Learning How to Inpaint from Global Image Statistics

Anat Levin Assaf Zomet Yair Weiss
School of Computer Science and Engineering,
The Hebrew University of Jerusalem,
91904, Jerusalem, Israel
E-Mail: {alevin,zomet,yweiss}@cs.huji.ac.il

Abstract

Inpainting is the problem of filling-in holes in images. Considerable progress has been made by techniques that use the immediate boundary of the hole and some prior information on images to solve this problem. These algorithms successfully solve the local *inpainting problem but they must, by definition, give the same completion to any two holes that have the same boundary, even when the rest of the image is vastly different.*

In this paper we address a different, more global *inpainting problem. How can we use the rest of the image in order to learn how to inpaint? We approach this problem from the context of statistical learning. Given a training image we build an exponential family distribution over images that is based on the histograms of local features. We then use this image specific distribution to inpaint the hole by finding the most probable image given the boundary and the distribution. The optimization is done using loopy belief propagation. We show that our method can successfully complete holes while taking into account the specific image statistics. In particular it can give vastly different completions even when the local neighborhoods are identical.*

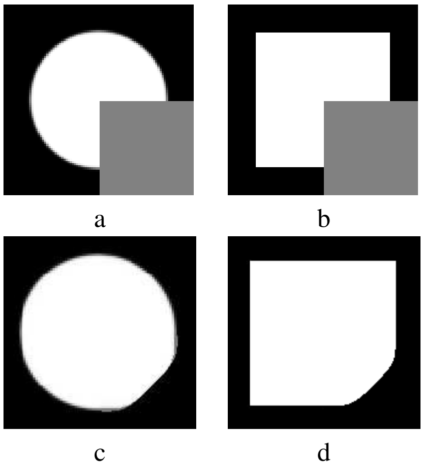

Figure 1. **a.-b.** Two images with holes. In both cases the boundary of the holes are identical thus local inpainting algorithms would complete them identically. **c.-d.** The results of the algorithm in [3] run with a single resolution. As can be expected from a local algorithm, the completion is identical. In this paper we ask: how can we use the global information in the image to cause the completions to be different?

1. Introduction

Inpainting, dis-occlusion and filling-in are various names for the same task: Given an image with a missing region (a hole), restore the values in the hole in an undetectable way [3]. Applications include restoration of old images, removal of overlaid text and logos and removal of objects from images.

The inpainting problem is clearly ill-posed. Any method must therefore use some prior assumptions about the unknown missing values and their relations with the known hole neighborhood. Most existing approaches (see e.g. [17] for a review) use a generic prior on images (e.g. high smoothness, low total variation or low curvature) and use an optimization to find the most probable completion given the prior model and the immediate boundary of the hole. We call these approaches *local* inpainting algorithms.

Despite the impressive successes of local approaches, they must by definition give identical completions when the immediate boundary of the hole is identical. Consider Fig. 1-a,b: two images of a square and a circle, each with a missing square region on the bottom-right part. While the circle and square are different, the small neighborhoods around the holes are identical. Indeed, up to numerical error, the gradients and gray levels in the immediate boundary of the hole, are identical. Thus any algorithm that is based on a generic model and these boundary conditions will restore the two holes identically. Figs 1-c,d show the results

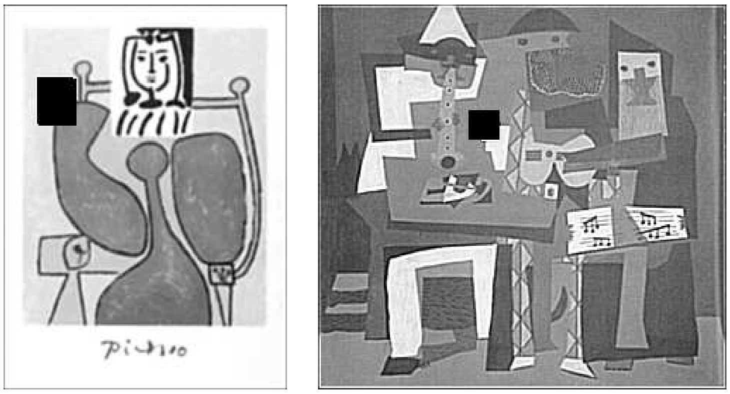

Figure 2. Two Pablo Picasso paintings with holes. The vicinity of the hole is nearly identical in the two paintings but the global style is vastly different. We would like the completion to be different.

of the local algorithm in [3][1]. While the completions are very reasonable given the local information, they do not appear "perceptually correct". Evidently our visual system is taking more global information into account.

