Finding Distractors In Images

Ohad Fried
Princeton University
ohad@cs.princeton.edu

Eli Shechtman
Adobe Research
elishe@adobe.com

Dan B Goldman
Adobe Research
dgoldman@adobe.com

Adam Finkelstein
Princeton University
af@cs.princeton.edu

Abstract

We propose a new computer vision task we call "distractor prediction." Distractors are the regions of an image that draw attention away from the main subjects and reduce the overall image quality. Removing distractors — for example, using in-painting — can improve the composition of an image. In this work we created two datasets of images with user annotations to identify the characteristics of distractors. We use these datasets to train an algorithm to predict distractor maps. Finally, we use our predictor to automatically enhance images.

1. Introduction

Taking pictures is easy, but editing them is not. Professional photographers expend great care and effort to compose aesthetically-pleasing, high-impact imagery. Image editing software like Adobe Photoshop empowers photographers to achieve this impact by manipulating pictures with tremendous control and flexibility – allowing them to carefully post-process good photos and turn them into great photos. However, for most casual photographers this effort is neither possible nor warranted. Last year Facebook reported that people were uploading photos at an average rate of 4,000 images *per second*. The overwhelming majority of these pictures are casual – they effectively chronicle a moment, but without much work on the part of the photographer. Such cases may benefit from semi- or fully-automatic enhancement methods.

Features like "Enhance" in Apple's iPhoto or "Auto Tone" in Photoshop supply one-click image enhancement, but they mainly manipulate *global* properties such as exposure and tone. Likewise, Instagram allows novices to quickly and easily apply eye-catching filters to their images. Although they have some more localized effects like edge darkening, they apply the same recipe to any image. However, local, image-specific enhancements like removing distracting areas are not handled well by automatic methods. There are many examples of such *distractors* – trash on the ground, the backs of tourists visiting a monument, a car driven partially out of frame, etc. Removing distractors demands a time-consuming editing session in which the user manually selects the target area and then applies features like iPhoto's "Retouch Tool" or Photoshop's "Content Aware Fill" to swap that area with pixels copied from elsewhere in the image.

In this work we take the first steps towards semi-automatic distractor removal from images. The main challenge towards achieving this goal is to automatically identify what types of image regions a person might want to remove, and to detect such regions in arbitrary images. To address this challenge we conduct several studies in which people mark distracting regions in a large collection of images, and then we use this dataset to train a model based on image features.

Our main contributions are: (1) defining a new task called "distractor prediction", (2) collecting a large-scale database with annotations of distractors, (3) training a prediction model that can produce distractor maps for arbitrary images, and (4) using our prediction model to automatically remove distractors from images.

In the following sections we describe related work (Section 2), describe and analyze our datasets (Section 3), explain our distractor prediction model (Section 4), evaluate our predictor (Section 5) and present applications of distractor prediction (Section 6). With this publication we also make available an annotated dataset containing images with distractors as well as code for both analyzing the dataset and computing distractor maps.

2. Related Work

A primary characteristic of distractors is that they attract our visual attention, so they are likely to be somewhat correlated with models of visual saliency. Computational saliency methods can be roughly divided into two groups: human fixation detection [13, 11, 15] and salient object detection [7, 6, 22, 20]. Most of these methods used ground-truth gaze data collected in the first 3-5 seconds of viewing (a few get up to 10 seconds) [15]. Although we found some correlation between distractor locations and these early-viewing gaze fixations, it was not high. Our hypothesis

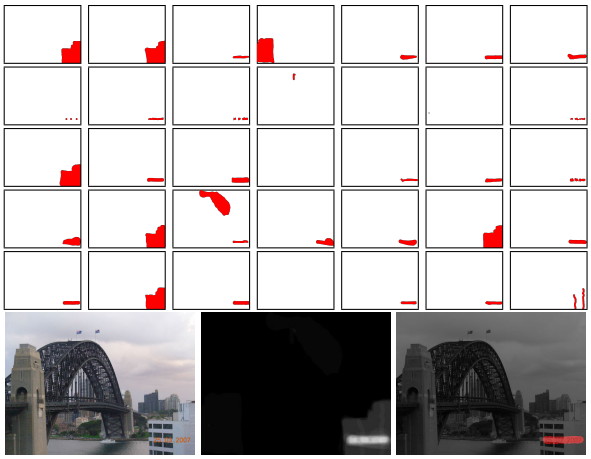

Figure 1. User annotations. Top: 35 user annotations for one input image. Bottom, from left to right: input image, average annotation, overlay of thresholded annotation with input image. We collected 11244 such annotations for 1073 images.

is that humans start looking at distractors after longer periods of time, and perhaps only look directly at them when following different viewing instructions. Existing computational saliency methods are thus insufficient to define visual distractors, because the main subject in a photo where people look first usually has a high saliency value. Moreover, many of these methods (especially in the second category) include components that attenuate the saliency response away from the center of the image or from the highest peaks - exactly in the places we found distractors to be most prevalent.

