IEEE Computer Vision and Pattern Recognition (CVPR) 2010

A probabilistic image jigsaw puzzle solver

Taeg Sang Cho[†], Shai Avidan[‡], William T. Freeman[†]
[†] Massachusetts Institute of Technology
[‡] Tel-Aviv University
taegsang@mit.edu, shai.avidan@gmail.com, billf@mit.edu

Abstract

We explore the problem of reconstructing an image from a bag of square, non-overlapping image patches, the jigsaw puzzle problem. Completing jigsaw puzzles is challenging and requires expertise even for humans, and is known to be NP-complete. We depart from previous methods that treat the problem as a constraint satisfaction problem and develop a graphical model to solve it. Each patch location is a node and each patch is a label at nodes in the graph.

A graphical model requires a pairwise compatibility term, which measures an affinity between two neighboring patches, and a local evidence term, which we lack. This paper discusses ways to obtain these terms for the jigsaw puzzle problem. We evaluate several patch compatibility metrics, including the natural image statistics measure, and experimentally show that the dissimilarity-based compatibility – measuring the sum-of-squared color difference along the abutting boundary – gives the best results. We compare two forms of local evidence for the graphical model: a sparse-and-accurate evidence and a dense-and-noisy evidence. We show that the sparse-and-accurate evidence, fixing as few as $4-6$ patches at their correct locations, is enough to reconstruct images consisting of over 400 patches. To the best of our knowledge, this is the largest puzzle solved in the literature. We also show that one can coarsely estimate the low resolution image from a bag of patches, suggesting that a bag of image patches encodes some geometric information about the original image.

1. Introduction

We explore the problem of reconstructing an image from a bag of square image patches, the jigsaw puzzle problem. Given square, non-overlapping patches sampled from an image grid, our goal is to reconstruct the original image from them.

A jigsaw puzzle is an intellectually intriguing problem, which is also provably technically challenging. Demaine *et al.* [5] show that the jigsaw puzzle problem is NP-complete when the pairwise affinity of jigsaw pieces is unreliable. Despite the challenge, many scientific problems, including speech descrambling [23], DNA / RNA modeling [14], reassembling archeological relics [2] and document fragments [24], can be modeled as jigsaw puzzles. The NP-complete complexity of jigsaw puzzles has also been exploited in cryptography [3, 7].

In this paper, we focus on solving image jigsaw puzzles with square pieces. This type of puzzles, sometimes called jig swap puzzles, is missing the shape information of individual pieces, which is critical for evaluating pairwise affinities among them. Therefore this problem formulation is even more challenging than solving conventional jigsaw puzzles. This, however, is a good framework for analyzing structural regularities in natural images since it requires us to focus on the image content to solve the puzzle.

This paper also lays groundwork for addressing the patch-based image editing / image synthesis problems in which the image layout is required, but is not readily apparent. For example, in the patch transform image editing scenario [4], one needs to know the image layout in order to synthesize a visually pleasing image. However, in some cases, – for instance, when we mix patches from multiple images to synthesize a single image –, it's unclear what the image layout should be. This paper studies how well we can recover the image layout and a natural looking image from a bag of image patches. Such statistical characterization of images is useful for image processing and image synthesis tasks.

We use a graphical model to solve the jigsaw puzzle problem: Each patch location is a node in the graph and each patch is a label at each node. Hence, the problem is reduced to finding a patch configuration that is most likely on the graph. Cho *et al.* [4] solved this problem in their patch transform work, but assumed access to a low-resolution version of the original image, information not available for the jigsaw puzzle problem. Nevertheless, we are assured that we can solve the jigsaw puzzle problem if we can address the simpler problem of the lack of a low resolution image.

We evaluate two methods to address this issue. The

first approach estimates a low resolution image from a bag of patches. The estimated low resolution image serves as dense-and-noisy local evidence for the graphical model. The second approach is to fix a small number of patches, called anchor patches, at their correct locations. Anchor patches serve as sparse-and-accurate local evidence. We can view the anchor patches as injected geometric information. We study how much geometric information is needed to reliably reconstruct an image from its bag of patches.

We demonstrate successful image reconstructions of 20 test images. The results suggest that the spatial layout of a bag of patches is quite constrained by the patches in the bag, and that a simple bag of patches does not throw away as much geometric information as might be thought.

Contribution We summarize our contributions as below:

- We explore a number of patch compatibility metrics for the graphical model. We show that the dissimilarity-based compatibility – measuring the sum-of-squared color difference along the abutting boundary – is the most discriminative.
- We evaluate two strategies to model the evidence term in the graphical model: dense-and-noisy evidence and sparse-and-accurate evidence. The first approach estimates the low resolution image from a bag of patches. The second approach assumes that few patches, called anchor patches, are fixed at their correct location in the puzzle.
- We introduce three measures to evaluate the puzzle reconstruction accuracy, and show that our algorithm can reconstruct real images reliably.