The global statistics are similarly important in painting restoration, we would want to restore a hole in a Mondrian painting very differently from a locally identical hole in a Bruegel painting (indeed even a Picasso from the blue period should be restored completely differently from a Picasso in the cubist period). Figure 2 shows an example. The immediate vicinity of the two holes are very similar but the global statistics are very different: the cubist painting has a different "look" compared to the Blue period painting and we want inpainting algorithms to preserve this look. We call this problem "learning how to inpaint".

In this paper we address this problem in the context of statistical learning. We use the input image to learn a probability distribution over images that is in the exponential family. We then use loopy belief propagation to find the most probable completion of the hole under this learned distribution.

1.1 Previous work

In most existing inpainting approaches the hole is estimated as the most "smooth" continuation of the local structure of the image, where smoothness can be defined in different ways. This main advantage of this approach is that when properly formulated it can be applied to an arbitrary image patch with minimal user interaction. A smooth continuation can be defined in various ways. Bertalmio et. al. [3], inspired by professional art restorators , propagate gradient direction and gray values from the surrounding neighborhood into the hole. They formulate the process elegantly in a PDE framework, and solve it using fast iterative solvers.

In a later paper [1], they use similar ideas, but reformulate the inpainting process in a variational framework. They propose to minimize the following cost function over the image I and its normal field α:

$$J = \int_{x,y} |div(\alpha)|^p (a + b|\nabla I|) + \gamma \int_{x,y} (|\nabla I| - \alpha \cdot \nabla I) \quad (1)$$

The $|div(\alpha)|$ term penalizes curvature, the $|\nabla I|$ term penalizes large gradients and the second integral is a relaxation of the constraint that the α is indeed the normal field of I. As we discuss in the next section, the impressive results in [1] using only local image operators motivated us to use only local image statistics to define our prior.

Another way to define a smooth filling-in was presented by Chan and Shen [4], who minimized the total variation in the result image. As mentioned in [1], such approach handles noise very well, but tends to complete straight lines.

Filling-in of holes is also performed by texture synthesis algorithms where it is assumed that the missing data is part of a (usually homogeneous) texture. The region can be filled-in by a texture synthesis engine, e.g. [16, 8, 22, 9, 6]. The texture-synthesis approach can process large holes, and fill them with rich structures learned from similar regions in the image. We found two application of texture synthesis to image inpainting. Hirani and Totsuka [13] fill in a selected texture by combining spectral and spatial information, achieving impressive results. Criminisi et. al. used an exampler-based approach, adapting the synthesis method of Efros and Leung [9] to image inpainting [5]. Recent works combine texture synthesis with inpainting of structure [2].

Filling-in of holes in simple images consisting of a single contour has also been extensively studied in the human vision literature [19, 21]. These approaches also rely on a notion of smooth continuation, but in general they cannot be applied to an arbitrary image patch in a natural image.

2 Exponential family models of image statistics

In order to learn how to inpaint, we want a method that will capture the "look" of a training image in a probability distribution over images. The main challenge is to estimate the parameters of such a distribution from a single image. Obviously, if one represented image patch statistics using a huge look up table of all possible $20x20$ image patches and their respective probabilities then learning how to inpaint would be trivial but estimating such a table is, of course, impossible.