Another line of related work focuses on automatic image cropping [29, 21, 34]. While cropping can often remove some visual distractors, it might also remove important content. For instance, many methods just try to crop around the most salient object. Advanced cropping methods [34] also attempt to optimize the layout of the image, which might not be desired by the user and is not directly related to detecting distractors. Removing distractors is also related to the visual aesthetics literature [16, 19, 23, 30] as distractors can clutter the composition of an image, or disrupt its lines of symmetry. In particular, aesthetics principles like simplicity [16] are related to our task. However, the computational methods involved in measuring these properties don't directly detect distractors and don't propose ways to remove them.

Image and video enhancement methods have been proposed to detect dirt spots, sparkles [17], line scratches [14] and rain drops [8]. In addition, a plethora of popular commercial tools have been developed for face retouching: These typically offer manual tools for removing or attenuating blemishes, birth marks, wrinkles etc. There have been also a few attempts to automate this process (e.g., [32]) that require face-specific techniques. Another interesting work [28] focused on detecting and de-emphasizing distracting texture regions that might be more salient than the main object. All of the above methods are limited to a certain type of distractor or image content, but in this work we are interested in a more general-purpose solution.

	Mechanical Turk	Mobile App
Number of images	403	376
Annotations per image	27.8 on average	1
User initiated	No	Yes
Image source	Previous datasets	App users

Table 1. Dataset comparison.

3. Datasets

We created two datasets with complementary properties. The first consists of user annotations gathered via Amazon Mechanical Turk. The second includes real-world use cases gathered via a dedicated mobile app. The Mechanical Turk dataset is freely available, including all annotations, but the second dataset is unavailable to the public due to the app's privacy policy. We use it for cross-database validation of our results (Section 5). Table 1 and the following subsections describe the two datasets.

3.1. Mechanical Turk Dataset (MTurk)

For this dataset we combined several previous datasets from the saliency literature [1, 15, 18] for a total of 1073 images. We created a Mechanical Turk task in which users were shown 10 of these images at random and instructed as follows:

> *For each image consider what regions of the image are disturbing or distracting from the main subject. Please mark the areas you might point out to a professional photo editor to remove or tone-down to improve the image. Some images might not have anything distracting so it is ok to skip them.*

The users were given basic draw and erase tools for image annotation (Figure 2). We collected initial data for the entire dataset (average of 7 annotations per image) and used it to select images containing distractors by consensus: An image passed the consensus test if more than half of the distractor annotations agree at one or more pixels in the image. 403 images passed this test and were used in a second experiment. We collected a total of 11244 annotations, averaging 27.8 annotations per image in the consensus set (figure 1).

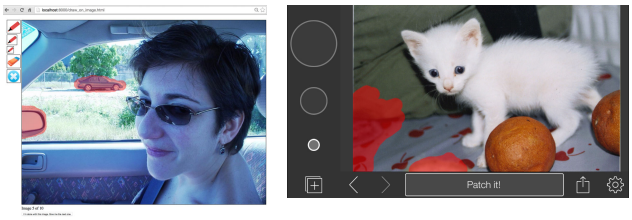

Figure 2. Data collection interfaces. Left: MTurk interface with basic tools for marking and removal. Right: Mobile app using inpainting with a variable brush size, zoom level and undo/redos.

3.2. Mobile App Dataset (MApp)

Although the Mechanical Turk dataset is easy to collect, one might argue that it is biased: Because Mechanical Turk workers do not have any particular expectations about image enhancement and cannot see the outcome of the intended distractor removal, their annotations may be inconsistent with those of real users who wish to remove distractors from images. In order to address this, we also created a second dataset with such images: We created a free mobile app (Figure 2) that enables users to mark and remove unwanted areas in images. The app uses a patch-based hole filling method [4] to produce a new image with the marked area removed. The user can choose to discard the changes, save or share them. Users can opt-in to share their images for limited research purposes, and over 25% of users chose to do so.

Using this app, we collected over 5,000 images and over 44,000 fill actions (user strokes that mark areas to remove from the image). We then picked only images that were exported, shared or saved by the users to their camera roll (*i.e.* not discarded), under the assumption that users only save or share images with which they are satisfied.

Users had a variety of reasons for using the app. Many users removed attribution or other text to repurpose images from others found on the internet. Others were simply experimenting with the app (e.g. removing large body parts to comical effect), or clearing large regions of an image for the purpose of image composites and collages. We manually coded the dataset to select only those images with distracting objects removed. Despite their popularity, we also excluded face and skin retouching examples, as these require special tools and our work focused on more general images. After this coding, we used the 376 images with distractors as our dataset for learning.

3.3. Data Analysis

Our datasets afford an opportunity to learn what are the common locations for distractors. Figure 3 shows the average of all collected annotations. It is clear that distractors tend to appear near the boundaries of the image, with some bias towards the left and right edges. We use this observation later in Section 4.2.

