Quality Prediction for Image Completion

Johannes Kopf, Microsoft Research
Wolf Kienzle, Microsoft Research
Steven Drucker, Microsoft Research
Sing Bing Kang, Microsoft Research

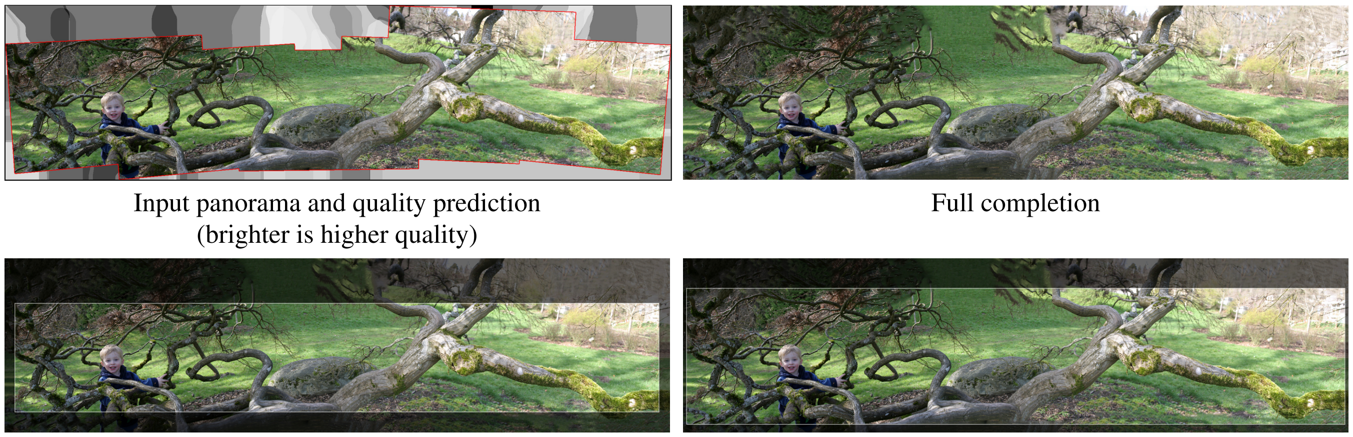

Input panorama and quality prediction (brighter is higher quality)
Full completion
Best crop using only "known" pixels ("conservative crop")
Our optimized crop based on quality prediction

Figure 1: *Our data-driven technique is capable of predicting image completion quality (top left) before the completion is actually computed (top right). Based on our prediction, we compute an optimal crop rectangle that tries to include as many known pixels as possible while avoiding low-quality regions (bottom right). Compared to previous cropping approaches that do not fill in (bottom left) we can usually include a larger amount of the input image in our crop. Our algorithm* only *completes the cropped region, thus saving a significant amount of computation compared to full completion.*

Abstract

We present a data-driven method to predict the quality of an image completion method. Our method is based on the state-of-the-art non-parametric framework of Wexler *et al.* [2007]. It uses automatically derived search space constraints for patch source regions, which lead to improved texture synthesis and semantically more plausible results. These constraints also facilitate performance prediction by allowing us to correlate output quality against features of possible regions used for synthesis. We use our algorithm to first crop and then complete stitched panoramas. Our predictive ability is used to find an optimal crop shape *before* the completion is computed, potentially saving significant amounts of computation. Our optimized crop includes as much of the original panorama as possible while avoiding regions that can be less successfully filled in. Our predictor can also be applied for hole filling in the interior of images. In addition to extensive comparative results, we ran several user studies validating our predictive feature, good relative quality of our results against those of other state-of-the-art algorithms, and our automatic cropping algorithm.

Links: DL PDF WEB

1 Introduction

Image completion or inpainting is a popular image editing tool for object removal and replacement or digital photograph restoration. A variety of completion algorithms have been developed in the scientific community over the past decade, and image completion is now a feature in commercial photo editing software such as Adobe Photoshop. In most previous work, image completion is used to fill holes after unwanted objects are removed. The same algorithms, however, can also be used to extend an image beyond its original boundaries. This is useful for filling beyond the irregular boundaries of a stitched panorama—this application is the focus of this paper.

Casually shot panoramas often have irregular boundaries (e.g., Figure 1). Most users, however, prefer output images with rectangular boundaries. The trivial solution implemented by most stitching software is to crop to the largest box that is fully contained within the panorama. This simple method often removes large parts of the input. The alternative is to apply any existing completion algorithm to fill the missing regions of the panorama bounding box.

Unfortunately, all existing image completion algorithms fail on occasion; the failure typically shows up as either inability to synthesize some textures well or results that are semantically implausible (see Figure 2). In addition, it is difficult to anticipate when and where such algorithms will fail given an arbitrary input image.

In this paper, we use machine learning to predict the quality of image completion. To support this prediction, we design our image completion algorithm to produce high quality results and to allow associations between completed pixels and known pixels to be created. We build on the existing non-parametric optimization framework of Wexler *et al.* [2007] (which is also implemented in the *Content Aware Fill* feature of Adobe Photoshop[1]). Previous work showed that this algorithm performs best if the source locations

[1] see http://www.adobe.com/technology/projects/content-aware-fill.html

for patches are constrained to certain areas, e.g., see [Barnes et al. 2009]. We use a heuristic that automatically derives search space constraints based on overlapping texture segmentation. These constraints allow us to design a simple method to predict algorithm performance. Based on the prediction we compute a crop shape *before* the completion is actually carried out, avoiding unnecessarily completing cropped pixels.

To validate and train our prediction function, we ran a Mechanical Turk user study to obtain about 9,500 "good" / "bad" labels on crops from completed images from a large number of subjects. These labels are used to estimate our prediction function via cross-validation. We tested our algorithm on an extensive collection of input images, and in another user study, compared our results with those of various state-of-the-art completion algorithms. In yet another user study, we evaluated the performance of our automatic crop optimization. In addition to the examples included in this paper, all of our results and comparisons are included in the supplementary material.

2 Previous Work

While there is a substantial amount of previous work on image completion (also referred to as inpainting or image filling), to the best of our knowledge, we are not aware of any methods that can predict quality *before* completion. Being able to predict quality would allow the system to determine if the input image can be properly restored or edited, or in our case, estimate the desired crop of a panorama. As a result, the system completes only what is needed. Note that existing cropping techniques such as that of [Zhang et al. 2005] use only available image data for cropping.