Our approach is motivated by the success of *exponential family* distributions in the modeling of images and natural language [22, 7, 12, 14]. In this approach the probability of

[1]We thank Joan Verdera for providing these results. The results are with a single resolution. Multi resolution results may be different for the two figures

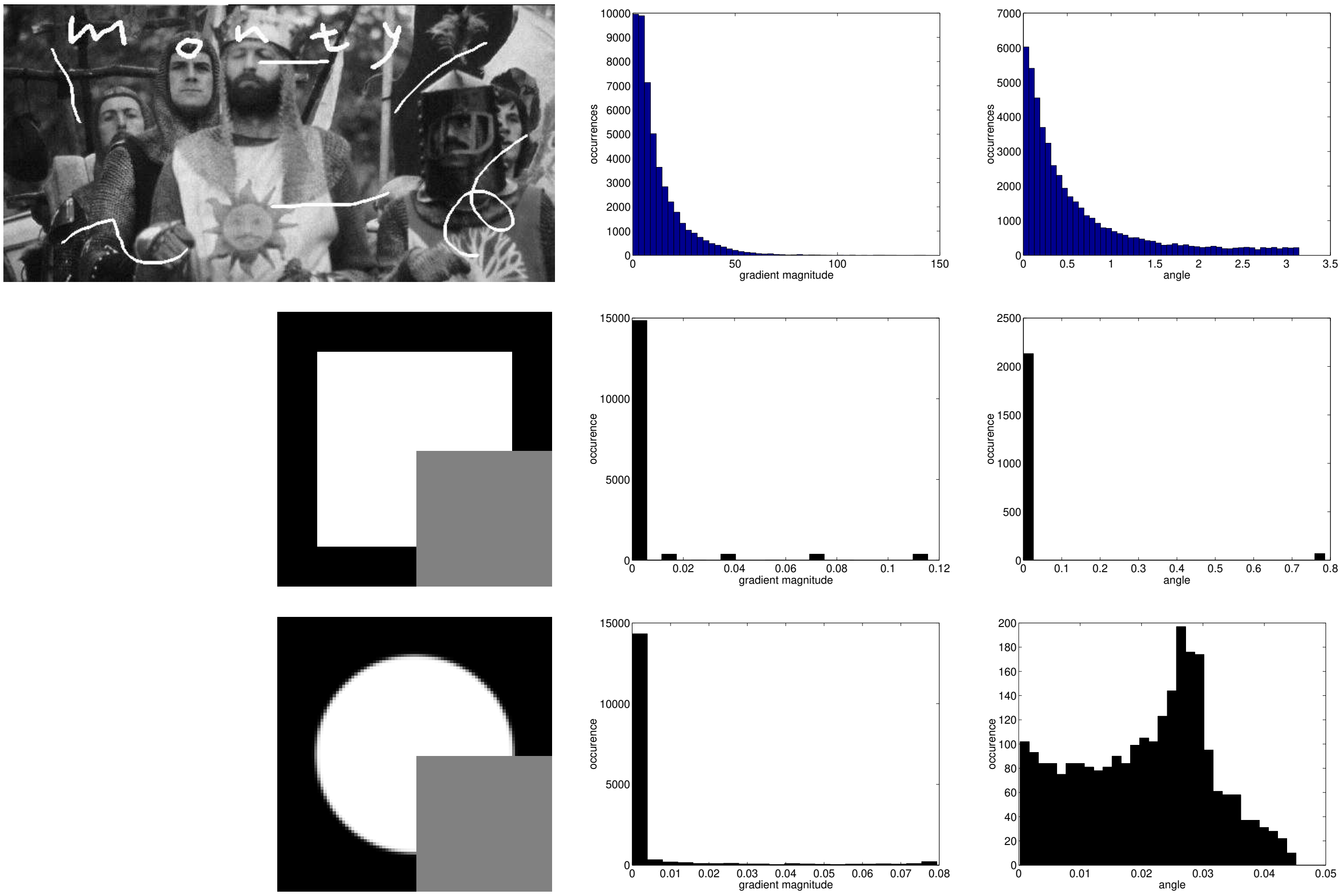

Figure 3. The marginal statistics of the features we are using: gradient magnitude (center) and relative gradient angle (right). These simple, local histograms capture the different "look" of the three images. Note that the axis limits in the angle histograms are vastly different for the square and the circle: in the circle all angular differences are between 0 and 0.05.

an image is defined by means of a small number of *sufficient statistics* or features, each of which can be evaluated at an arbitrary location in the image. The probability of an image is given by:

$$P(I; \{\Phi_i\}) = \frac{1}{Z} e^{\sum_i \sum_{x,y} \Phi_i(f_i(x,y))} \qquad (2)$$

Where $f_i(x, y)$ is the value of feature i at location (x, y) in the image I, and Z is a normalization factor. The parameters of an exponential family distribution are the choice of features and the tables $\Phi_i(f)$: intuitively the larger the average value of $\Phi_i(f)$ for a particular image, the more probable the image is.