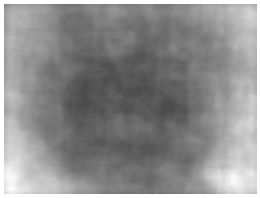

Figure 3. An average of all collected annotations. Distractors tend to appear near the image boundary.

We can also investigate which visual elements are the most common distractors. We created a taxonomy for the following objects that appeared repeatedly as distractors in both datasets: **spot** (dust or dirt on the lens or scene), **highlight** (saturated pixels from light sources or reflections), **face**, **head** (back or side), **person** (body parts other than head or face), **wire** (power or telephone), **pole** (telephone or fence), **line** (straight lines other than wires or poles), **can** (soda or beer), **car**, **crane**, **sign**, **text** (portion, not a complete sign), **camera**, **drawing**, **reflection** (*e.g.* in images taken through windows), **trash** (garbage, typically on the ground), **trashcan**, **hole** (*e.g.* in the ground or a wall).

Treating each annotated pixel as a score of 1, we thresholded the average annotation value of each image in the MTurk dataset at the top 5% value, which corresponds to 0.18, and segmented the results using connected components (Figure 1). For each connected component we manually coded one or more tags from the list above. The tag **object** was used for all objects which are not one of the other categories, whereas **unknown** was used to indicate regions that do not correspond to discrete semantic objects. We also included three optional modifiers for each tag: **boundary** (a region touching the image frame), **partial** (usually an occluded object) and **blurry**. Figure 4 shows the histograms of distractor types and modifiers. Notice that several distractor types are quite common. This insight leads to a potential strategy for image distractor removal: training task-specific detectors for the top distractor categories. In this work we chose several features based on the findings of figure 4 (*e.g.* a text detector, a car detector, a person detector, an object proposal method). An interesting direction for future work would be to implement other detectors, such as electric wires and poles.

All distractor annotations for the MTurk dataset are freely available for future research as part of our dataset.

4. Distractor Prediction

Given input images and user annotations, we can construct a model for learning distractor maps. We first segment each image (section 4.1). Next we calculate features

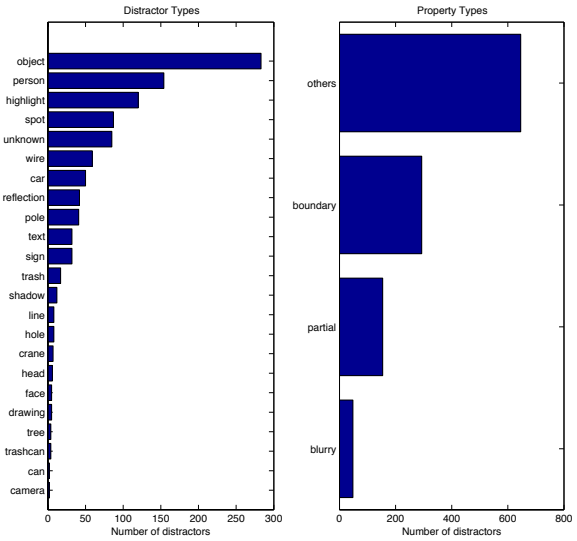

Figure 4. Distractor Types. Left: histogram of distractor types described in Section 3.3. Right: histogram of distractor properties, indicating whether the distractor is close to the image boundary, occluded/cropped or blurry.

for each segment (section 4.2). Lastly, we use LASSO [31] to train a mapping from segment features to a distractor score — the average number of distractor annotations per pixel (section 4.3). The various stages of our algorithm are shown in figure 5.

4.1. Segmentation

Per-pixel features can capture the properties of a single pixel or some neighborhood around it, but they usually do not capture region characteristics for large regions. Thus, we first segment the image and later use that segmentation to aggregate measurements across regions. For the segmentation we use multi-scale combinatorial grouping (MCG) [2]. The output of MCG is in the range of $[0, 1]$ and we use a threshold value of 0.4 to create a hard segmentation. This threshold maximizes the mean distractor score per segment over the entire dataset. We found empirically that the maximum of this objective segments distracting objects accurately in many images.

4.2. Features

Our datasets and annotations give us clues about the properties of distractors and how to detect them. The distractors are, by definition, salient, but not all salient regions are distractors. Thus, previous features used for saliency prediction are good candidate features for our predictor (but not sufficient). We also detect features that distinguish main subjects from salient distractors that might be less important, such as objects near the image boundary. Also, we found several types of common objects appeared

