

Image Completion Using Efficient Belief Propagation via Priority Scheduling and Dynamic Pruning

Nikos Komodakis, Georgios Tziritas

University of Crete, Computer Science Department, P.O. Box 2208, Heraklion, Greece

E-mails: {komod,tziritas}@csd.uoc.gr

Phone: +30 2810 393585, Fax: +30 2810 393591

Abstract

In this paper, a new exemplar-based framework is presented, which treats image completion, texture synthesis and image inpainting in a unified manner. In order to be able to avoid the occurrence of visually inconsistent results, we pose all of the above image-editing tasks in the form of a discrete global optimization problem. The objective function of this problem is always well-defined, and corresponds to the energy of a discrete Markov Random Field (MRF). For efficiently optimizing this MRF, a novel optimization scheme, called Priority-BP, is then proposed, which carries two very important extensions over the standard Belief Propagation (BP) algorithm: “priority-based message scheduling” and “dynamic label pruning”. These two extensions work in cooperation to deal with the intolerable computational cost of BP, which is caused by the huge number of labels associated with our MRF. Moreover, both of our extensions are generic, since they do not rely on the use of domain-specific prior knowledge. They can therefore be applied to any MRF, *i.e.* to a very wide class of problems in image processing and computer vision, thus managing to resolve what is currently considered as one major limitation of the Belief Propagation algorithm: its inefficiency in handling MRFs with very large discrete state-spaces. Experimental results on a wide variety of input images are presented, which demonstrate the effectiveness of our image-completion framework for tasks such as object removal, texture synthesis, text removal and image inpainting.

Index Terms

Image completion, texture synthesis, Markov Random Fields, Belief Propagation, optimization.



Fig. 1: *Object removal* is just one of the many cases where image completion needs to be applied. In the specific example shown above, the user wants to remove a person from the input image on the left. He therefore simply marks a region around that person and that region must then be filled automatically so that a visually plausible outcome is obtained.

I. INTRODUCTION

The problem of image completion can be loosely defined as follows: given an image which is incomplete, *i.e* it has missing regions (*e.g* see Figure 1), try to fill its missing parts in such a way that a visually plausible outcome is obtained at the end. Although stating the image completion problem is very simple, the task of actually trying to successfully solve it, is far from being a trivial thing to achieve. Ideally, any algorithm that is designed to solve the image completion problem should have the following characteristics:

- it should be able to successfully complete complex natural images,
- it should also be able to handle incomplete images with (possibly) large missing parts and
- in addition, all these should take place in a fully automatic manner, *i.e* without intervention from the user.

Also, ideally, we would like any image completion algorithm to be able to handle the related problem of texture synthesis as well. According to that problem, given a small texture as input, we are then asked to generate an arbitrarily large output texture, which maintains the visual characteristics of the input (*e.g* see Figure 2(a)). It is exactly due to all of the above requirements that image completion is, in general, a very challenging problem. Nevertheless, it can be very useful in many areas, *e.g* it can be important for computer graphics applications, image editing, film post-production, image restoration, etc.

It has thus attracted a considerable amount of research over the last years. Roughly speaking, there have been three main approaches so far, for dealing with the image completion problem (see Fig. 2(b)):

- statistical-based methods,
- PDE-based methods,
- as well as exemplar-based methods.

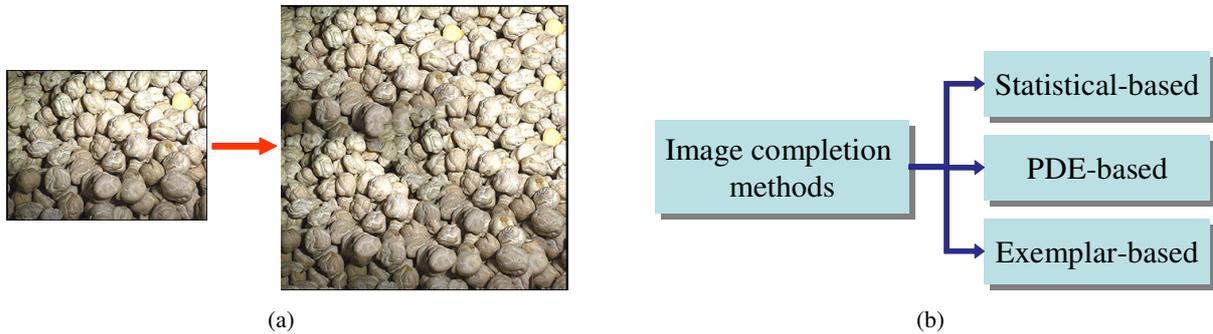


Fig. 2: (a) The texture synthesis problem. (b) The three main approaches to image completion.