Learning exponential family distributions from data is a difficult but well studied problem. Typically, one chooses the features based on a notion of informativeness and then chooses the tables based on the maximum likelihood or maximum entropy criterion [22, 7]. For exponential family distributions these criteria are identical and simplify to a requirement that the predicted marginals of all features match the empirical marginals in the data:

$$P(F_i = s; \{\Phi\}) = \hat{P}(F_i = s) \qquad (3)$$

The computational difficulty arises from the left hand side of equation 3: calculating the predicted marginals requires summing over all possible images, and while Monte-Carlo methods can be applied they are still too slow for our application where the model is learned anew for every image that we process.

We simplify the learning in two ways. First, the features in the distribution are fixed for all images. Motivated by previous work on natural image statistics [15, 18] and the work of [1] we chose two features that we thought would be informative: gradient magnitude $F_1(x, y) = |\nabla I(x, y)|$ and pairwise gradient angle. Pairwise gradient angle is defined for every pair of neighboring pixels $(x_1, y_1), (x_2, y_2)$.

Denote the gradients $g_1 = \nabla I(x_1, y_1)$, $g_2 = \nabla I(x_2, y_2)$. We define the angle θ_{12} between the two gradients as $\theta_{12} = \cos^{-1}(g1^T g2/|g1||g_2|)$ and set:

$$F_2(x_1, y_1, x_2, y_2) = \theta_{12} \tag{4}$$

Note that this feature is measured for all pairs of neighboring pixels: we do not assume that we can decide which pairs of gradients belong to the same curve. Since the angle of the gradient is noisy at gradients of low magnitude, we only measure the angle feature at gradients above a certain threshold. Figure 3 shows the marginal histograms of these two features on a natural image and the two synthetic images described earlier. Note that the different histograms capture the different "looks" of the three images. The fact that the natural image is more textured than the synthetic images is captured in the magnitude histogram: there are much more nonzero gradients in the real image. The difference between the circle and the square is captured in the relative angle histogram. In the square the distribution is bimodal, indicating that adjacent pixels either have identical angle (i.e. along straight lines) or have a very different angle (i.e. at sharp corners). In the circle, adjacent pixels always have similar angle (i.e. there are no sharp corners). (Note that the axis limits in the angle histograms are vastly different for the square and the circle: in the circle all angular differences are between 0 and 0.05).

The second simplification we make is the method by which we estimate Φ. To understand the complexity of solving equation 3 note that equation 2 with our choice of features is equivalent to a Markov Random Field distribution on the gradient field, $g(I)$ of an image I:

$$P(g(I)) = \frac{1}{Z} \prod_i \Psi_i(g_i) \prod_{<ij>} \Psi_{ij}(g_i, g_j) \tag{5}$$

where $< ij >$ refers to pairs of neighboring pixels in the image and $\Psi_i(g_i) = \exp(\Phi_1(|g_i|))$ and $\Psi_{ij}(g_i, g_j) = \exp(\Phi_2(\theta_{ij})$. Thus estimating Φ_{ij} is as difficult as estimating the potential functions in a MRF.

Since exact ML estimation of the potentials of the MRF is intractable, a number of approximations have been proposed. We used an approximation similar to the one used in [10] where the potentials are approximated by the empirical marginal and conditional probabilities. Specifically we set:

$$\Psi_i(g_i) = \hat{P}(|g_i|) \tag{6}$$
$$\Psi_{ij}(g_i, g_j) = \hat{P}(\theta_{ij}) \tag{7}$$

It is easy to show that when the MRF is singly connected these settings will give rise to Φ tables that exactly satisfy the maximum likelihood equation (eq. 3). Thus if we wanted to model a single scan line of an image, these parameters would be exact. In our case, of course, the graph has many loops so these parameters are *not* the maximum likelihood parameters, but this approximation has proven to be successful in a number of vision applications [10].