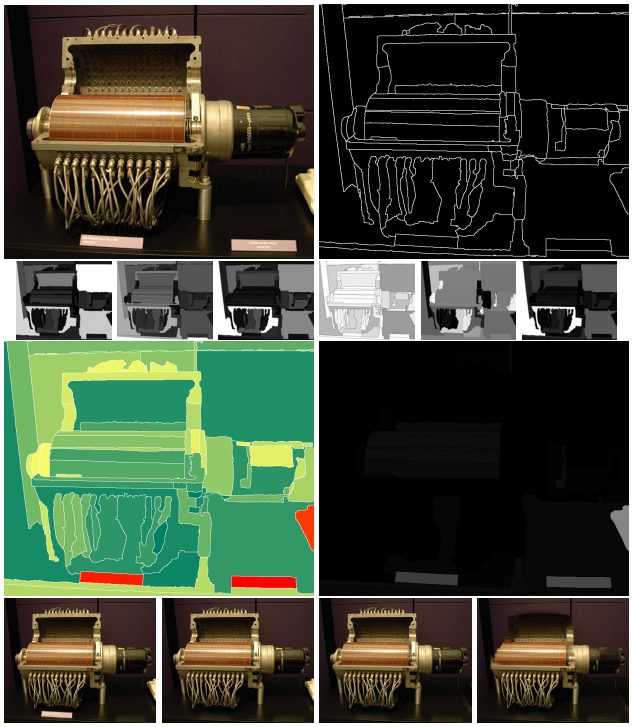

Figure 5. Various stages of our algorithm. Top left: original image. Top right: MCG segmentation. 2nd row: examples of 6 of our 192 features. 3rd row left: our prediction, red (top three), yellow (high score) to green (low score). 3rd row right: ground truth. Bottom row: distractor removal results, with threshold values 1, 2, 3, 20 from left to right. The 3 distractors are gradually removed (3 left images), but when the threshold is too high, artifacts start appearing (far right).

frequently, and included specific detectors for these objects.

We calculate per-pixel and per-region features. The 60 per-pixel features are:

- (3) Red, green and blue channels.
- (3) Red, green and blue probabilities (as in [15]).
- (5) Color triplet probabilities, for five median filter window sizes (as in [15]).
- (13) Steerable pyramids [27].
- (5) Detectors: cars [10], people [10], faces [33], text [24], horizon [25, 15].
- (2) Distance to the image center and to the closest image boundary.
- (7)* Saliency prediction methods [12, 13, 20, 22, 25].
- (2)* Object proposals [35]: all objects, top 20 objects. We sum the scores to create a single map.
- (2) Object proposals [35]: all objects contained in the outer 25% of the image. We create 2 maps: one by summing all scores and another by picking the maximal value per pixel.
- (9) All features marked with $*$, attenuated by $\mathcal{F} = 1 - \mathcal{G}/\max(\mathcal{G})$. $\mathcal{G}$ is a Gaussian the size of the image with a

standard deviation of $0.7*\sqrt{d_1 * d_2}$ (d_1 and d_2 are the height and width of the image) and centered at the image center.

- (9) All features marked with $*$, attenuated by $\mathcal{F}$ as defined above, but with the Gaussian centered at the pixel with the maximal feature value.

All the per-pixel features are normalized to have zero mean and unit variance.

To use these per-pixel features as per-segment features, we aggregate them in various ways: For each image segment, we calculate the mean, median and max for each of the 60 pixel features, resulting in 180 pixel-based features per segment.

Lastly, we add a few segment-specific features: area, major and minor axis lengths, eccentricity, orientation, convex area, filled area, Euler number, equivalent diameter, solidity, extent and perimeter (all as defined by the Matlab function `regionprops`). All features are concatenated, creating a vector with 192 values per image segment.

4.3. Learning

For each image in our dataset we now have a segmentation and a feature vector per segment. We also have user markings. Given all the markings for a specific image, we calculate the average marking (over all users and all segment pixels) for each segment. The calculated mean is the ground truth distractor score for the segment.

We use Least Absolute Selection and Shrinkage Operator (LASSO) [31] to learn a mapping between segment features and a segment's distractor score. All results in this paper are using LASSO with 3-fold cross validation. Using LASSO allows us to learn the importance of various features and perform feature selection (section 4.4).

Besides LASSO, we also tried linear and kernel SVM [9] and random forests with bootstrap aggregating [5]. Kernel SVM and random forests failed to produce good results, possibly due to overfitting on our relatively small dataset. Linear SVM produced results almost as good as LASSO, but we chose LASSO for its ability to rank features and remove unnecessary features. Nonetheless, linear SVM is much faster and may be a viable alternative when training time is critical.

4.4. Feature Ranking

Table 2 shows the highest scoring features for each of the datasets. We collect the feature weights from all leave-one-out experiments using LASSO. Next we calculate the mean of the absolute value of weights, providing a measure for feature importance. The table shows the features with the largest mean value.

Using our feature ranking we can select a subset of the features, allowing a trade-off between computation and accuracy. Using the label Fk to denote the set of k highest scoring features when trained using all features, Table 4 shows results with only F5 and F10 features. Although these results are less accurate than the full model, they can be calculated much faster. (But note that feature sets Fk as defined above are not necessarily the optimal set to use for training with k features.)