In order to briefly explain the main limitations of current state-of-the-art methods for image completion, we next provide a short review of related work for each one of the three classes mentioned above.

Statistical-based methods: These methods are mainly used for the case of texture synthesis. Typically, what these methods do is that, given an input texture, they try to describe it by extracting some statistics through the use of compact parametric statistical models. *E.g* Portilla and Simoncelli [1] use joint statistics of wavelet coefficients for that purpose, while Heeger and Bergen [2] make use of color histograms at multiple resolutions for the analysis of the textures. Then, in order to synthesize a new texture, these methods typically start with an output image containing pure noise, and keep perturbing that image until its statistics match the estimated statistics of the input texture. Besides the synthesis of still images, parametric statistical models have been also proposed for the case of image sequences. *E.g* Soatto et al. [3] have proposed the so-called *dynamic texture* model, while a similar idea has been also described by Fitzgibbon in [4]. A parametric representation for image sequences had been previously presented by Szmur and Picard [5] as well. These parametric models for video have been mainly used for modeling and synthesizing dynamic stochastic processes, such as smoke, fire or water.

However, the main drawback of all methods that are based on parametric statistical models is that, as already mentioned, they are applicable only to the problem of texture synthesis, and not to the general problem of image completion. But even in the restricted case of texture synthesis, they can synthesize only textures which are highly stochastic and usually fail to do so for textures containing structure as well. Nevertheless, in cases where parametric models are applicable, they allow greater flexibility with respect to the modification of texture properties. *E.g* Doretto and Soatto [6] can edit the speed, as well as other properties of a video texture, by modifying the parameters of the statistical model they are using (which is a linear dynamical system in their case). Furthermore, these methods can be very useful for

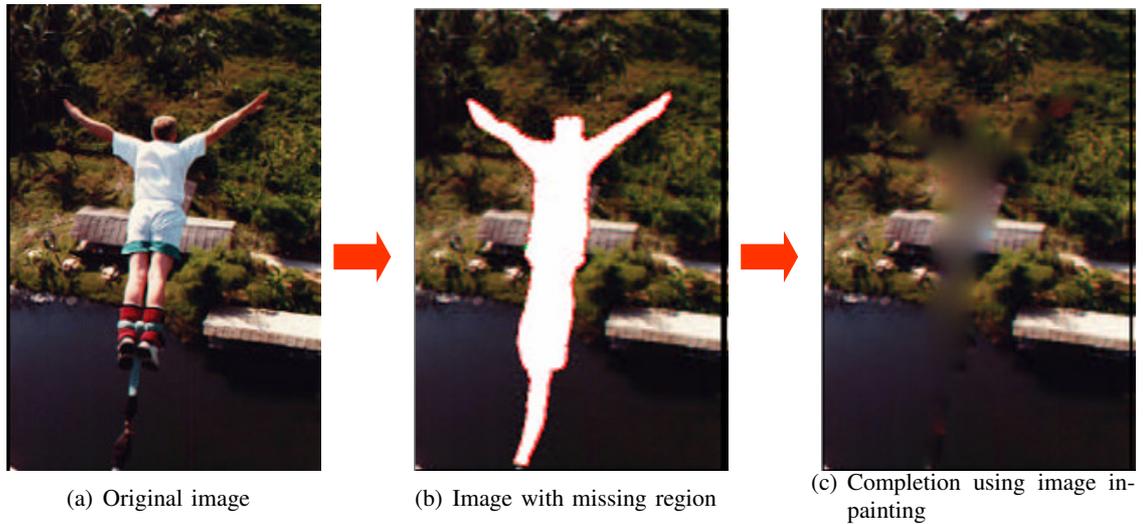


Fig. 3: Image inpainting methods, when applied to large or textured missing regions, very often oversmooth the image and introduce blurring artifacts.

the process which is reverse to texture synthesis, *i.e* the analysis of textures.