2. Background

Freeman and Gardner [8] were the first to propose an algorithm for solving jigsaw puzzles. Many papers [10, 11, 16, 21] assume using classic jigsaw pieces with distinct shapes, and focus on matching the shape of the pieces to solve the puzzle. Kosiba *et al.* [12] considered both the boundary shape *and* the image contents, and many papers [13, 15, 22] followed suit. Most algorithms solve the puzzle in two steps. The frame pieces are assembled first and the interior is filled in with a greedy algorithm. To date, the maximum number of jigsaw pieces completed by these algorithms is 320 (16x20) [15], and most of them report the reconstruction result on just one or few images. We present a global optimization framework for solving the jigsaw puzzle problem, and show the effectiveness on multiple images.

We adopt the image model in Cho *et al.* [4] to solve the image jigsaw puzzle. The patch transform synthesizes an image from a set of image patches. Let $\mathbf{y}$ be a low resolution version of the original image, $p(y_i|x_i)$ be the local evidence term that steers the reconstructed image $\mathbf{x}$ to have a similar scene structure as $\mathbf{y}$, and i be the index of the patch locations. To reconstruct an image, the patch transform maximizes the following probability:

$$P(\mathbf{x};\mathbf{y}) = \frac{1}{Z}\prod_{i=1}^{N}\prod_{j\in\mathcal{N}(i)} p(y_i|x_i)p_{i,j}(x_j|x_i)p(x_i)E(\mathbf{x}) \tag{1}$$

where $p_{i,j}(x_j|x_i)$ is the probability of placing a patch x_j in the neighborhood of another patch x_i, $\mathcal{N}(i)$ is the Markov blanket of a node i, and $E(\mathbf{x})$ is an exclusion term that discourages patches from being used more than once. In contrast to Cho *et al.* [4], we do not assume we know what the low resolution image $\mathbf{y}$ is.

We can interpret Eq. (1) as a graphical model, and find the patch configuration $\mathbf{x}$ that maximizes the probability Eq. (1) using loopy belief propagation. The message from a node j to a node i is:

$$m_{ji}(x_i) \propto \sum_{x_j} p_{i,j}(x_i|x_j)p(y_j|x_j)\prod_{l\in\mathcal{N}(j)\setminus i} m_{lj}(x_j) \tag{2}$$

We can find the marginal probability at a node i by gathering all messages from its neighbors and the local evidence:

$$b_i(x_i) = p(y_i|x_i)\prod_{j\in\mathcal{N}(i)} m_{ji}(x_i) \tag{3}$$

$E(\mathbf{x})$ is a factor node that gathers messages from all nodes. $E(\mathbf{x})$ suppresses the use of the patch l if any of the other nodes already claimed the patch l with a high probability. In terms of message passing, the factor f sends a message m_{fi} to a node i:

$$m_{fi}(x_i = l) \approx \prod_{t\in S\setminus i}(1 - m_{tf}(x_t = l)) \tag{4}$$

where m_{tf} is the marginal probability at node t, and S is the set of all image nodes. We use this model, which Cho *et al.* [4] used for image editing, to solve jigsaw puzzles.

3. Compatibility

The pair-wise patch compatibility $P_{i,j}(x_j|x_i)$ tells us how likely it is for a patch x_j to appear next to another patch x_i. There are four types of compatibilities for each pair of patches: the compatibility of placing the patch x_j to the left/right/top/bottom of the patch x_i. If the pairwise compatibility between patches is accurate, we can solve the jigsaw puzzle in a polynomial time using a greedy algorithm [5]. Given this importance, we carefully evaluate different compatibility metrics.

We compare five types of compatibility measures: a dissimilarity-based compatibility, a boosting-based compatibility, a set-based compatibility, an image statistics-based compatibility, and the combination of a dissimilarity-based and image statistics-based compatibility as in Cho *et al.* [4].

3.1. Compatibility metrics

Dissimilarity-based compatibility We compute the dissimilarity between patches x_j, x_i by summing the squared color difference along the abutting boundaries. For example, the left-right (LR) dissimilarity between x_j, x_i is

$$D_{LR}(x_j, x_i) = \sum_{k=1}^{K} \sum_{l=1}^{3} (x_j(k, u, l) - x_i(k, v, l))^2 \quad (5)$$

where patches x_j, x_i are regarded as $K \times K \times 3$ matrices, u indexes the last column of x_j, and v indexes the first column of x_i. We compute the color difference in the normalized LAB color space, where chrominance components are normalized to have the same variance as the luminance component. We convert this squared difference to a probability by exponentiating the color difference D:

$$P_{i,j}(x_j|x_i) \propto \exp\left(-\frac{D(x_j, x_i)}{2\sigma_c^2}\right) \quad (6)$$

where σ_c is adaptively set as the difference between the smallest and the second smallest $D(x_j, x_i)$ among all x_j. Note that the dissimilarity is not a distance: $D(x_j, x_i) \neq D(x_i, x_j)$.