Texture quality assessment techniques are the closest approaches to ours. The objective function of Kwatra *et al.* [2005] measures similarity of local patches to the target texture, but this measure is a poor predictor of quality (as shown in Section 5). Swarmy *et al.* [2011] show that a linear combination of image parameters (such as intensity mean, variance, entropy, and band information) can be used to assess texture quality. In both cases, however, the texture has to be generated first.

We now briefly survey representative methods for image completion, which we classify as primarily example-based or diffusion-based. Komodakis and Tziritas [2007] provided an excellent review of inpainting techniques.

Diffusion-based techniques, often referred to as "inpainting," typically work well for small or narrow holes, e.g., for removing scratches from a scanned old photograph. They are less appropriate for completing stitched panoramas with large open-ended missing regions due to their inability to synthesize textures. One good representative of diffusion-based techniques is that of Bertalmio *et al.* [2000]; it prolongs isophote lines in the missing areas.

Example-based techniques tend to be more effective for filling larger holes than diffusion-based techniques. Efros and Leung [1999] popularized the use of non-parametric sampling for texture synthesis (and by extension, image completion). Many example-based techniques are based on this core concept. Representative techniques include block-based matching with structure-based priority [Criminisi et al. 2003], use of hierarchical filtering as initialization and adaptive regions instead of patches [Drori et al. 2003], optimization of a texture energy function [Wexler et al. 2007; Kwatra et al. 2005; Darabi et al. 2012], application of user specification of structure [Sun et al. 2005], MRF with exemplars as sample labels [Komodakis and Tziritas 2007], and search for globally-transformed patches [Mansfield et al. 2011].

Kawai *et al.* [2008] adapted the work of Wexler *et al.* [2007] using an SSD-based objective function to handle regular (fine-grained) textures. In addition, they compensate for local brightness changes by linearly fitting the intensity of the matched patch.

Pritch *et al.* [2009] cast the problem of image synthesis as finding an optimal shift-map of pixels based on global factors (such as image size and object arrangement) and local features (such as saliency map) as well as spatial regularization. They have demonstrated their technique on image retargeting, rearrangement, and completion.

Many of the previous methods rely on quickly finding similar image patches. The recent PatchMatch algorithm [Barnes et al. 2009] greatly improves their speed by replacing the previously employed tree-based search techniques with a much faster randomized algorithm.

Practically all the automatic techniques that rely on exemplars (with the exception of [Matsushita et al. 2006]) do not restrict their search. As a result, such techniques are prone to the same problems as those of [Wexler et al. 2007], where perceptually implausible patches could be used for image completion. In addition, they were not designed with performance prediction in mind. It is difficult to anticipate the degree of success or failure within the missing regions.

3 Overview

Our algorithm builds on the non-parametric optimization algorithm introduced by Wexler *et al.* [2007]. We implemented the optimization as described in that paper using the weighted average updating rule (we did not find it necessary to use the much slower mean-shift based updating rule). We describe more implementation details of the algorithm in a supplementary document.

We bias our algorithm toward continuing image content near the boundary into the missing region. Each missing pixel may only be filled from a tightly constrained part of the known region. Knowing in advance where every missing pixel may come from enables us to predict the perceived quality of the completed result. Using training data, we learn a function that maps low-level features of the closest known image regions to the perceived quality of the completed result. The low-level features include color, edge density, edge orientation, contour length, and region size. Our prediction function is learnt and validated from data that we collected in a Mechanical Turk user study, where subjects were asked to categorize random patches from the completed regions as "good" or "bad".

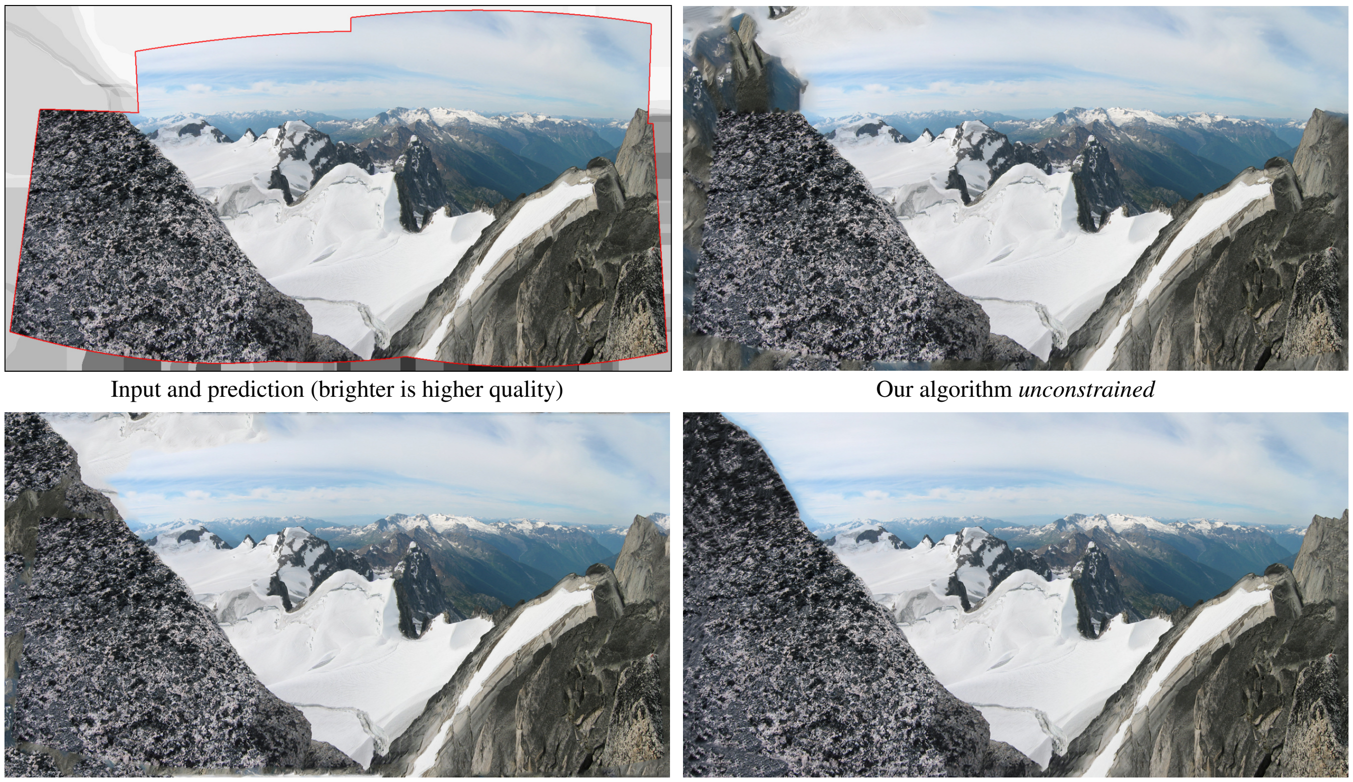
In the next section, we review the optimization framework that our algorithm is based on and then describe our extension that improves the result quality and makes the algorithm more predictable.