PDE-based methods: These methods, on the other hand, try to fill the missing region of an image through a diffusion process, by smoothly propagating information from the boundary towards the interior of the missing region. According to these techniques, the diffusion process is simulated by solving a partial differential equation (PDE), which is typically non-linear and of high order. This class of methods has been first introduced by Bertalmio et al. in [7], in which case the authors were trying to fill a hole in an image by propagating image Laplacians in the isophote direction. Their algorithm was trying to mimic the behavior of professional restorators in image restoration. In another case, the partial differential equations, that have been employed for the image filling process, were related to the Navier-Stokes equations in fluid dynamics [8], while Ballester et al. [9] have derived their own partial differential equations by formulating the image completion problem in a variational framework. Furthermore, recently, Bertalmio et al. [10] have proposed to decompose an image into two components. The first component is representing structure and is filled by using a PDE based method, while the second component represents texture and is filled by use of a texture synthesis method. Finally, Chan and Shen [11] have used an elastica based variational model for filling the missing part of an image.

However, the main disadvantage of almost all PDE based methods is that they are mostly suitable for image inpainting situations. This term usually refers to the case where the missing part of the image consists of thin, elongated regions. Furthermore, PDE-based methods implicitly assume that the content

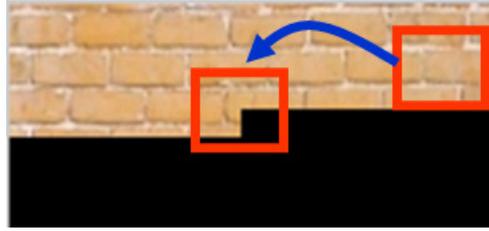


Fig. 4: Exemplar-based methods synthesize textures simply by copying patches from the observed part to the missing part of the image.

of the missing region is smooth and non-textured. For this reason, when these methods are applied to images where the missing regions are large and textured, they usually oversmooth the image and introduce blurring artifacts (*e.g.* see Figure 3). On the contrary, we would like our method to be able to handle images that contain possibly large missing parts. In addition to that, we would also like our method to be able to fill arbitrarily complex natural images, *i.e.* images containing texture, structure or even a combination of both.

Exemplar-based methods: Finally, the last class of methods consists of the so-called *exemplar-based* techniques, which actually have been the most successful techniques up to now. These methods try to fill the unknown region simply by copying content from the observed part of the image. Starting with the seminal work of Efros and Leung in [12], these methods have been mainly used for the purpose of texture synthesis. All exemplar-based techniques for texture synthesis that have appeared until now, were either pixel-based [13], [14], or patch-based [15], [16], [17], meaning that the final texture was synthesized one pixel, or one patch at a time (by simply copying pixels or patches from the observed image respectively). Somewhere in between is the method of Ashikhmin [18], where a pixel-based technique, that favors the copy of coherent patches, has been used in this case. Usually, patch-based methods achieve results of higher quality, since they manage to implicitly maintain higher order statistics of the input texture. Among patch-based methods, one should mention the work of Kwatra et al. [15], who managed to synthesize a variety of textures by making use of computer vision graph-cut techniques. Another interesting work is that of Hertzmann et al. [19], where the authors try to automatically learn painting styles from training data that consist of input-output image pairs. The painting styles, once learnt, can then be applied to new input images. Also, Efros and Freeman [20] use an exemplar-based method to perform *texture transfer*, *i.e.* rendering an object with a texture that has been taken from a different object. Exemplar-based methods for texture synthesis have been also used for the case of video. *E.g.* Schodl et al. [21] are able to synthesize new video textures simply by rearranging the recorded frames of an input video,

while the texture synthesis method of Kwatra et al. [15], that has been mentioned above, applies to image sequences as well.