The only tweakable parameter in our model is the definition of the region over which the histograms are calculated. This can either be the entire image, a subregion of the input image or even a different image that has a similar "look" to the one the user wants.

To summarize: in order to fill in a hole in the image, we first measure histograms of our features over the training image, and then search for an integrable gradient field that agrees with the image gradients on the boundary of the hole and maximizes the probability defined by equations 5,6.

3 Optimization using loopy Belief Propagation

In order to find the most probable filling-in we need to optimize the probability, conditioned on the hole boundary B:

$$P(g|B) = \frac{1}{Z_2} \prod_i \Psi_i(g_i) \prod_{<ij>} \Psi_{ij}(g_i, g_j) \prod_{<ijk>} \Psi_{ijk}(g_i, g_j, g_k) \tag{8}$$

where Z_2 is a normalization factor and Ψ_{ijk} enforces integrability of the gradient field (differentiating the x derivative with respect to y should give the same answer as differentiating the y derivative with respect to x). The product is taken over all gradients inside the hole and those in the boundary, and gradients on the boundary are fixed to their observed values.

Naive optimization of equation 8 is of course exponential in the size of the hole. The max-product belief propagation algorithm [20] is a local, message passing algorithm that can be used to perform the optimization. It is guaranteed to find the global optimum when the graph has no loops. In our case, the graphical model defined by equation 8 has many loops. Nevertheless motivated by the recent results on similar graphs [10, 11] we expected good results for our problem. When it converges, the max-product algorithm is guaranteed to find a gradient field $\{g_i\}$ that is a local maximum of equation 8 with respect to a large neighborhood [20]. Finally, given the gradient field we integrate it by robustly solving the following linear equation:

$$Dx = g \tag{9}$$

where x is a vectorized version of the image, D is the differentiation matrix of size $2n \times n$ (where n is the number of pixels) and g is a vectorized version of the gradient field. This is an overconstrained set of equations and we find the solution that minimizes the ℓ_1 norm using linear programming.

In order to run the max-product belief propagation algorithm one needs to discretize the gradient field. We used

120 candidate gradients chosen using a clustering algorithm from the gradients of the input image surrounding the hole.

4 Experiments

In the previous section, we saw that the marginal statistics of the square and circle are quite different. Can our algorithm use this difference to correctly learn how to inpaint?

The results are shown in figure 4. The training image was the full image. To avoid aliasing artifacts we anti-aliased both figures. Since the number of distinct gradients in the areas surrounding the holes here are quite small, we augmented the discretization with an additional set of gradients chosen to tile the space of orientations. The same discrete set of gradients was used for both images. Note that the algorithm correctly adapts to the particular image and completes a circle in one case and a square in the second case, despite the fact that the local neighborhoods are identical. This is due to the vastly different relative angle histograms in the two images. For the square image, the relative angle is almost always zero, except for a few instances where the relative angle is large. For the circle image, on the other hand, the relative angle is typically small but nonzero. The right column shows what happens when each image is filled-in based on marginal statistics from the other image.

Figure 5 shows the output of our algorithm on the two Pablo Picasso paintings with different styles. For each image, we used a small patch from that image to estimate the histograms. To avoid aliasing artifacts the images are smoothed. While the holes in the two images are not identical, they are very similar locally. Yet our algorithm completes sharp corners for the cubist painting and smooth curves for the blue period painting.

Can we learn how to inpaint in real images ? Figure 6 shows a result. We trained the algorithm on an urban scene and a fruit image (top row). We then used the urban scene statistics to fill-in the holes in the ruins image and the fruit scene statistics to fill-in the fruit image. As a result, our algorithm completes sharp corners on the ruins but smooth curves on the fruit. Again, note that the boundary of the hole in the fruit image is very similar (up to rotation) to the boundaries of the top two holes in the ruins image. Thus classical inpainting approaches should give similar results in the two cases.