4 Image Completion Algorithm

Our algorithm minimizes a "texture energy" term which measures the extent to which the *synthesized region* deviates from the *known region* over a set of overlapping local patches. The basic form of the energy minimization problem is

$$\min_{\{\mathbf{t_i},\mathbf{s_i}\}} \sum_{i \in \Omega} \|\mathbf{t_i} - \mathbf{s_i}\|, \tag{1}$$

where Ω is the set of center pixels of all 7×7 patches that are completely contained within the image domain and overlap at least one missing pixel. By "image domain," we mean the minimum bounding box containing the panorama. $\mathbf{t_i}$ denotes a (target) patch centered at pixel i, and $\mathbf{s_i}$ is a (source) patch in the known region K that is close in appearance to $\mathbf{t_i}$.


Input and prediction (brighter is higher quality)

Our algorithm *unconstrained*

Adobe Photoshop CS5 (*Content Aware Fill*)

Our Algorithm *with* automatic constraints

Figure 2: *Unguided image completion often produces semantically implausible results. Please zoom into the PDF to see details. Note the artifacts in the top left corner and near the bottom. Both ours and Photoshop's implementation of [Wexler et al. 2007] synthesized the wrong rock texture and snow patches into the sky. In previous work these issues have been addressed by using* manual *search space constraints. We propose a fully automatic algorithm to derive such constraints. Our result, while not artifact free, appears more plausible. Many more examples can be found in the supplementary material.*

The energy is minimized in an iterative fashion, alternating between minimizing with respect to set of $\mathbf{t_i}$ or $\mathbf{s_i}$ while the other set is fixed. Minimizing the $\mathbf{s_i}$ assignments requires finding the nearest neighbor for each synthesized patch $\mathbf{t_i}$. This task is the most compute-intensive, and requires using approximative techniques to keep the run time reasonable (we use the PatchMatch algorithm [Barnes et al. 2009]). Minimizing $\mathbf{t_i}$ amounts to a simple averaging of the overlapping $\mathbf{s_i}$ assignments. Our results are computed in a coarse-to-fine fashion using an image pyramid. For details, please refer to [Wexler et al. 2007] and the pseudo-code in the supplementary material.

4.1 Automatic search space constraints

This completion algorithm is prone to getting stuck in bad local minima, and is sensitive to its initial state. The most successful applications start with initializations that are already close to the final result: image retargeting [Simakov et al. 2008] uses a scaled copy of the input, while image reshuffling [Barnes et al. 2009] starts from a copied known region. In both cases, the amount of repair is small. For plain image completion the algorithm is typically initialized randomly, which makes the quality of the results hard to predict. Without a random initialization the algorithm tends to exhibit quality issues such as blurry/incorrect textures and semantically implausible results, as shown in Figure 2.

Certain algorithms are *manually* guided toward better solutions by either constraining the locations of some pixels (e.g., [Barnes et al. 2009]) or specifying structure (e.g., [Sun et al. 2005]). In other works, the search space is constrained using simple heuristics. Pérez *et al.* [2004] restrict the source region to a band of constant radius around the missing region. However, this simple solution is problematic: if the radius is too small, not enough data can be sampled for effective completion, which may result in repetitions; if the radius is too large, the results will suffer from the same problems as observed in the *unconstrained* result in Figure 2. Bornard *et al.* [2002] describe a more localized heuristic. For every missing pixel they restrict the search space to the smallest centered window that contains at least a certain number of known pixels. While this largely avoids the problem of selecting incorrect textures the windows are still not well adapted to the image content.

We use a similar heuristic, which uses texture segmentation to select large restriction regions with relatively homogeneous content, in order to facilitate predictability and to achieve continuation of semantic regions. We considered using image segmentation techniques [Jia and Tang 2003], however, it is difficult to control the granularity of the resulting segments. In addition, these segments are non-overlapping, which might lead to artifacts in the form of artificial hard edges in the result.

Instead, we oversegment into superpixels and then associate each superpixel with a cluster of similar superpixels surrounding it. These clusters are homogeneous and overlapping as desired. We experimented with several superpixel algorithms including normalized cuts [Yu and Shi 2003] and graph-based based segmentation [Felzenszwalb and Huttenlocher 2004]. It turned out that the added complexity of these algorithms was not necessary for achieving good results; instead, we partition the entire image into non-overlapping square tiles, each being 16×16 pixels. There are three

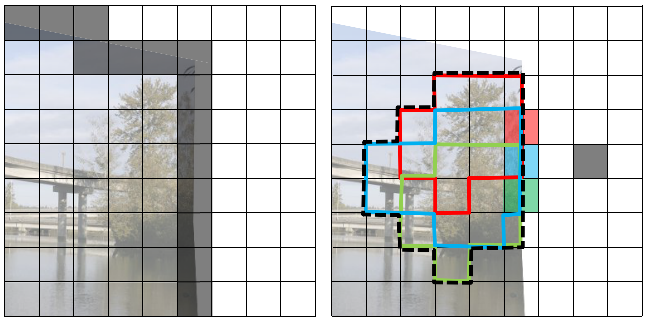


Figure 3: *Computation of search space constraints. The image is partitioned into square tiles, the white area is part of the missing region.* Left: *boundary tiles (dark), known and missing tiles (light).* Right: *for the dark missing tile the closest boundary tile is shown in blue. The red and green boundary tiles are within 1.5× distance of the closest boundary tile. Their respective segments are outlined in red, green, and blue. The union of all three segments yields the restricted search space (outlined with dashed black line).*

(non-disjunct) categories of tiles: *boundary tiles* overlap or touch the known image boundary, *known tiles* contain at least one known pixel, and *missing tiles* contain at least one missing pixel (Figure 3a). Note that each boundary tile is also either "known", "missing", or both.