As already explained in the previous paragraph, exemplar-based methods have been mainly used for the purpose of texture synthesis up to now. Recently, however, there have been a few authors who have tried to extend these methods to image completion as well. But, in this case, a major drawback of related approaches stems from their greedy way of filling the image, which can often lead to visual inconsistencies. Some techniques try to alleviate this problem by asking assistance from the user instead. *E.g* Jian Sun *et al* [22] require the user to specify the curves on which the most salient missing structures reside (thus obtaining a segmentation of the missing region as well), while Drori *et al* [23] use what they call “points of interest”. Also, some other methods [24] rely on already having a segmentation of the input image. But it is a well known fact that natural images segmentation is an extremely difficult task and, despite extensive research, no general method for reliably solving it currently exists. Some other methods [25], [26] are preferring to take a more global approach and formulate the problem in a way that a deterministic EM-like optimization scheme has to be used for image completion. It is well known, however, that expectation-maximization schemes are particularly sensitive to the initialization and may get easily trapped to poor local minima (thus violating the spirit of a global approach). For fixing this problem, one must resort to the use of multi-scale image completion. Although this might help in some cases, it is still not always safe. *E.g* any errors that may occur during the image completion process at the coarse scale, will probably carry through at finer scales as well. Finally, recent exemplar-based methods also place emphasis on the order by which the image synthesis proceeds, usually using a confidence map for this purpose [27], [23]. However, two are the main handicaps of related existing techniques. First, the confidence map is computed based on heuristics and ad hoc principles, that may not apply in the general case, and second, once an observed patch has been assigned to a missing block of pixels, that block cannot change its assigned patch thereafter. This last fact reveals the greediness of these techniques, which may again lead to visual inconsistencies.

In order to overcome all the limitations of the above mentioned methods, a new exemplar-based approach for image completion is proposed [28], which makes the following contributions:

- 1) Contrary to greedy synthesis methods, we pose image completion as a discrete global optimization problem with a well defined objective function. In this manner, we are able to avoid the occurrence of visual inconsistencies during the image completion process, and manage to produce visually plausible results.
- 2) No user intervention is required by our method, which manages to avoid greedy patch assignments

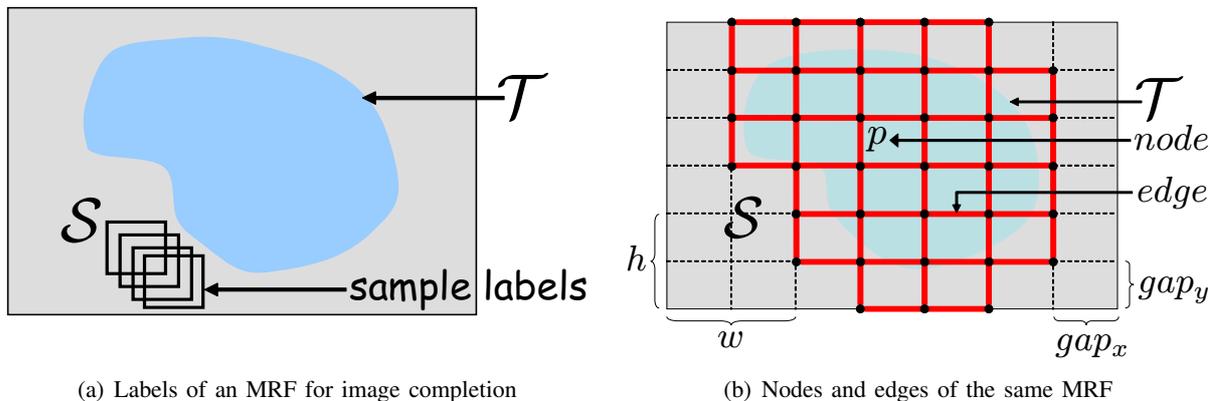


Fig. 5: (a) The labels associated with the image completion problem are chosen to be all $w \times h$ patches of the source region \mathcal{S} . (b) In this figure, we show the nodes (black dots) and edges (red segments) of the MRF that will be used during image completion. For this particular example, the w, h parameters were set equal to $w = 2gap_x, h = 2gap_y$.

by maintaining (throughout its execution) many candidate source patches for each block of missing pixels. In this way, each missing block of pixels is allowed to change its assigned patch many times throughout the execution of the algorithm, and is not enforced to remain tied to the first label that has been assigned to it during the early stages of the completion process.