In the previous two examples, our algorithm can successfully adapt to the image statistics and give completions that depend not only on the local boundary of the hole but also on the global "look" of the image. This is in contrast to existing inpainting approaches that give identical completions when the boundary is identical. However, given the success of existing approaches in many images, one wonders: will our algorithm cause a decrease in performance in images where the boundary information is sufficient?

Figure 7 shows that our algorithm also performs well in the cases where classical algorithms perform well. In fact, the completion is very similar to the one calculated by [3]. This is because the marginal statistics in this image (figure 3) favor "smooth" completions: since both the magnitude histogram and the angle histogram are peaked at zero, the learned distribution favors completions with short lines and low curvature.

5 Discussion

Inpainting is obviously an ill-posed problem and hence is impossible without some assumption about the statistics of images. In that sense, all previous approaches to inpainting can also be viewed as having an implicit probabilistic model of images: e.g. that images tend to be smooth [3] or that images tend to contain homogeneous texture [8]. In this work, we have asked: how can we learn this probabilistic assumption from the input image? We have shown that a model based on histograms of local features can capture the "look" of an image and shown how to use these histograms to define the filling in of a hole.

In future work we would like to extend our model to use more image features. In particular the probability model presented in this paper does a poor job of representing texture. It would be interesting to see whether a small number of additional features (e.g. the ones used in [16]) would enable our algorithm to inpaint textured regions. A promising direction to explore is to choose the features anew for the particular image that needs to be inpainted. This would require efficient approximations to the Minimax approach used in [22]. An alternative way to use our method in textured images is to use the decomposition into a "structure" image and a "texture image" proposed in [2] and apply our current method only to the structure image.

In future work we would also like to explore other optimization methods. While our framework is probabilistic, we are interested only in the most probable completion so calculation of marginal probabilities is not necessary. We are interested in exploring some of the powerful optimization techniques used in local inpainting to find the most probable completion given our model. In particular, our current optimization algorithm is single scale and hence our completions are not as sharp as the state of the art local algorithms. One approach worth exploring is to initialize the local algorithms with our completion and thus obtain a sharper image.

The challenge of defining simple probability models that capture the "look" of an image is common to a large number of problems in vision and image processing including superresolution, denoising, transparency analysis and more. We believe that progress in learning how to inpaint will directly translate into progress in these additional domains.

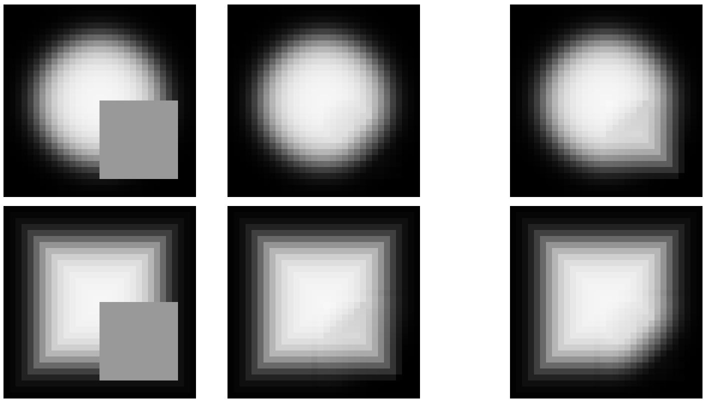

Figure 4. Filling in a circle and a rectangle. The completion depends on the marginal statistics. *Middle column:* Each figure is completed based on marginal statistics taken from the same image. The circle completion is curved and the square completion has a sharp corner. *Right column:* Each image is completed based on marginal statistics taken from the other image.