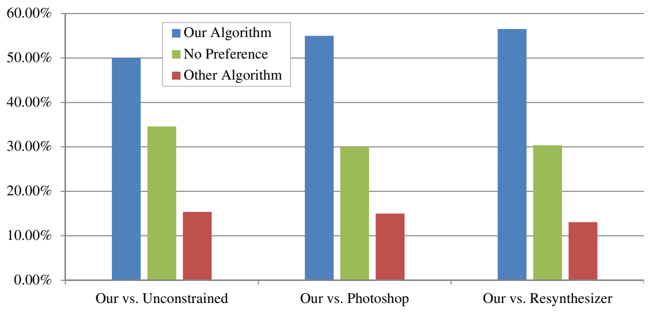
For every boundary tile, we compute a "segment" comprising of surrounding known tiles of sufficiently similar texture (decribed in the next paragraph). Next, we associate with each missing tile a restriction region that is formed as the union of several overlapping segments. This union contains the segments of all boundary tiles within 1.5 times the distance of the closest boundary tile. We associate every pixel within a missing tile with the restriction region of that tile. This algorithm has the desired effect of producing small restriction regions near to the image boundary, forcing the completion to continue semantic regions, whereas further away from the boundary it results in larger restriction regions giving the completion algorithm more freedom. It works both for holes on the outside of the image (panoramas) and in the interior of the image. The algorithm is illustrated in Figure 3, right.

For computing the homogeneous segments around boundary tiles, we consider the tiles as a four-connected grid graph. Each edge is weighted by the affinity of the attached tiles, computed as the Earth Mover's distance (EMD) of their color histograms (one 16-bin histogram per channel). We also tried χ^2 and Euclidean distance but achieved slightly better results using EMD. For each boundary tile, we compute the shortest distance path to every other tile using Dijkstra's algorithm. We define the segment as the union of all tiles whose distance is smaller than the threshold $\tau = 5$ (the distance between two bins in the EMD is set to 1). If the segment has fewer than 32 tiles, the threshold is automatically adjusted to select at least 32 tiles.

4.2 Evaluation

We conducted a user study with 13 participants (3 female and 10 male). We tested our algorithm against three competitors: our algorithm *without* automatic search space constraints, Adobe Photoshop CS5's Content Aware Fill, and Resynthesizer[2], a popular plugin for the GIMP image editor. Mechanical Turk was not used because of the high resolution display requirement. Each participant was presented with 60 pairs of images on two monitors (each image was shown on a full 24" diagonal monitor). 30 of these images were panoramas with holes on the outside, and the other 30 had holes in the interior of the image. The ordering was pseudo-randomly varied such that each participant compared our algorithm against 20 examples from each of the other techniques. Results are shown in Figure 4. A χ^2 analysis between each condition indicates that our algorithm was significantly preferred over each of the other techniques.

[2]http://www.logarithmic.net/pfh/resynthesizer



Our *vs.* unconstrained	Our *vs.* Photoshop	Our *vs.* Resynthesizer
$\chi^2(2, 260) = 46.92$ $p < 0.0001^*$	$\chi^2(2, 260) = 63.70$ $p < 0.0001^*$	$\chi^2(2, 260) = 74.68$ $p < 0.0001^*$

* shows statistically significant result.

Figure 4: *Results of the user study comparing our algorithm against several other state-of-the-art algorithms.*

5 Quality Prediction

The automatic search space constraints are designed in part to facilitate prediction of quality. In this section, we describe how we generate the prediction function that maps a given missing pixel to a measure of perceptual quality.

The data used to learn the prediction function was obtained using Mechanical Turk. From a set of 125 completed panoramas, we randomly selected 64×64 squares with at least half missing pixels and marked them with red bounding boxes in the image. We cropped 380×380 regions around the squares to provide visual context and asked subjects to rate the completion in the square as either "good" or "bad". We generated 1,500 batches with twelve questions each. Each batch contained ten real questions and two control questions with obvious answers. These control questions enabled us to prune subjects who were not performing the task properly, a common problem on Mechanical Turk. A subject is considered invalid if fewer than 80% of the control questions were incorrectly answered. After pruning, we have 66 valid subjects (down from 117 subjects), generating a total of 8,738 "good" and 802 "bad" labeled examples of completed regions. The last two numbers suggest that, on average, users will find about 92% of our filled-in regions "good", i.e., plausible-looking.

We use the labeled data to learn a function that predicts the perceived quality of a completed missing region. Recall that each missing pixel i is constrained to come from a certain restriction region $R_i = \bigcup S_j$, composed as the union of several homogeneous segments S_j. Our prediction function learns the correlation between the perceived quality of a completed pixel and some low-level features of the segments comprising the restriction region.

The feature vector $\mathbf{u}_j$ of a segment S_j has the following components: color histograms (separate for each channel), edge density (percentage of pixels that are edge pixels), edge orientation histogram, and histograms of contour and straight line lengths. Each histogram is characterized by its entropy, mean, and standard deviation. We provide additional implementation details in a supplementary document.

The feature vector $\mathbf{v_i}$ of a missing pixel i is defined as:

$$\mathbf{v_i} = \begin{pmatrix} \sum_j a_j \mathbf{u}_j / \sum_j a_j \\ x_i \\ y_i \\ d_i \\ b_i \end{pmatrix}, \qquad (2)$$

where x_i, y_i are the image coordinates of the missing pixel (normalized so that each ranges between 0 and 1), d_i is its distance to the boundary of the nearest known region, a_j is the size of segment j, and b_i is the area of R_i (typically smaller than $\sum_j a_j$, as can be deduced from Figure 3).