- 3) Our formulation applies not only to image completion, but also to texture synthesis, as well as to image inpainting, thus providing a unified framework for all of these tasks.
- 4) To this end, a novel optimization scheme is proposed, the “Priority-BP” algorithm, which carries 2 major improvements over standard belief propagation: “*dynamic label pruning*” and “*priority-based message scheduling*”. Together, they bring a dramatic reduction in the overall computational cost of BP, which would otherwise be intolerable due to the huge number of existing labels. We should finally note that both extensions are generic and can be used for the optimization of any MRF (*i.e* they are applicable to a very wide class of problems in image processing and computer vision). Priority-BP can thus be viewed as a generic way for efficiently applying belief-propagation to MRFs with very large discrete state-spaces, thus dealing, for the first time, with what was considered as one of the main limitations of BP up to now.

II. IMAGE COMPLETION AS A DISCRETE GLOBAL OPTIMIZATION PROBLEM

Given an input image \mathcal{I}_0 , as well as a *target region* \mathcal{T} and a *source region* \mathcal{S} (where \mathcal{S} is always a subset of $\mathcal{I}_0 - \mathcal{T}$), the goal of image completion is to fill \mathcal{T} in a visually plausible way simply by copying patches from \mathcal{S} . We propose to turn this into a discrete optimization problem with a well defined objective

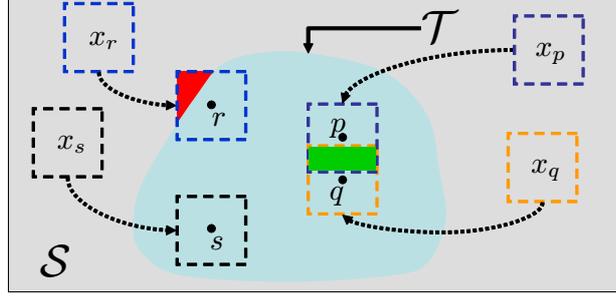


Fig. 6: An illustration of how the MRF potential functions are computed: for the boundary node r , its label cost $V_r(x_r)$ will be an SSD over the red region, while for nodes p, q their potential $V_{pq}(x_p, x_q)$ will be an SSD over the green region. Node s is an interior node and so its label cost $V_s(x_s)$ will always be zero.

function. To this end, we propose the use of the following discrete Markov Random Field (MRF):

The labels \mathcal{L} of the MRF will consist of all $w \times h$ patches from the source region \mathcal{S}^1 (see Figure 5(a)). For defining the nodes of the MRF, an image lattice will be used with an horizontal and vertical spacing of gap_x and gap_y pixels respectively. The MRF nodes \mathcal{V} will be all lattice points whose $w \times h$ neighborhood intersects the target region, while the edges \mathcal{E} of the MRF will make up a 4-neighborhood system on that lattice (see Figure 5(b)).

In this context, assigning a label (*i.e* a patch) to a node, will amount to copying that patch over the node's position in the image space. Based on this fact, we will next define the energy of our MRF, so that only patches that are consistent with each other, as well as with the observed region, are allowed to be copied into the missing region. To this end, the single node potential $V_p(x_p)$ (called *label cost* hereafter), for placing patch x_p over node p , will encode how well that patch agrees with the source region around p , and will equal the following sum of squared differences (SSD):

$$V_p(x_p) = \sum_{dp \in [-\frac{w}{2}, \frac{w}{2}] \times [-\frac{h}{2}, \frac{h}{2}]} \mathcal{M}(p + dp) (\mathcal{I}_0(p + dp) - \mathcal{I}_0(x_p + dp))^2. \quad (1)$$

In the above formula, $\mathcal{M}(\cdot)$ denotes a binary mask, which is non zero only inside region \mathcal{S} (due to this fact, *i.e* due to $\mathcal{M}(\cdot)$ being zero outside \mathcal{S} , the label costs of interior nodes, *i.e* nodes whose $w \times h$ neighborhood does not intersect \mathcal{S} , will obviously be all zero). In a similar fashion, the pairwise potential $V_{pq}(x_p, x_q)$, due to placing patches x_p, x_q over neighbors p, q , will measure how well these patches agree at the resulting region of overlap, and will again be given by the SSD over that region (see Figure 6). Note that gap_x and gap_y are set so that such a region of overlap always exists.