MTurk	MApp
Torralba saliency[1†]	Torralba saliency[2†]
RGB Green[3]	Coxel saliency[3]
Torralba saliency[2†]	RGB Probability (W=0)[1]
Itti saliency[1†]	RGB Probability (W=2)[3]
RGB Blue probability[3]	Text detector[1]
RGB Probability (W=4)[1]	RGB Green probability[1]
RGB Probability (W=2)[1]	Boundary object proposals[1]
Object proposals[3‡]	Hou saliency[3‡]
RGB Probability (W=16)[1]	Hou saliency[1†]
Distance to boundary[3]	Steerable Pyramids[2]

[1]mean [2]median [3]max

[†]attenuated with an inverted Gaussian around image center

[‡]attenuated with an inverted Gaussian around maximal value

Table 2. Feature selection. For each dataset we list the features with the highest mean feature weight across all experiments.

5. Evaluation

For evaluation, we plot a receiver operating characteristic (ROC) curve (true positive rate vs. false positive rate) and calculate the area under the curve (AUC) of the plot. We use a leave-one-out scheme, where all images but one are used for training and the one is used for testing. We repeat the leave-one-out experiment for each of our images, calculating the mean of all AUC values to produce a score.

Table 3 summarizes our results. A random baseline achieves a score of 0.5. We achieve an average AUC of 0.81 for the MTurk dataset and 0.84 for the MApp dataset. The LASSO algorithm allows us to learn which of the features are important (section 4.4) and we also report results using subsets of our features (Table 4). As expected, these results are not as good as the full model, but they are still useful when dealing with space or time constraints (less features directly translate to less memory and less computation time). We also report results for previous saliency methods as well as a simple adaptation to these methods where we multiply the saliency maps by an inverted Gaussian (as described in Sec. 4.2). This comparison is rather unfair since saliency methods try to predict all salient regions and not just distractors. However, the low scores for these methods show that indeed distractor prediction requires a new model based on new features.

Method	MTurk AUC	MApp AUC
Random	0.50	0.50
Saliency IV‡ [25]	0.55	0.56
Saliency I [20]	0.57	0.53
Saliency I‡ [20]	0.58	0.59
Saliency II [22]	0.59	0.57
Saliency II‡ [22]	0.59	0.57
Saliency III‡ [12]	0.62	0.59
Saliency III [12]	0.65	0.69
Saliency I† [20]	0.67	0.65
Saliency II† [22]	0.68	0.68
Saliency III† [12]	0.70	0.75
Saliency IV† [25]	0.72	0.72
Saliency IV [25]	0.74	0.76
Our method	**0.81**	**0.84**
Average Human	0.89	-

†attenuated with an inverted Gaussian around image center
‡attenuated with an inverted Gaussian around maximal value

Table 3. AUC scores. We compare against saliency prediction methods as published, and the same methods attenuated with an inverted Gaussian, as described in Section 4.2. Our method outperforms all others. As an upper bound we report average human score, which takes the average annotation as a predictor (per image, using a leave-one-out scheme). We also compared against individual features from Table 2 (not shown): all scores were lower than our method with a mean score of 0.59.

Method	Average # of used features	MTurk AUC
Ours (F-5 features)	3.40	0.72
Ours (F-10 features)	7.43	0.80
Ours (all features)	28.06	**0.81**

Table 4. AUC scores. Results using all 192 features and subsets F5 and F10 described in Section 4.4. Column 2 is the mean (over all experiments) of the number of features that were not zeroed out by the LASSO optimization. F10 produces a score similar to the full model, while using 5% of the features.

5.1. Inter-Dataset Validation

Using Mechanical Turk has the advantage of allowing us to get a lot of data quickly, but might be biased away from real-world scenarios. We wanted to make sure that our images and annotation procedure indeed match distractor removal "in the wild". For that purpose we also created dataset collected using our mobile app (MApp dataset), which contains real world examples of images with distractors that were actually removed by users. We performed inter-dataset tests: training on one dataset and testing on the other, the results are summarized in table 5. We show good results for training on MTurk and testing on MApp (0.86) and vice versa (0.78). The MApp dataset contains a single annotation per image (vs. 27.8 on average for the MTurk one). We therefore believe that the value 0.78 can be improved as the MApp dataset grows.

Train Dataset	Test Dataset	# of features	# of used features	AUC
MTurk	MApp	192	37	0.86
MApp	MTurk	192	25	0.78

Table 5. Inter-dataset AUC scores. We train on one dataset and test on the other, in order to validate that our MTurk dataset is similar enough to the real-world use cases in MApp to use for learning.

6. Applications

We propose a few different applications of distractor prediction. The most obvious application is automatic inpainting (Section 6.1), but the ability to identify distracting regions of an image can also be applied to down-weight the importance of regions for image retargeting (Section 6.2). We also posit that image aesthetics and automatic cropping can benefit from our method (Section 7).