Boosting-based compatibility We train a boosting classifier to identify matching edges by deriving a feature vector from boundary pixels. Given patches x_i and x_j, we take a 2-pixel band from each patch at the abutting boundary, and sum the squared difference of all pairwise 2-pixel bands in x_i and x_j. This captures the correlation between pixels at the abutting boundary. When there are K pixels per column, the feature vector is $3 \times 4K^2$ dimensional (i.e. 3 for the color channels). We train the classifiers using a Gentle boost algorithm [9, 19], with 35200 true samples, and 35200 false samples. We use the classifier margin as the compatibility.

Set-based compatibility The set-based compatibility is inspired by the bidirectional similarity [18]. The set dissimilarity is the minimum distance between the $K \times K$ patch at the abutting boundary of two patches x_i, x_j and all other patches in the database. We use the sum of squared color difference as the distance. We exponentiate the distance as in Eq. (6) to convert it to a compatibility. Under this measure, a patch pair is compatible if their boundary region is similar to one of the patches in the database. In our implementation, we sample the boundary region half from the left patch and the other half from the right patch, but other ratios are possible as well.

Image statistics-based compatibility Weiss and Freeman [20] present a set of filters that lies in the null space of natural images. We convolve the $K \times K$ patch at the abutting boundary of two patches with these filters. Patch pairs with a small filter response at the boundary are given a high compatibility score as in [4, 20].

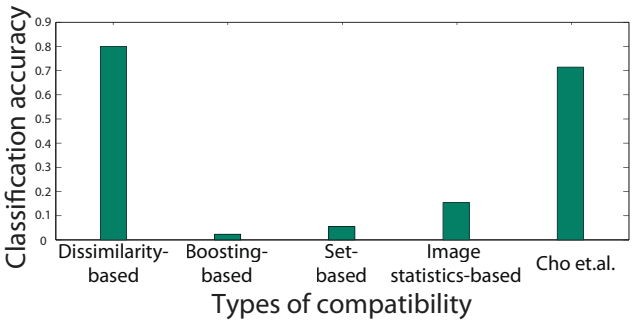


Figure 1. We evaluate five compatibility metrics based on the classification criterion. For each image in the test set consisting of 20 images, we find the portion of correct patch pairs that received the highest compatibility score among other candidates. We show the average classification accuracy of 20 images. We observe that a dissimilarity-based compatibility metric is the most discriminative.

The compatiblity in Cho *et al.* [4] Cho *et al.* [4] combines the dissimilarity-based compatibility and the image statistics-based compatibility by multiplying the two.

3.2. Evaluation

Patch pairs that were adjacent in the original image should receive the highest compatibility score among others. We use this characteristic as a criterion for comparing compatibility metrics. For each patch x_i, we find the match x_j with the highest compatibility, and compute for what fraction of test patches x_i the compatibility metric assigns the highest compatibility to the correct neighbor. We performed this test on 20 images. Figure 1 shows the average classification accuracy for each compatibility metric.

Interestingly, the naive dissimilarity-based compatibility measure outperforms other sophisticated compatibility measures under the classification criterion. We can attribute this observation to the fact that the patch neighbor classification problem is that of finding the *best* match among the set of patches from the *same* image, not that of finding a patch boundary that looks as similar as possible to training images. Learning-based compatibility metrics measure how natural the boundary regions are and do not necessarily preserve the likeliness ranking. The compatibility metric in Cho *et al.* [4] is useful for finding visually pleasing patch matches *other* than the correct match and is useful for image editing purposes. However, for the purpose of solving jigsaw puzzles, the dissimilarity metric is the most reliable, giving the highest classification accuracy.

We also observe that the compatibility performance depends on the image content. Images with high classification accuracy tend to have more texture variations, whereas images with low classification accuracy lack details. To solve

the jigsaw puzzle, we use the dissimilarity-based compatibility.

4. Local evidence

The local evidence determines the image layout. Without it, the belief propagation algorithm in Section 2 generates images that do not conform to standard image layouts. In Cho *et al*. [4], the local evidence term at pixel i favors patches with a similar mean RGB color as the i^{th} pixel in the low resolution image:

$$p(y_i|x_i = l) \propto exp\left(-\frac{(y_i - m(l))^2}{2\sigma_e^2}\right) \quad (7)$$

where $m(l)$ is the mean color of patch l, i indexes pixels, and $\sigma_e = 0.4$. In the jigsaw puzzle problem, however, we do not have the low resolution image $\mathbf{y}$.

We explore two strategies to emulate a low resolution image: dense-and-noisy local evidence and sparse-and-accurate local evidence.

4.1. A dense-and-noisy local evidence

We estimate dense-and-noisy local evidence from a bag of image patches. We represent a bag of image patches as a patch histogram, and learn the correspondence between a patch histogram and a low resolution image.