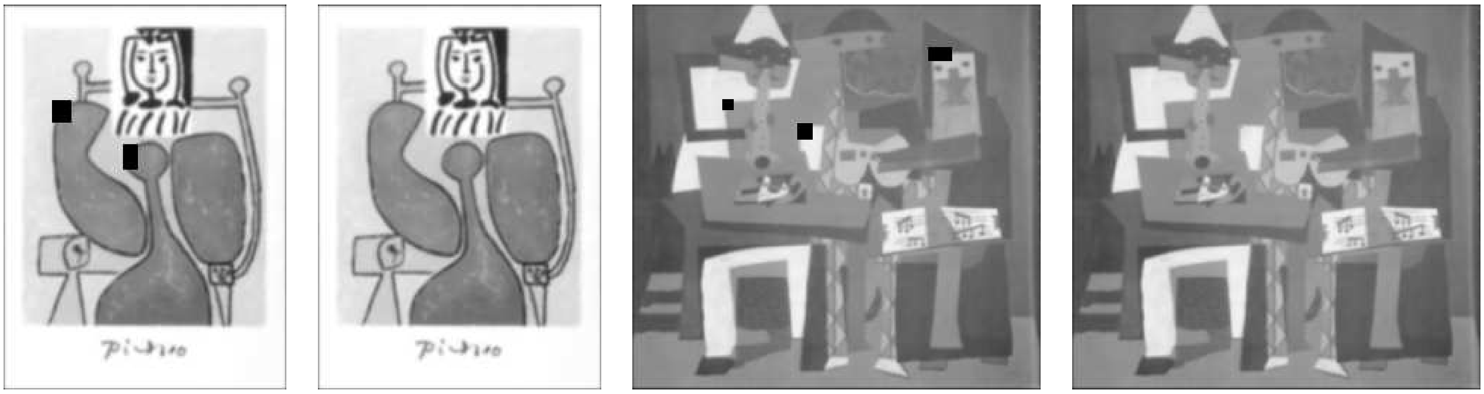

Figure 5. The output of our algorithm on the two Pablo Picasso paintings with different styles. The algorithm completes sharp corners for the cubist painting and smooth curves for the blue period painting.

References

[1] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Trans. Image Processing*, 10(1), August 2001.

[2] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[3] M. Bertalmo, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of SIGGRAPH*, July 2000.

[4] T. Chan and J. Shen. Mathematical models for local non-texture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002.

[5] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages II:721–728, 2003.

[6] J. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH*, pages 361–368, August 1997.

[7] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. PAMI*, 19(4):380–393, 1997.

[8] A. Efros and W. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH*, pages 341–346, August 2001.

[9] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Int. Conf. on Computer Vision*, pages 1033–1038, 1999.

[10] W. Freeman and E. Pasztor. Learning to estimate scenes from images. In M. Kearns, S. Solla, and D. Cohn, editors, *Adv. Neural Information Processing Systems 11*. MIT Press, 1999.

[11] B. Frey, R. Koetter, and N. Petrovic. Very loopy belief propagation for unwrapping phase images. In *Adv. Neural Information Processing Systems 14*. 2001.

[12] D. Heeger and J. Bergen. Pyramid-based texture analsysis and synthesis. *Computer Graphics*, 1995.

