the optimal patch size for inpainting is approximately 10% of the minimum target mask dimension. As seen in Figure 4, a patch size that is too small can lead to noticeable repetitive patterns in the inpainted area while a patch size that is too large can cause incorrect textures to be inpainted. With the optimal patch size chosen, there is minimal error in inpainting and the patchiness can be mitigated by applying NLM filtering.



(a) Patch size 7 (b) Patch size 17

(c) Patch size 27

Figure 4: Comparison of different patch sizes

4.3. Local source selection

Compared to manual target selection, automatic target selection can lead to incorrect source patches selected, particularly towards the center of the target region. We attempt to mitigate this by

1. Reducing the patch size as the percentage of the target region to be filled falls below a given threshold
2. Reducing the source region from the entire image to a local neighbourhood as the area still to be inpainted decreases

As shown in figure 2b, the combination of a smaller patch size and local neighbourhood source region causes the inpainting algorithm to use one patch repeatedly, which leads to an extremely patchy pattern. Although this is slightly improved after applying NLM filtering, the final image is still less plausible than the original result in 2a.

(a) Person of interest not removed from the source region (b) Smaller patch size and local neighbourhood for inner target region

(c) Final result with detected target (d) Result with manually selected target

Figure 5: Inpainting for different model configurations

5. Conclusion and Future Work

Our work presents an end-to-end pipeline for automatically detecting and removing unwanted background people from photographs that is able to achieve comparable results to hand-crafted manual target selection. We find that when using automatically generated masks, it is essential to remove the foreground person of interest from the source region to prevent the algorithm from incorrectly selecting source patches from this region. Further, the patch size to be used can also be automatically selected in proportion to the target region mask size.

A large limitation of the model is the simple method in which the masks are generated, thus a clear method would be to generate a mask that better approximates the shape of the person to be removed. One possible method to consider is to start from the simple mask and gradually shrink the region by calculating gradients. Given a simple background, we would expect the outline of the person to have the largest gradients. This could also be made simpler by first high pass filtering the image to bring out edges.

References

[1] Matlab implementation of inpainting algorithm by a. criminisi. `https://github.com/ikuwow/inpainting_criminisi2004/graphs/contributors`, 2015.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[3] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE transactions on image processing*, 12(8):882–889, 2003.

[4] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2414–2423. IEEE, 2016.

[7] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[10] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.

[11] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[12] R. A. Servedio. Perceptron, winnow, and pac learning. *SIAM Journal on Computing*, 31(5):1358–1369, 2002.

[13] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[15] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *null*, page 734. IEEE, 2003.

[16] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.