Let t_i be the label of pixel i ("good" = 1 or "bad" = -1). Our goal is to construct a function $f(\mathbf{v}_i)$ that predicts the unknown label from a feature vector $\mathbf{v_i}$. We use Gentle AdaBoost [Friedman et al. 2000], a standard machine learning algorithm for binary classification, to combine our feature vector into a scalar quality prediction. The prediction function has the form

$$f(\mathbf{v}_i) = \sum_{k=1}^{m} h_k(\mathbf{v}_i), \qquad (3)$$

which is a sum of m regression stumps

$$h_k(\mathbf{v}_i) = \begin{cases} l_k & \text{if } \mathbf{v}_i^{e_k} > t_k \\ r_k & \text{otherwise} \end{cases}, \qquad (4)$$

where $\mathbf{v}_i^{e_k}$ denotes the e_kth element of the feature vector $\mathbf{v}_i$. Gentle AdaBoost determines the model parameters e_k, t_k, l_k, and r_k by solving the problem

$$\min_f \sum_{i \in T} \exp(-t_i f(\mathbf{v}_i)), \qquad (5)$$

which penalizes positive values of f on "bad" examples and negative values on "good" that examples. Here, T denotes the set of labeled training examples. The number of regression stumps m is a design parameter of the learning algorithm. It is set empirically by cross-validation: we randomly split the training data into five validation folds, and then for each fold learn a prediction function using the data from the remaining four training folds. The average prediction performance on the five validation folds provides an estimate of the generalization performance of the model. We repeated this process for $m = 1, \ldots, 128$ and found that the performance increased with m until about $m = 32$, and then leveled off.

Figure 5 illustrates the effectiveness of the learning approach. The discriminative power of our learned predictor f is higher than any feature taken by itself (the individual elements of $\mathbf{v}_i$). As an aside, we also tested the pixel-wise energy Eq. 1 as a feature, but found it to be a poor quality predictor (AUC = 0.53, standard error: 0.006) compared to our dedicated quality features (Eq. 2, AUC = 0.77). Please note that this evaluation and all other results presented here were computed on images that were *not* part of the training set. We show examples of our quality prediction throughout the paper and in the supplementary material.

The quality predictions $f(\mathbf{v_i})$ are in arbitrary units. To make the quality parameter in our method more intuitive, we transform the quality predictions into probabilities using Platt's method [1999]. This method constructs a sigmoid mapping from the raw quality predictions $f(\mathbf{v_i})$ to estimates of "good" or "bad" label probabilities, $p_i \in [0; 1]$. Note that since f is a sum of regression stumps both f and p_i are real-valued and piecewise constant. p_i is very fast to compute, since it requires only $m = 32$ compare-adds per pixel.

6 Automatic Cropping Algorithm

We use our prediction function to compute an optimal crop shape C that includes as many of the known pixels as possible while avoiding pixels that are predicted as low quality. These objectives are achieved by solving the optimization problem

$$\min_C \left|\overline{C} \cap K\right| \quad \text{subject to} \quad \mu_p(C) = \frac{1}{|C|} \sum_{i \in C} p_i \geq \tau_p, \qquad (6)$$

where $\overline{C}$ denotes the region outside the crop shape, and $\mu_p(C)$ is the average predicted quality inside the crop shape ($p_i = 1$ for known pixels). The solution minimizes the number of excluded known pixels while ensuring a minimum average probability τ_p inside the crop shape.

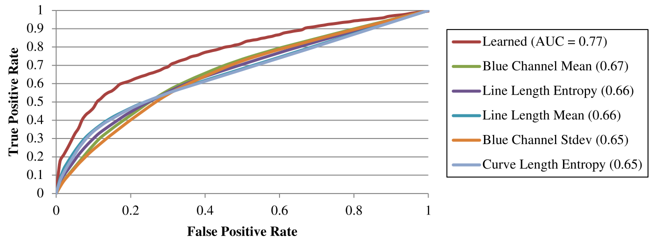


Figure 5: *ROC (receiver operating characteristic) curves for our learned prediction function vs. the five most predictive individual features (out of 21). The area under the ROC curve (AUC) measures discriminability of each predictor. AUC = 0.5 denotes chance level, AUC = 1.0 indicates perfect discrimination.*

The parameter τ_p balances between the two high-level objectives: higher values lead to more aggressive crops, since less potentially low-quality areas are allowed in the crop. In practice, we found $\tau_p \in [0.99, 1]$ to be a reasonable range of values. The values for τ_p are in the upper range because the subjects of the user study from Section 5 rated about $9/10$ of all samples "good", resulting in relatively high p_i's everywhere. Figure 6 illustrates the effect of varying τ_p. For all other results in this paper and in the supplementary materials we set $\tau_p = 0.9925$, the setting that scored best in the evaluation study described below and generally seems to work well across a wide range of panoramas (please refer to the results in the supplementary material). Figure 7 illustrates how our cropping optimization behaves on panoramas that are predicted mostly high or mostly low quality.

While the most commonly used shape is a rectangle, the formulation above supports arbitrary parametric crop shapes. We have experimented with various shapes including trapezoids, convex n-gons, ellipses, T-shapes, and L-shapes. Figure 9 shows various results using general shapes.

To solve the constrained optimization problem in Eq. 6, we first replace it with a sequence of unconstrained subproblems using the logarithmic barrier method [Nocedal and Wright 2000]. Instead of solving Eq. 6, we solve the unconstrained combined objective/barrier problem

$$\min_C \left|\overline{C} \cap K\right| - \lambda \log \mu_p(C). \qquad (7)$$

The minimizer of Eq. 7 approaches the solution of Eq. 6 as $\lambda \to 0$. We use the Simplex algorithm [Nelder and Mead 1965] as implemented in the GNU scientific library [Galassi et al. 2009] to solve this problem. Since the objective function may contain local minima, we approximate the global minimizer by starting the optimization from 100 random initial states and use the best result.