6.1. Distractor Removal

The goal of distractor removal is to attenuate the distracting qualities of an image, to improve compositional clarity and focus on the main subject. For example, distracting regions can simply be inpainted with surrounding contents. To illustrate this application we created a simple interface with a single slider that allows the user to select a distractor threshold (figure 6). All segments are sorted according to their score and the selected threshold determines the number of segments being inpainted (figure 5). For a full demo of this system please see our supplementary video. Some before and after examples are shown in figure 7. We chose a rank order-based user interface as it is hard to find one threshold that would work well on all images, however we found that if distractors exist in an image they will corre-

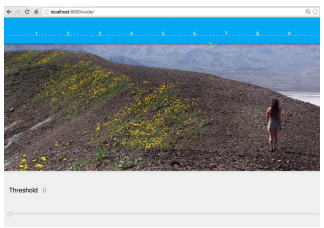

Figure 6. User interface for distractor removal. The user selects the amount of distracting segments they wish to remove. We refer the reader to the accompanying video for a more lively demonstration.

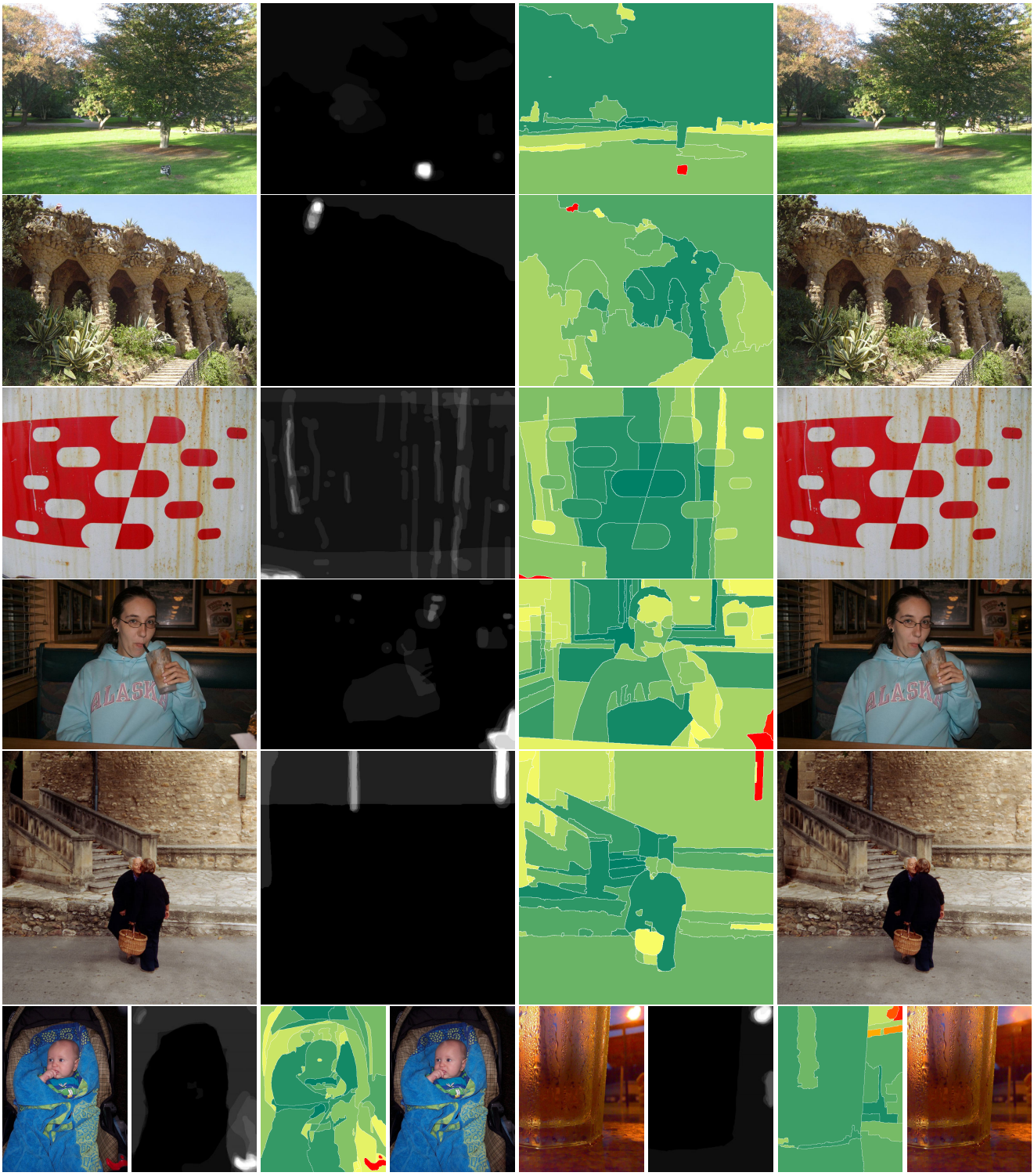

Figure 7. Examples of distractor removal results. Each quadruplet shows (from left to right): (1) Original image. (2) Normalized average ground-truth annotation. (3) Order of segments as predicted by our algorithm. (4) Distractor removal result. We urge the reader to zoom in or to look at the full resolution images available as supplementary material. The number of segments to remove was manually selected for each image. Segments are shown on a green-to-yellow scale, green being a lower score. Segment selected for removal are shown on an orange-to-red scale, red being a higher score. Notice how the red segments correlate with the ground-truth annotation. Also notice that we manage to detect a variety of distracting elements (a sign, a person, an abstract distractor in the corner, etc.)