The patch histogram We create a patch vocabulary by sampling patches from training images, and clustering them. To have enough patches that are representative of various textures, we sample $8,500,000$ patches of size 7×7 from $15,000$ images taken from the LabelMe database [17].

We explore two types of patch representations for clustering: color-based and gradient-based. The color-based representation rasterizes a patch into a 147 (7x7x3) dimensional feature vector. The gradient-based feature sums the x,y gradient of a gray-scale patch along every row and column. We augment the 28-dimensional (7x2x2) gradient-based feature with the mean RGB values, generating a 31 dimensional vector. The motivation behind this representation is that similar patches tend to have similar gradient profiles. We reduce the dimensionality of these representations to retain 98% of the original signal variance through Principal Component Analysis (PCA).

Clustering millions of high dimensional features is not a trivial task. We cluster the patches in two steps. First, we cluster patches sampled from the same image into L clusters. We compute the cluster center for each cluster by averaging patches that belong to the same cluster. Then we re-cluster L cluster centers from all images to find the N global clusters. We used the fast K-means algorithm [6] for clustering. In this paper, $L = 20, N = 200$.

Given the N cluster centers, we can associate each image with a patch histogram h. The i^{th} entry of a patch histogram h counts the number of patches that belong to the i^{th} cluster. The patch histogram is fairly sparse since each image consists of 432 patches.

The patches within boxes in Figure 2 are the 20 most occurring cluster centers when we represent patches using (a) a gradient-based feature or (b) a color-based feature. The gradient-based feature uses the gray level and the edge information, whereas the color-based feature uses the gray level and the color information.

Properties of the patch clusters We can predict where in the image each patch cluster is most likely to occur. To do so, we back-project the patch cluster centers to training images, and observe where in the image they occur most frequently. We count the number of times a patch from a certain cluster appears at each patch location. This is called the patch cluster probability map. The patch cluster probability maps are shown in Figure 2, pointed by the arrows from the corresponding cluster centers.

Probability maps of the gradient-based patch representation show that clusters corresponding to edges tend to be in the foreground, but do not have strong spatial constraints. The clusters encoding intensity information carry more spatial information: bright patches usually appear at the top since objects near the sky (or the sky itself) are brighter than other objects in the scene.

The clusters from the color-based patch representation capture both intensity and color information. The patch probability maps show that some colors correspond to natural lighting, background scenes, or vignetting effects, and some other colors correspond to foreground objects. For example, a blue patch predominantly occurs in the upper half of the image, whereas brown and dark red colors most frequently correspond to foreground objects. The patch maps show a rich set of location constraints for different patch classes. (We anticipate that other feature representations, such as SIFT, would show similarly rich spatial localization structure.) This structure allows us to very roughly place each feature in the image, or to estimate a low-resolution image from the bag of features.

A probability map can be used as a patch prior. If a cluster s appears often at node i, patches that belong to the cluster s are given higher probability to appear at node i.

Image estimation through regression We learn a linear regression function A that maps the patch histogram h to the low resolution image $\mathbf{y}$, training A on images that were also used to find cluster centers. We use the color-based patch representation since it captures more spatial information.

Let columns of H be the patch histograms of training images, and columns of Y be the corresponding low resolution images. We learn the regression function A as follows [1]:

$$A = YH^T(HH^T)^{-1} \quad (8)$$

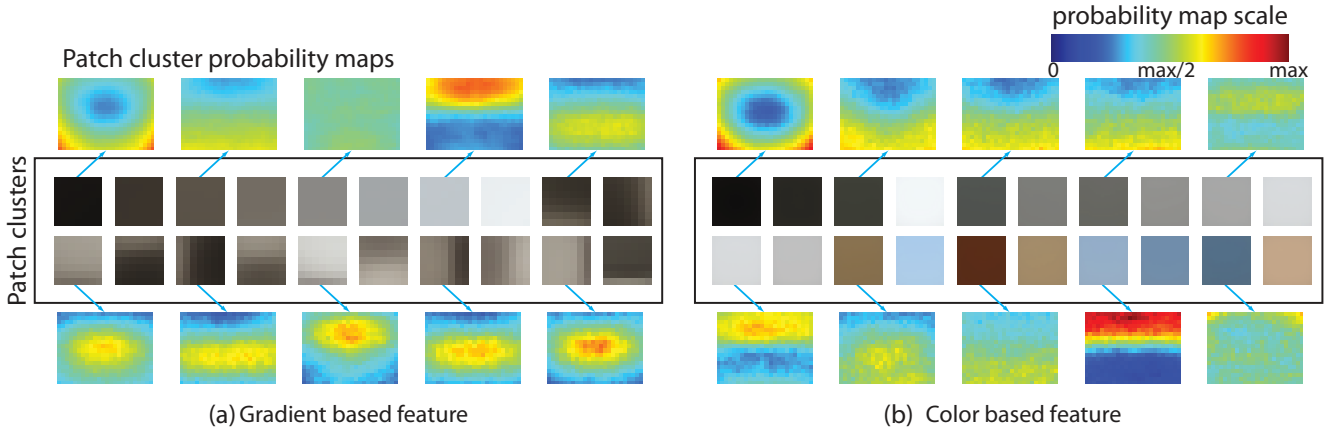


Figure 2. The patches in rectangular boxes are the top 20 most occurring patch cluster centers when we use (a) a gradient-based / (b) a color-based representation. Around the boxes are patch probability maps for a subset of cluster centers, pointed by the arrows from the corresponding patch cluster.