6.1 Evaluation for Rectangles

We ran a user study involving 13 subjects (10 males and 3 females) to evaluate our automatic cropping technique. Each subject was shown 20 full panoramas randomly selected from a subset of 50 panoramas on a high-resolution 24" monitor. For each panorama,

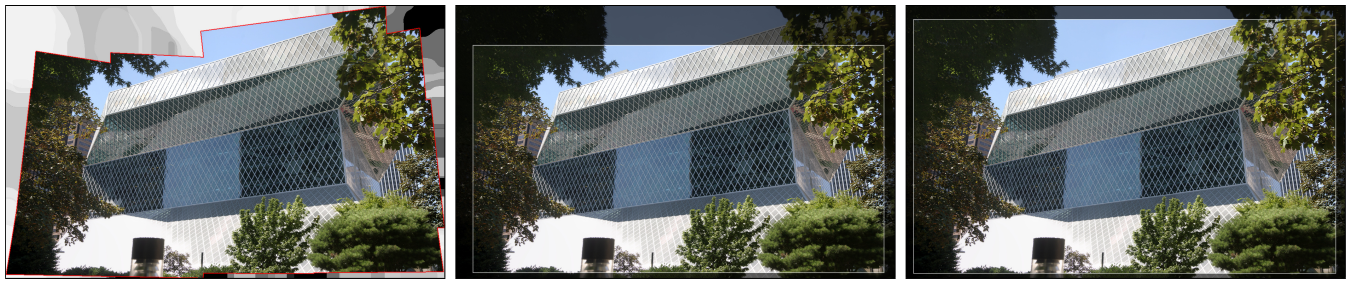

Input and prediction (brighter is better) More aggressive crop, $\mu_p = 0.9975$ Less aggressive crop, $\mu_p = 0.9925$

Figure 6: *The μ_p parameter controls the balance between our high level goals when cropping: a high setting (middle) avoids low-quality areas but might crop more; a lower setting (right, our default setting) includes more of the known pixels at the expense of a potentially lower completion quality.*

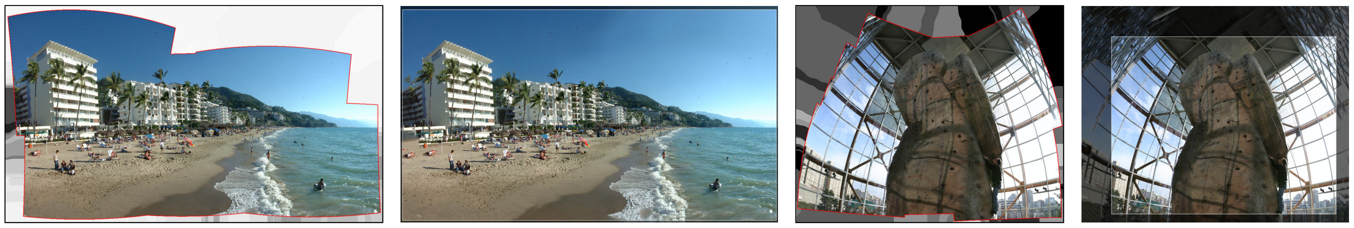

Input and prediction (brighter is higher quality) Cropped followed by image completion Input and prediction (brighter is higher quality) Cropped followed by image completion

Figure 7: *An "easy" and a "hard" input image. For the left panorama our prediction function indicates most of the missing regions* can *be completed well; our automatic crop is large. For the right input the prediction function indicates most of the missing regions* can not *be completed well. Our automatic crop is more conservative and avoids most of these regions.*

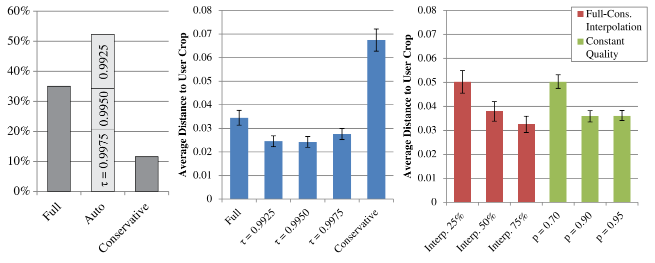


Figure 8: *Left: Results of the user study comparing non-cropped images (full), versus intelligently cropped (smart), versus no-cropping (conservative) or over cropping (extracons). Users significantly preferred smart cropping: $\chi^2(3, 260) = 165.94, p < 0.0001\%$. Middle: Average Hausdorff distances to crops chosen by the subjects of our user study. Our automatic crops are on average closer to the user preferences than either full or conservative cropping. Right: Average Hausdorff distances for linearly interpolated crop boxes between "full" and "conservative," and for cases where the predicted quality of each missing pixel is set to a constant.*

we generated 5 versions: the full uncropped completion, three autocrops with $\mu_p = 0.9925, 0.9950, 0.9975$, and a conservative crop (i.e., maximum known region crop). We also added an overcropped version that eliminated known pixels to help determine when subjects were cropping based on framing judgements as opposed to noticeable artifacts. For each set of panoramas, the original stitched photographs were first shown to the subject and subjects could use the left/right arrow keys to cycle through the differently cropped panoramas. Subjects were asked to choose a panorama that they would want to share with friends. Mechanical Turk was not used because of the high-resolution requirement. Subjects finished on average in 7.5 mins (standard deviation of 2.67 mins).

The results of the study showed that participants preferred some form of guided cropping 52% of the time. They used the uncropped version 35% and the conservative cropping only 11.5% of the time (see Figure 8, left). This analysis shows only part of the picture; depending on the prediction distribution, different crop choices may appear to be very similar. Such examples are shown in Figure 7, where our smart cropping is almost the same as the full image and conservative crop, respectively. In such cases, arbitrary decisions may be made, skewing the results.

To account for this issue, we performed another analysis that takes into account the relative shape distances between different crops. For each crop version selected by a user, we compute the symmetric Hausdorff distance to all available crop choices (normalized by the longer dimension of the image). The Hausdorff distance is a commonly used shape similarity metric, e.g., for template matching in computer vision applications. The results of this analysis (blue bars in Figure 8, middle) show that our guided crops have the smallest average Hausdorff distances to the crops selected by the subjects. This shows a clear preference for our guided crops over the full image or conservatively cropped images. We also compare various naïve crops to the user choices (Figure 8, right). The results in red show the average distances of crops achieved by interpolating the crop shapes between the full and conservative versions. The results in green were achieved using the same optimization as our results (using $\tau_p = 0.9925$), but using the same constant quality value for every missing pixel. For both cases, the distances are significantly higher (and hence less desirable) than for our best setting.