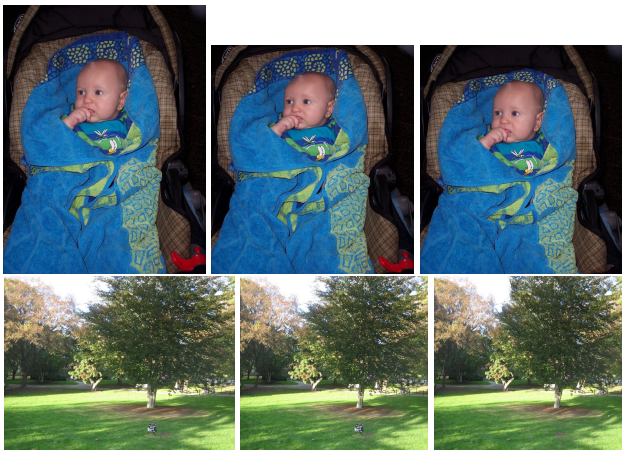

Figure 8. Image retargeting using seam carving [3]. From left to right: original image, retargeted image using a standard cost function (gradient magnitude), retargeted image using distractor cost function described in Section 6.2.

spond to the first few segments with the highest score.

6.2. Image Retargeting

Image retargeting is the task of changing the dimensions of an image, to be used in a new layout or on a different device. Many such methods have been developed in the past years [3, 26]. In addition to the input image and the desired output size, many of them can take an importance mask, which may be derived (based on image gradients, saliency prediction and gaze data) or provided by the user.

We can thus use our distractor prediction model to enhance a retargeting technique such as seam-carving [3]. For this application we view the distractor map as a complement to a saliency map: Whereas saliency maps give information regarding areas we would like to keep in the output image, a distractor map gives information regarding areas we would like to remove from the output. We thus calculate the gradient magnitudes of the image (G) and our distractor prediction map (D). Next, we invert the map ($D' = 1 - D$) and normalize for zero mean and unit standard deviation ($\hat{G}, \hat{D}'$). Our final map is $\hat{G} + \alpha\hat{D}'$. We use $\alpha = 1$.

Even this simple scheme produces good results in many cases. In figure 8, notice how the top-right image does not contain the red distractor and the bottom-right image does not contain the sign on the grass. (See figure 7 for the full distractor maps for these images.) However, we believe that a model which combines saliency and distractor maps will produce superior results. The creation of such a model is left for future work.

7. Conclusion and Future Work

We have acquired a dataset of distracting elements in images, used it to train a learning algorithm to predict such

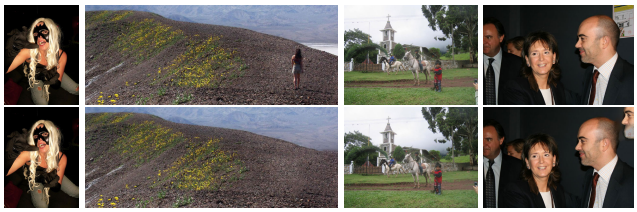

Figure 9. Two model failures, a segmentation failure, and an in-painting failure (see Section 7). Top row: original images. Bottom row: output images.

regions in novel images, and applied our predictor to a novel system that can in-paint distractors, removing them from an image with little or no user input.

Although our system shows great promise, there is plenty of room for improvement. Figure 9 illustrates several cases where our approach produces unsatisfactory results: The first two cases on the left illustrate a failure of our learned model. We predict the patch on the jeans of the main subject, and an entire person, even though they are critical parts of the main subject or the composition. The third example shows a segmentation failure, where only part of the arm at the lower right corner is removed. The last shows a removal-method failure, in which the sign behind the right person is correctly detected as distracting, but our patch-based hole filling method failed to remove it properly and instead duplicated the person.

Each of these failures suggests directions for future work. The first two cases suggest our model could be improved by using features related to image composition, a main subject detector, or relations between different elements in the image. The segmentation failure suggests focusing on improving the segmentation using the detected saliency. And of course, other image manipulations beyond patch-based hole filling [4] could be used to attenuate distractors like the last example: Since color saturation and contrast are key components of distractors, we can also consider removing them via de-saturation, exposure and contrast attenuation, blurring and various other methods. Implementing several removal methods and learning a model to automatically select the best one for a given distractor is an interesting direction for future work.

There are also additional applications of distractor prediction that we have not fully explored. For example, in addition to retargeting and inpainting, automatic cropping [34] could make use of distractor maps. However, since objects cut off at the edge of frame are often highly distracting, one would have to take into account the change in prediction that occurs as a result of the crop itself. One could also consider the use of distractor prediction as a cue for computational image aesthetics methods.