¹Hereafter each label (*i.e* patch) will be represented by its center pixel

Based on this formulation, our goal will then be to assign a label $\hat{x}_p \in \mathcal{L}$ to each node p so that the total energy $\mathcal{F}(\hat{x})$ of the MRF is minimized, where:

$$\mathcal{F}(\hat{x}) = \sum_{p \in \mathcal{V}} V_p(\hat{x}_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(\hat{x}_p, \hat{x}_q). \quad (2)$$

Intuitively, any algorithm optimizing this energy is, roughly speaking, trying to assemble a huge jigsaw puzzle, where the source patches correspond to the puzzle pieces, while region \mathcal{T} represents the puzzle itself.

One important advantage of our formulation is that it also provides a unified framework for texture synthesis and image inpainting. *E.g.* to handle texture synthesis (where one wants to extend an input texture T_0 to a larger region T_1), one suffices to set $\mathcal{S} = T_0$ and $\mathcal{T} = T_1 - T_0$. Put otherwise, you place the input texture T_0 at any location inside the larger region T_1 and then you let the image completion algorithm fill the remaining part of region T_1 . Moreover, thanks to the fact that we reduce image completion to an energy optimization problem, our framework allows the use of (what we call) “*completion by energy refinement*” techniques, one example of which we will see later.

III. PRIORITY-BP

Furthermore, an additional advantage is that we now can hopefully apply belief propagation (*i.e.* a state-of-the-art optimization method) to our energy function. Unfortunately, however, this is not the case. The reason is the intolerable computational cost of BP, caused by the huge number of existing labels. Motivated by this fact, one other major contribution of this work is the proposal of a novel MRF optimization scheme, called Priority-BP, that can deal exactly with this type of problems, and carries two significant extensions over standard BP: one of them, called *dynamic label pruning*, is based on the key idea of drastically reducing the number of labels. However, instead of this happening beforehand (which will almost surely lead to throwing away useful labels), pruning takes place on the fly (*i.e.* while BP is running), with a (possibly) different number of labels kept for each node. The important thing to note is that only the beliefs calculated by BP are used for that purpose. This is exactly what makes the algorithm generic and applicable to any MRF. Furthermore, the second extension, called *priority-based message scheduling*, makes use of label pruning and allows us to always send cheap messages between the nodes of the graphical model. Moreover, it considerably improves BP’s convergence, thus accelerating completion even further.

The significance of our contribution also grows due to the fact that (as we shall see) Priority-BP is a generic algorithm, applicable to any MRF energy function. This is unlike any prior use of Belief

Propagation [29] and, therefore, our method resolves, for the first time, what is currently considered as one of the main limitations of BP: its inefficiency to handle problems with a huge number of labels. In fact, this issue has been a highly active research topic over the last years. Until now, however, the techniques that have been proposed were valid only for restricted classes of MRFs [30], [31]. Not only that, but our priority-based message scheduling scheme can be used (independently of label pruning) as a general method for accelerating the convergence of BP.

A. Loopy Belief-Propagation

In order to explain the reason why belief-propagation has an intolerable computational cost in our case, we will first briefly describe in this section how that algorithm works. Belief propagation is an iterative algorithm, which tries to find a MAP estimate by iteratively solving a finite set of equations until a fixed point is obtained [32]. However, before one is able to understand how this set of equations comes up, he must first get acquainted with the notion of “*messages*”, which is another central concept in BP. In fact, belief propagation does nothing more than continuously propagating local messages between the nodes of an MRF graph. At every iteration, each node sends messages to all of its neighboring nodes, while it also accepts messages from these nodes. This process repeats until all the messages have stabilized, *i.e.* they do not change any more. Therefore, the set of equations, whose fixed point one tries to find, actually corresponds to the equations governing how messages are updated during each iteration.