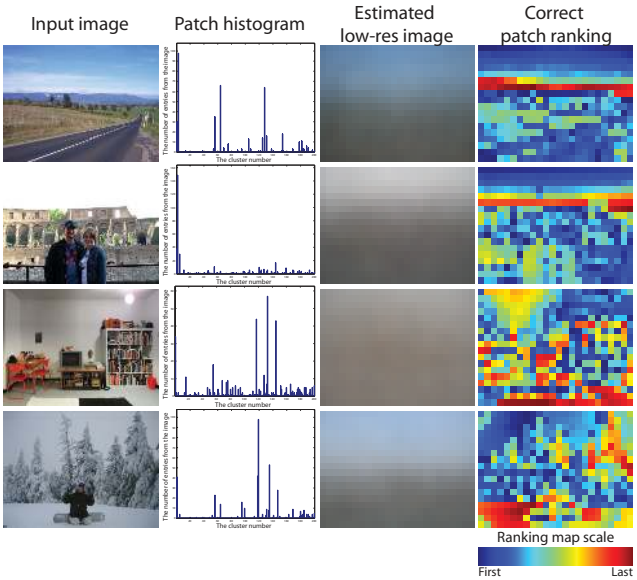


Figure 3. The patch histogram can be used to estimate a low resolution image. The regression function generates a low resolution image that resembles the original image, but we can nonetheless find examples that fail (the last row). The last column is a patch rank map: At each node, we order patches based on the likelihood given the estimated low resolution image, and show the rank of the true patch. Most of the correct patch rankings score high using the regression function – the ideal result is deep blue everywhere.

The size of the original image is roughly 700×500, and the estimated low resolution image is 24×18.

Figure 3 shows some experimental results. We observe that the regression can coarsely predict the low resolution image. This observation counters the intuition that the bag of features does not encode any spatial information. One possible explanation is that there are enough structural regularities in images so that a bag of features implicitly captures the geometric information. For example, when there are many patches that belong to a blue cluster, it's likely that they constitute a blue sky. Of course, it is easy to find failure examples: the blue tone of snow is misclassified as sky in the last example in Figure 3. Nevertheless, the regression function learns important image regularities: something bright should be at the top, illuminating foreground objects at the bottom.

We quantitatively evaluate the accuracy of the estimated low resolution image using a patch rank map. At each node, we order patches based on the likelihood given the estimated low resolution image, and show the rank of the true patch. Ideally, we want the true patch to have rank 1 at all nodes. We observe from the last column in Figure 3 that the patch rank is quite low in most nodes, except for nodes that correspond to foreground objects or the transition between the background and the foreground. On average (across all nodes and across 20 test images), the true patch has a rank 151 among 432 patches. We used the linear regression model because the kernel regression did not noticeably improve the quality of estimated low resolution images, yet required much more computation.

4.2. A sparse-and-accurate local evidence

We explore another strategy to emulate the low resolution image in Eq. (7). We study a scenario where some patches are associated with the correct positions in the puzzle. We name these patches the *anchor patches*. This is a generalization of the puzzle solving strategy that first fixes the four corner pieces and works its way inward. We show that the puzzle solver accuracy improves as we add more anchor patches and as the anchor patches are spread out uniformly across the image.

5. Solving the jigsaw puzzle

We reconstruct the jigsaw puzzle by maximizing $p(\mathbf{x})$ (Eq. (1)) using loopy belief propagation [4]. Since loopy belief propagation can fall into a local minimum, we run

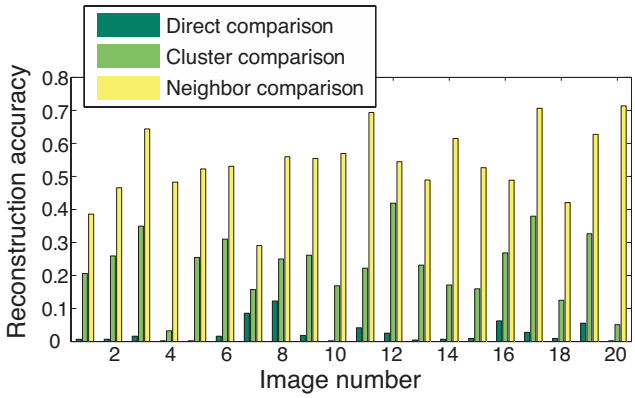


Figure 4. The image reconstruction accuracy with the estimated low resolution image for 20 different test images.

loopy belief propagation three times with random seeds and pick the best reconstruction in terms of the reconstruction accuracy. Each image is broken into 432 patches of size 28 x 28, which is down-sampled to 7 x 7 for low resolution image estimation.