7 Additional Results

We tested our algorithm on several hundred panoramas and images with holes in the interior. In the supplementary material, we show an extensive comparison to various state-of-the-art algorithms

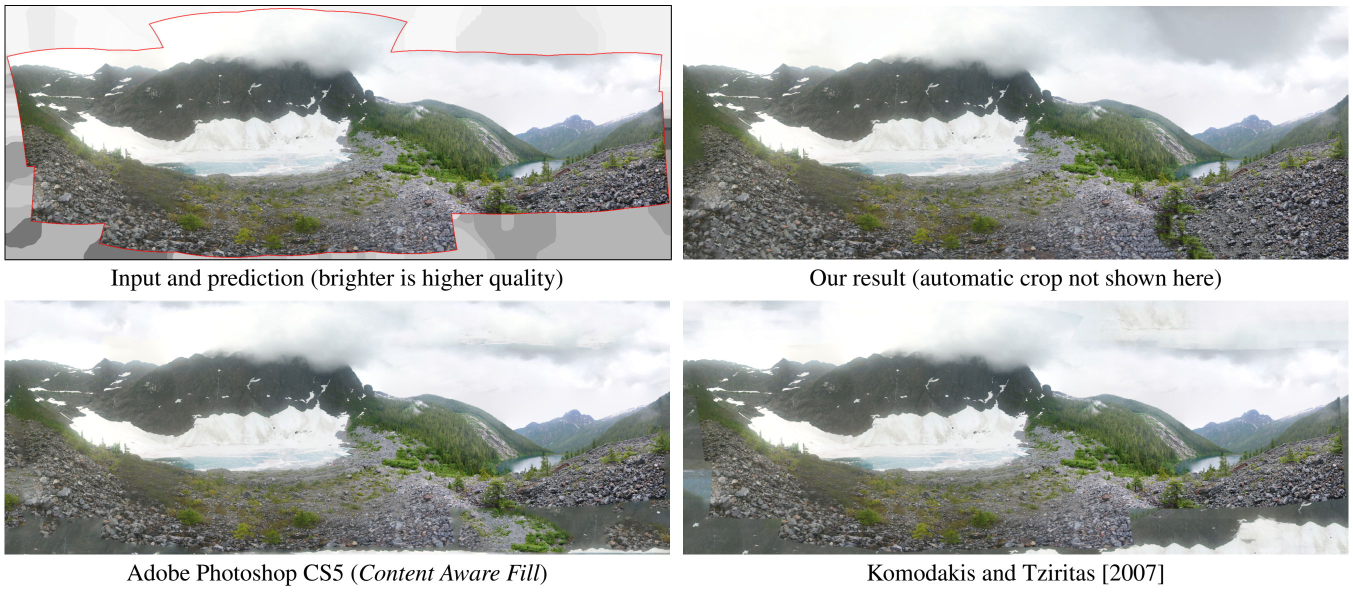

Input and prediction (brighter is higher quality)
Our result (automatic crop not shown here)
Adobe Photoshop CS5 (*Content Aware Fill*)
Komodakis and Tziritas [2007]

Figure 10: *Comparison with other image completion algorithms. Please zoom into the PDF to see details. Note the blurring problem in the Photoshop and Komodakis and Tziritas' results at the bottom corners. The results of both previous methods also contain incorrectly synthesized texture at the bottom and semantic errors, i.e. snow patches in the sky. Many more examples can be found in the supplementary material.*

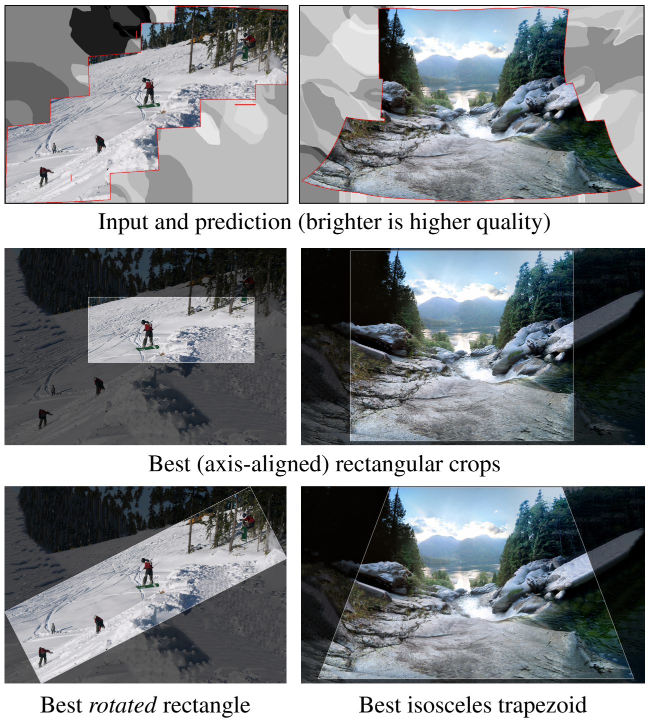

Input and prediction (brighter is higher quality)
Best (axis-aligned) rectangular crops
Best *rotated* rectangle
Best isosceles trapezoid

Figure 9: *Results of using alternative crop shapes. In both cases a significantly higher portion of the known pixels is included in the crop.*

(Adobe Photoshop CS5 Content Aware Fill, GIMP Resynthesizer, [Pritch et al. 2009], [Komodakis and Tziritas 2007], [Criminisi et al. 2003]) on 25 representative samples from each class. An example is shown in Figure 10. In addition, we show automatic cropping results for the 25 panoramas; representative results are shown throughout the paper.

On a dual Intel Xeon E5640 PC, we observe the following median timings for the 25 panoramas included in the supplementary material. We believe these numbers can be reduced with code optimization.

	Full completion	With auto-cropping
Restriction regions	0.32s	0.22s
Feature extraction	not applicable	0.93s
Crop optimization	not applicable	1.78s
Completion	13.29s	6.52s
Total	**13.70s**	**9.17s**

For these panoramas, our automatic crops contain on average slightly less than 50% of the missing pixels. Since the completion algorithm runtime is roughly linear in the number of missing pixels, this leads to a significant speed-up compared to first completing the full panoramas before cropping.

8 Limitations and Future Work

Image completion remains a very challenging problem. Like other recent approaches, our algorithm lacks higher-level (object-level) understanding of the input image. Thus, it will on occasion generate semantically implausible results, although our source location restriction significantly reduces these problems. Our cropping optimization currently ignores scene context and may crop out important objects in the scene. As seen in Figure 12, it fails to realize the importance of the two subjects. A possible solution is to use face and/or saliency detectors.