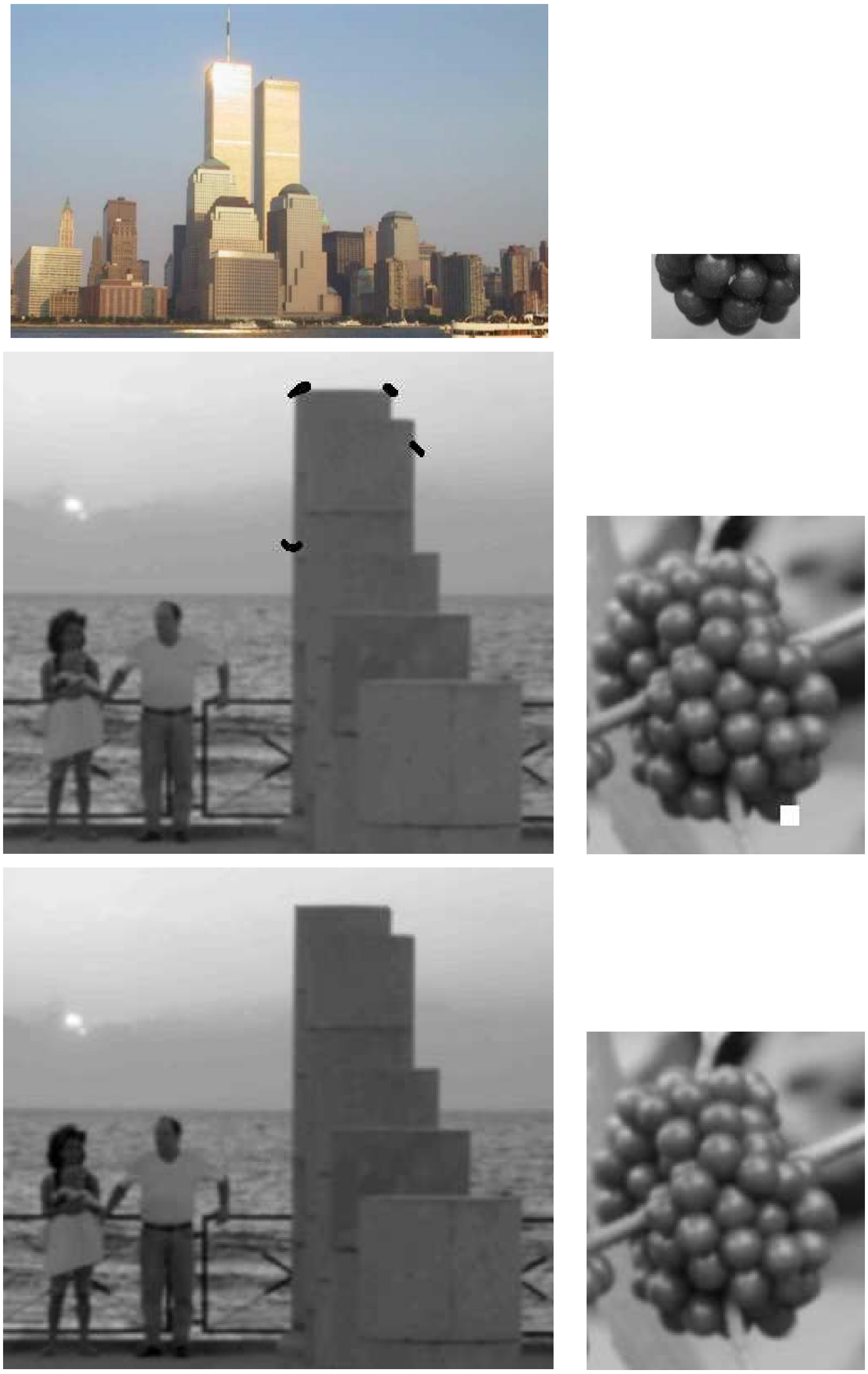

Figure 6. Filling in two images with different statistics. *Top:* Two different training images. In the urban scene there are many corners while in the fruit scene most curves are smooth. *Middle:* Two images to be filled in. We would like holes to be filled in differently even though the boundaries of the top two holes in the ruins image are very similar to the boundary of the hole in the fruit image. *Bottom:* The results of our algorithm. The algorithm trained on an urban scene completes sharp corners while the one trained on fruit completes smooth curves.

Figure 7. Results of our algorithm on the Monty Python image.

[13] A. Hirani and T. Totsuka. Combining frequency and spatial domain information for fast interactive image noise removal. In *Proceedings of SIGGRAPH*, pages 269–276, August 1996.

[14] A. Levin, A. Zomet, and Y. Weiss. Learning to perceive transparency from the statistics of natural scenes. In *Advances in Neural Information Processing Systems*, volume 15. The MIT Press, 2002.

[15] B. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–608, 1996.

[16] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int'l J. Comput. Vision*, 40(1):49–71, 2000.

[17] J. Shen. Inpainting and the fundamental problem of image processing. *SIAM news*, 2003.

[18] E. Simoncelli. Statistical models for images:compression restoration and synthesis. In *Proc Asilomar Conference on Signals, Systems and Computers*, pages 673–678, 1997.

[19] S. Ullman. filling in the gaps: the shape of subjective contours and a model for their generation. *Biological Cybernetics*, 25, 1976.

[20] Y. Weiss and W. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):723–735, 2001.

[21] L. Williams and K. Thornber. Orientation, scale, and discontinuity as emergent properties of illusory contour shape. *Neural Computation*, 13(8), 2001.

[22] S. C. Zhu, Z. N. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.