Performance metrics While there has been an extensive work on solving jigsaw puzzles, there has not been any measure that evaluates the puzzle reconstruction performance because the previous works treated the puzzle solving as a binary problem. We propose three measures that gauge partial puzzle reconstruction performance:

- **Direct comparison:** the inferred patch labels are compared directly to the ground-truth patch labels. The reconstruction accuracy measures the fraction of nodes for which the algorithm inferred the correct patch.

- **Cluster comparison:** the inferred patch labels are mapped to clusters they belong to, and are compared to the ground-truth cluster labels. The reconstruction accuracy measures the fraction of nodes for which the algorithm inferred the correct cluster.

- **Neighbor comparison:** for each assigned patch label, we compute the fraction of four neighbor nodes that the algorithm assigned the correct patch (i.e. patches that were adjacent in the original image). The reconstruction accuracy is the average fraction of correct neighbor labels.

The direct comparison measure penalizes all patches that are assigned to wrong nodes, but the cluster comparison measure tolerates the assignment error as long as the assigned patch belong to the same cluster as the ground-truth patch. The neighbor comparison measure does not care about the exact patch assignment as long as patches that were adjacent in the original image remain adjacent.

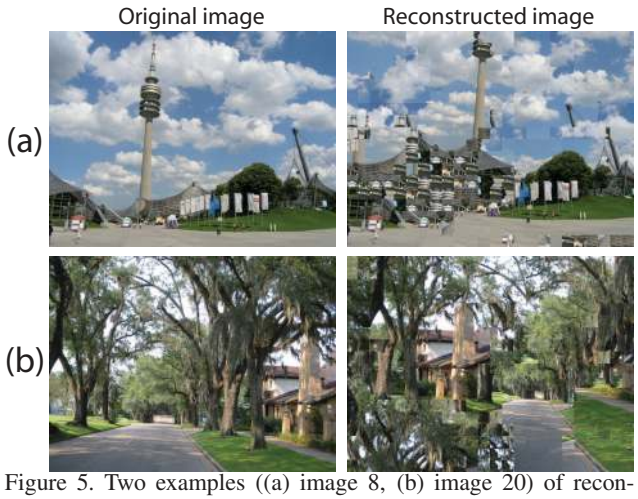


Figure 5. Two examples ((a) image 8, (b) image 20) of reconstructed images using the estimated local evidence.

5.1. Reconstruction with dense-and-noisy local evidence

We use the estimated low resolution image and the patch prior to solve the puzzle. The reconstruction accuracy for 20 test images is shown in Figure 4. Clearly, the graph suggests that it is hard to reconstruct the original image even given the estimated low resolution image.

To better understand Figure 4, we show two image reconstructions in Figure 5. The overall structure of reconstructed images is similar to that of the original images. Also, while parts of the image are not reconstructed properly, some regions are correctly assembled even though they may be offset from the correct position. The tower in Figure 5 (a) and the car road Figure 5 (b) have been laterally shifted. This can be attributed to the fact that the estimated low resolution image does not provide enough lateral information. Such shifts in image regions are not tolerated by the direct comparison measure and, possibly, the cluster comparison measure, but the neighbor comparison measure is more generous in this regard. In fact, under the neighbor comparison measure, the average reconstruction accuracy is nearly 55%, suggesting that many regions are assembled correctly but are slightly shifted.

5.2. Reconstruction with sparse-and-accurate local evidence

We study the jigsaw puzzle reconstruction performance with a sparse-and-accurate local evidence. In particular, we want to study how the number of anchor patches affect the image reconstruction accuracy. We run the image reconstruction experiments for 0 to 10 anchor patches.

Figure 6 illustrates that the location of anchor patches matters as well as the total number of anchor patches. If the anchor patches are more spread out, the image reconstruction performance improves. Therefore, we predefined the