Our prediction function fit is not perfect, most likely due to occasional mismatches in subject ratings in the training database. As a result, our prediction function would, on occasion, mislabel the missing regions (e.g., Figure 11, where mislabeling resulted in a smaller crop). A future course of action would be to either analyze the function on a per-person basis (i.e., personalize the automatic cropping function), or partition the data into clusters of similar preferences, with each having a different cropping function.

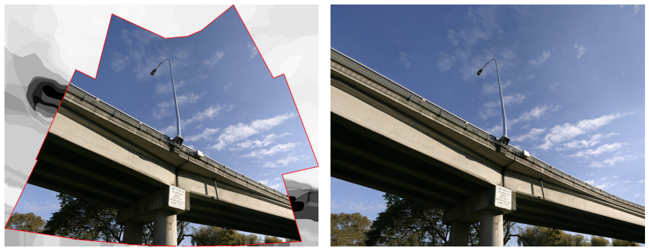

Input and prediction (brighter is higher quality) Entire completion

Figure 11: *Our prediction function sometimes tags "good" pixels as "bad." In this example, practically the entire missing region can be completed well. As a result of the overly-conservative prediction, the suggested crop is smaller than necessary.*

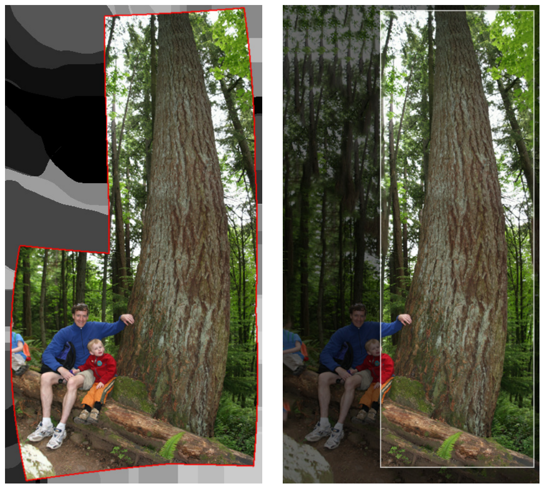

Input and prediction (brighter is higher quality) Cropped completion

Figure 12: *Example where our cropping fails to take into consideration object-level importance.*

References

BARNES, C., SHECHTMAN, E., FINKELSTEIN, A., AND GOLDMAN, D. B. 2009. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. on Graphics (Proceedings of Siggraph) 28*, 24:1–24:11.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. *ACM Trans. on Graphics (Proceedings of Siggraph) 19*.

BORNARD, R., LECAN, E., LABORELLI, L., AND CHENOT, J.-H. 2002. Missing data correction in still images and image sequences. *Proceedings of the tenth ACM international conference on Multimedia*, 355–361.

CRIMINISI, A., PEREZ, P., AND TOYAMA, K. 2003. Object removal by exemplar-based inpainting. In *CVPR*, 417–424.

DARABI, S., SHECHTMAN, E., BARNES, C., GOLDMAN, D. B., AND SEN, P. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. on Graphics (Proceedings of Siggraph) 31*, 4.

DRORI, I., COHEN-OR, D., AND YESHURUN, H. 2003. Fragment-based image completion. *ACM Trans. on Graphics (Proceedings of Siggraph) 22*, 303–312.

EFROS, A., AND LEUNG, T. 1999. Texture synthesis by non-parametric sampling. In *CVPR*, 1033–1038.

FELZENSZWALB, P., AND HUTTENLOCHER, D. 2004. Efficient graph-based image segmentation. *IJCV 59*, 2, 167–181.

FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. 2000. Additive logistic regression: a statistical view of boosting. *Annals of Statistics 28*, 2, 337–407.

GALASSI, M., DAVIES, J., THEILER, J., GOUGH, B., AND JUNGMAN, G. 2009. *GNU Scientific Library – Reference Manual, Third Edition*. Network Theory Ltd.

JIA, J., AND TANG, C.-K. 2003. Image repairing: robust image synthesis by adaptive nd tensor voting. *Proc. CVPR 2003*, 643–650.

KAWAI, N., SATO, T., AND YOKOYA, N. 2008. Image inpainting considering brightness change and spatial locality of textures and its evaluation. In *PSIVT '09*, 271–282.

KOMODAKIS, N., AND TZIRITAS, G. 2007. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Trans. Image Processing 16*, 2649–2661.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Trans. on Graphics (Proceedings of Siggraph) 24*, 795–802.

MANSFIELD, A., PRASAD, M., ROTHER, C., SHARP, T., KOHLI, P., AND VAN GOOL, L. 2011. Transforming image completion. In *British Machine Vision Conf. (BMVC)*.

MATSUSHITA, Y., OFEK, E., GE, W., TANG, X., AND SHUM, H.-Y. 2006. Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Anal. Mach. Intell. 28*, 7 (July), 1150–1163.

NELDER, J., AND MEAD, R. 1965. A simplex method for function minimization. *Computer Journal 7*, 308–313.

NOCEDAL, J., AND WRIGHT, S. J. 2000. *Numerical Optimization*. Springer.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2004. Patchworks: example-based region tiling for image editing. Tech. Rep. MSR-TR-2004-04, Microsoft Research.

PLATT, J. C. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, MIT Press, 61–74.

PRITCH, Y., KAV-VENAKI, E., AND PELEG, S. 2009. Shift-map image editing. In *ICCV'09*, 151–158.

SIMAKOV, D., CASPI, Y., SHECHTMAN, E., AND IRANI, M. 2008. Summarizing visual data using bidirectional similarity. In *CVPR*.

SUN, J., YUAN, L., JIA, J., AND SHUM, H.-Y. 2005. Image completion with structure propagation. *ACM Trans. on Graphics (Proceedings of Siggraph) 24*, 861–868.

SWAMY, D., CHANDLER, D., BUTLER, K., AND HEMAMI, S. 2011. Parametric quality assessment of synthesized textures. *Proc. Human Vision and Electronic Imaging 2011*.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time video completion. *TPAMI*.

YU, S., AND SHI, J. 2003. Multiclass spectral clustering. 313–319 vol.1.

ZHANG, M., ZHANG, L., SUN, Y., FENG, L., AND MA, W. 2005. Auto cropping for digital photographs. In *ICME*.