References

[1] H. Alers, H. Liu, J. Redi, and I. Heynderickx. Studying the effect of optimizing the image quality in saliency regions at the expense of background content. In *Proc. SPIE*, volume 7529, 2010. 2

[2] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 4

[3] S. Avidan and A. Shamir. Seam carving for content-aware image resizing. In *ACM Trans. on Graphics (Proc. SIGGRAPH)*, New York, NY, USA, 2007. ACM. 8

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, (3), Aug. 3, 8

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. 5

[6] K.-Y. Chang, T.-L. Liu, H.-T. Chen, and S.-H. Lai. Fusing generic objectness and visual saliency for salient object detection. In *International Conf. on Computer Vision (ICCV)*, 2011. 1

[7] M.-M. Cheng, G.-X. Zhang, N. Mitra, X. Huang, and S.-M. Hu. Global contrast based salient region detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 409–416, June 2011. 1

[8] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *International Conf. on Computer Vision (ICCV)*, pages 633–640, 2013. 2

[9] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, Dec. 2005. 5

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2010. 4

[11] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in Neural Information Processing Systems*, pages 545–552. MIT Press, 2007. 1

[12] X. Hou and L. Zhang. Saliency Detection: A Spectral Residual Approach. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2007. 4, 6

[13] L. Itti and C. Koch. A Saliency-Based Search Mechanism for Overt and Covert Shifts of Visual Attention. *Vision Research*, 40:1489–1506, 2000. 1, 4

[14] L. Joyeux, O. Buisson, B. Besserer, and S. Boukir. Detection and removal of line scratches in motion picture films. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 548–553, 1999. 2

[15] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *International Conf. on Computer Vision (ICCV)*, 2009. 1, 2, 4

[16] Y. Ke, X. Tang, and F. Jing. The design of high-level features for photo quality assessment. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 419–426, 2006. 2

[17] A. C. Kokaram. On missing data treatment for degraded video and film archives: A survey and a new bayesian approach. *IEEE Trans. on Image Processing*, 13(3):397–415, Mar. 2004. 2

[18] H. Liu and I. Heynderickx. Studying the added value of visual attention in objective image quality metrics based on eye movement data. In *IEEE International Conf. on Image Processing (ICIP)*, Nov. 2009. 2

[19] W. Luo, X. Wang, and X. Tang. Content-based photo quality assessment. In *International Conf. on Computer Vision (ICCV)*, Nov 2011. 2

[20] R. Mairon and O. Ben-Shahar. A closer look at context: From coxels to the contextual emergence of object saliency. In *European Conf. on Computer Vision (ECCV)*. 2014. 1, 4, 6

[21] L. Marchesotti, C. Cifarelli, and G. Csurka. A framework for visual saliency detection with applications to image thumbnailing. In *International Conf. on Computer Vision (ICCV)*, pages 2232–2239, 2009. 2

[22] R. Margolin, A. Tal, and L. Zelnik-Manor. What Makes a Patch Distinct? *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1139–1146, June 2013. 1, 4, 6

[23] N. Murray, L. Marchesotti, and F. Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. (In press). 2

[24] L. Neumann and J. Matas. Real-time scene text localization and recognition. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012. 4

[25] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001. 4, 6

[26] M. Rubinstein, D. Gutierrez, O. Sorkine, and A. Shamir. A comparative study of image retargeting. *ACM Transactions on Graphics*, 29(6):160:1–160:10, Dec. 2010. 8

[27] E. Simoncelli and W. Freeman. The Steerable Pyramid: A Flexible Architecture For Multi-Scale Derivative Computation. *IEEE International Conf. on Image Processing (ICIP)*, 1995. 4

[28] S. L. Su, F. Durand, and M. Agrawala. De-emphasis of distracting image regions using texture power maps. In *Applied Perception in Graphics & Visualization*, pages 164–164, New York, NY, USA, 2005. ACM. 2

[29] B. Suh, H. Ling, B. B. Bederson, and D. W. Jacobs. Automatic thumbnail cropping and its effectiveness. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 95–104, New York, NY, USA, 2003. ACM. 2

[30] X. Tang, W. Luo, and X. Wang. Content-Based Photo Quality Assessment. *IEEE Transactions on Multimedia (TMM)*, 2013. 2

[31] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. 4, 5

[32] W.-S. Tong, C.-K. Tang, M. S. Brown, and Y.-Q. Xu. Example-based cosmetic transfer. *Computer Graphics and Applications, Pacific Conference on*, 0:211–218, 2007. 2

[33] P. Viola and M. Jones. Robust Real-time Object Detection. In *International Journal of Computer Vision (IJCV)*, 2001. 4

[34] J. Yan, S. Lin, S. Bing Kang, and X. Tang. Learning the change for automatic image cropping. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 2, 8

[35] C. L. Zitnick and P. Dollár. Edge Boxes: Locating Object Proposals from Edges. In *European Conf. on Computer Vision (ECCV)*, 2014. 4