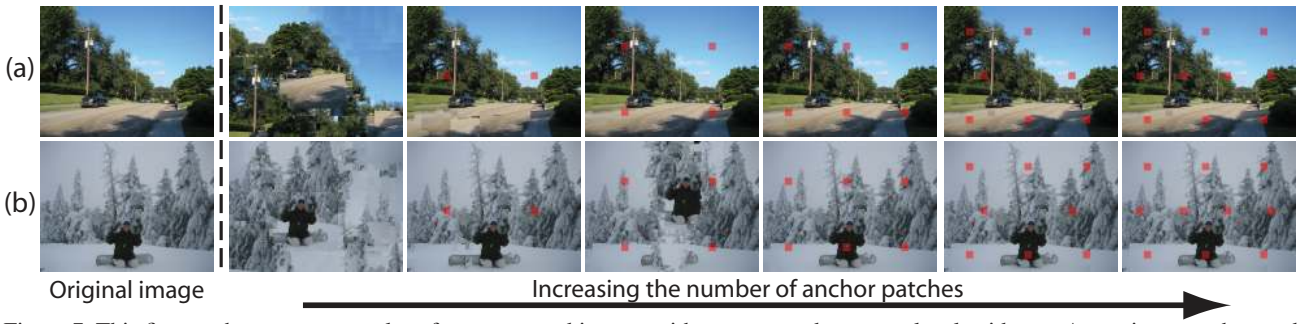


Figure 7. This figures shows two examples of reconstructed images with a sparse-and-accurate local evidence. As we increase the number of anchor patches (shown red), the algorithm's performance improves.

Figure 6. To improve the reconstruction accuracy, it's better to spread out the anchor patches (red) evenly across the image.

location of anchor patches such that they cover the image as uniformly as possible. This has an important consequence that even if we do not have anchor patches, we can loop over $\binom{432}{k}$ patch combinations to find the correct anchor patches at k predefined nodes. Figure 7 shows some image reconstruction results (see supplemental materials for more examples).

Figure 8(a) shows the reconstruction accuracy, averaged over the 20 test images. As expected, the average accuracy improves as we increase the number of anchor patches. Anchor patches serve as the local evidence for neighboring image nodes. As we add more anchor patches, more nodes become closer to anchor patches, and thus more nodes can reliably infer the correct patch label.

To calibrate the performance of the sparse-and-accurate local evidence scheme, we run another set of experiments with a quantized 6-bit true low resolution image. The reconstruction accuracy is overlaid on Figure 8. The performance of using a 6-bit true low resolution image is comparable to using $6-10$ anchor patches. This also suggests that solving the puzzle with the estimated low resolution image is extremely challenging. The estimated low resolution image should be as accurate as a 6-bit true low resolution image in order to perform comparably to using the sparse-and-accurate local evidence.

We also compared the performance of using the sparse-and-accurate local evidence to using a combination of anchor patches and the estimated low resolution image. The reconstruction performance is shown with dotted lines in Figure 8(a). When there are no anchor patches, the estimated low resolution image helps better reconstruct the original image. However, as we introduce anchor patches, on average, it is better *not* to have any noisy local evidence under all reconstruction measures. This is because the estimated low resolution image is too noisy.

5.3. Solving a smaller jigsaw puzzle

We have performed the same set of experiments on a smaller jigsaw puzzle. Each small jigsaw puzzle consists of 221 pieces. Figure 8(b) shows the reconstruction accuracy. The figure shows that we need fewer anchor patches to almost perfectly reconstruct images. In fact, we can perfectly reconstruct 5 images, and the top 15 images have the reconstruction accuracy higher than 90% under the most stringent direct comparison measure. A few images are difficult to reconstruct because they contain a large, uniform region.

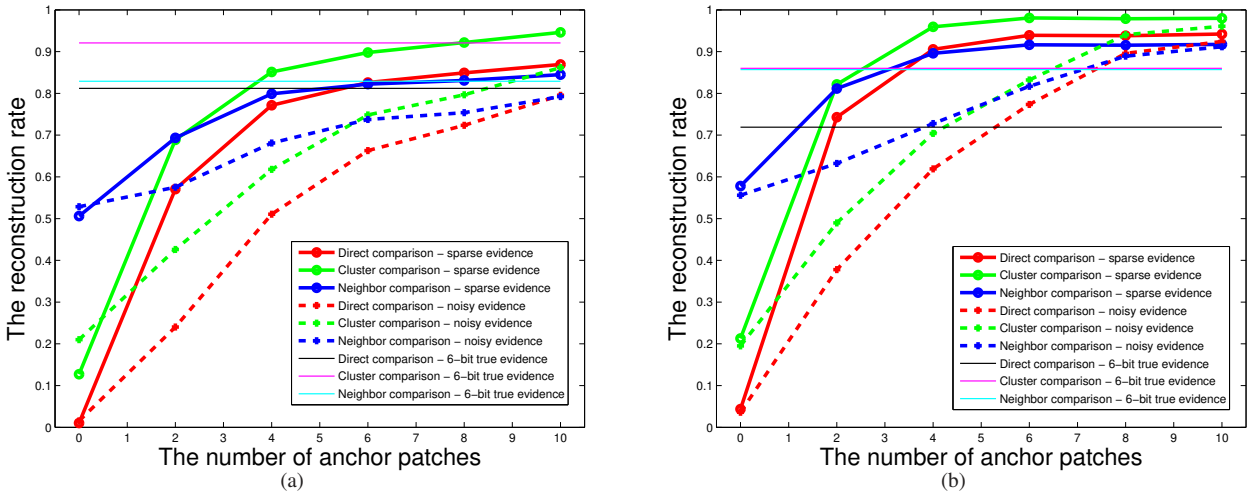
6. Conclusion

We introduce a probabilistic approach to solving jigsaw puzzles. A puzzle is represented as a graphical model where each node corresponds to a patch location and each label corresponds to a patch. We use loopy belief propagation to find the most likely configuration of patches on the graph. While we focused on solving a jigsaw puzzle with square pieces, we can easily augment the probabilistic framework to handle pieces with distinct shapes. The shaped pieces would improve the compatibility metric, improving the jigsaw puzzle reconstruction accuracy.

We have studied to what extent a bag of square image patches determines what images can be formed from edge-compatible placements of the image patches. The restricted class of images we can reconstruct from a histogram of image patches tells us about the structural regularity of images. This more general problem, of inferring what images are compatible with a given histogram of feature responses, occurs frequently in object recognition. Our work suggests an approach to try with more difficult sets of features such as Gabor jet histograms for texture representation, or bags of SIFT feature responses for object recognition.

Acknowledgment

This research is partially funded by NGA NEGI-1582-04-0004, by ONR-MURI Grant N00014-06-1-0734, and by



(a) (b)

Figure 8. The image reconstruction accuracy for two local evidence scenarios, as we increase the number of anchor patches, (a) with 432 jigsaw pieces. (b) with 221 jigsaw pieces. This figure shows that the image reconstruction improves as we increase the number of anchor patches, and that it's hard to reconstruct the image using the estimated low resolution image: the estimated low resolution image should be as accurate as a 6-bit true low resolution image to have comparable performance to using anchor patches.

gift from Microsoft, Google, Adobe. The first author is partially supported by Samsung Scholarship Foundation.

References

[1] C. Bishop. *Pattern recognition and machine learning*. Springer, 2006. 4

[2] B. J. Brown, C. Toler-Franklin, D. Nehab, M. Burns, D. Dobkin, A. Vlachopoulos, C. Doumas, and T. W. Szymon Rusinkiewicz. A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings. *ACM TOG (SIGGRAPH)*, 2008. 1

[3] C.-C. Chang, M.-S. Hwang, and T.-S. Chen. A new encryption algorithm for image cryptosystems. *Journal of Systems and Software*, 2001. 1

[4] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman. The patch transform and its applications to image editing. In *IEEE CVPR*, 2008. 1, 2, 3, 4, 5

[5] E. D. Demaine and M. L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23, 2007. 1, 2

[6] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, 2003. 4

[7] E.-J. Farn and C.-C. Chen. Novel steganographic method based on jig swap puzzle images. *Journal of electronic imaging*, 2009. 1

[8] H. Freeman and L. Garder. Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition. *IEEE TEC*, (13):118–127, 1964. 2

[9] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 2000. 3

[10] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. In *Symposium on Computational Geometry*, 2002. 2

[11] W. Kong and B. B. Kimia. On solving 2D and 3D puzzles using curve matching. In *IEEE CVPR*, 2001. 2

[12] D. A. Kosiba, P. M. Devaux, S. Balasubramanian, T. L. Gandhi, and K. Kasturi. An automatic jigsaw puzzle solver. In *IEEE ICPR*, volume 1, pages 616–618 vol.1, 1994. 2

[13] M. Makridis and N. Papamarkos. A new technique for solving a jigsaw puzzle. In *IEEE ICIP*, 2006. 2

[14] W. Marande and G. Burger. Mitochondrial DNA as a genomic jigsaw puzzle. *Science*, 2007. 1

[15] T. R. Nielsen, P. Drewsen, and K. Hansen. Solving jigsaw puzzles using image features. *PRL*, 2008. 2

[16] G. Radack and N. Badler. Jigsaw puzzle matching using a boundary-centered polar encoding. *CGIP*, 1982. 2

[17] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *IJCV*, 77:157–173, 2008. 4

[18] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *IEEE CVPR*, 2008. 3

[19] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *IEEE CVPR*, 2004. 3

[20] Y. Weiss and W. T. Freeman. What makes a good model of natural images? In *IEEE CVPR*, 2007. 3

[21] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan. Solving jigsaw puzzles by computer. *Annals of Operations Research*, 12(1-4):51–64, 1988. 2

[22] F.-H. Yao and G.-F. Shao. A shape and image merging technique to solve jigsaw puzzles. *PRL*, 2003. 2

[23] Y.-X. Zhao, M.-C. Su, Z.-L. Chou, and J. Lee. A puzzle solver and its application in speech descrambling. In *ICCEA*, 2007. 1

[24] L. Zhu, Z. Zhou, and D. Hu. Globally consistent reconstruction of ripped-up documents. *IEEE TPAMI*, 2008. 1