The set of messages sent from a node p to a neighboring node q will be denoted by $\{m_{pq}(x_q)\}_{x_q \in \mathcal{L}}$. Therefore, the number of such messages is always $|\mathcal{L}|$ (*i.e.* there exists one message per label in \mathcal{L}). Intuitively, message $m_{pq}(x_q)$ expresses the opinion of node p about assigning label x_q to node q . Furthermore, whenever we say that node p sends a message $m_{pq}(x_q)$ to node q , what we practically mean is that the following recursive update of the message $m_{pq}(x_q)$ is taking place:

$$m_{pq}(x_q) = \min_{x_p \in \mathcal{L}} \left\{ V_{pq}(x_p, x_q) + V_p(x_p) + \sum_{r: r \neq q, (r,p) \in \mathcal{E}} m_{rp}(x_p) \right\} \quad (3)$$

The interpretation of the above equation is that, if node p wants to send the message $m_{pq}(x_q)$ to node q (*i.e.* if node p wants to tell q its opinion about label x_q), then node p must first traverse each one of its own labels $x_p \in \mathcal{L}$, and then decide which one of them provides the greatest support for assigning label x_q to node q . The support of label x_p with respect to the assignment of x_q to node q is determined based on two factors:

- the compatibility between labels x_p and x_q (measured by the term $V_{pq}(x_p, x_q)$ in (3)), and

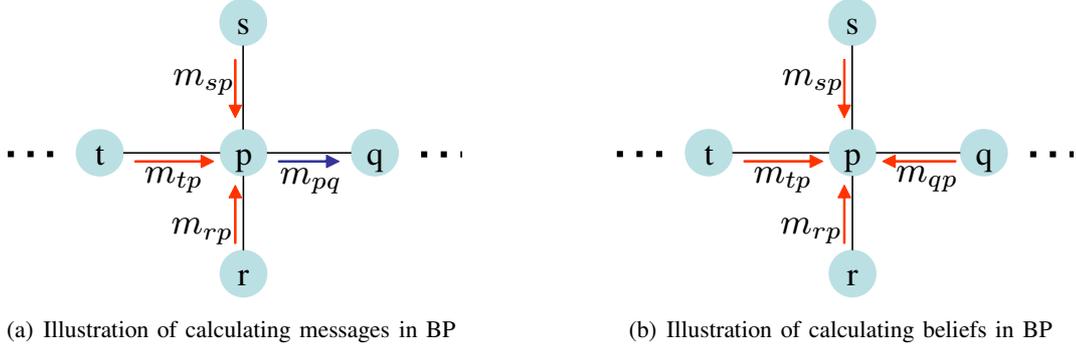


Fig. 7: (a) If a node p wants to send a message $m_{pq}(x_q)$ to a neighboring node q , then it must make use of the messages $m_{sp}(x_p), m_{rp}(x_p), m_{tp}(x_p)$ coming from the rest of its neighbors. (b) If a node p wants to calculate its belief $b_p(x_p)$ about any of the labels $x_p \in \mathcal{L}$, it must then collect the messages $m_{sp}(x_p), m_{qp}(x_p), m_{rp}(x_p), m_{tp}(x_p)$ coming from all of its neighboring nodes.

- the likelihood of assigning label x_p to node p as well. Obviously, on one hand, this likelihood will depend on the observed data at node p (see term $V_p(x_p)$ in (3)). On the other hand, node p must also ask for the opinion of its other neighbors about label x_p (this is measured by the sum $\sum_{r:r \neq q, (r,p) \in \mathcal{E}} m_{rp}(x_p)$ in (3)).

Therefore, before a node p sends a message to another node q , he must first consult the rest of its neighbors by receiving messages from them (see Figure 7(a)). Put otherwise, during belief propagation, all MRF nodes work in cooperation in order to make a decision about the labels that they should finally choose. This cooperation between nodes is reflected by the exchange of opinions, *i.e.* messages, which is taking place during the algorithm.

The updating of messages according to equation (3) continues until all messages finally converge, *i.e.* until all nodes agree with each other about which labels are appropriate for them. Then, after convergence, a set of so-called *beliefs* $\{b_p(x_p)\}_{x_p \in \mathcal{L}}$ is computed for every node p in the MRF. These beliefs are estimated using the following equation:

$$b_p(x_p) = -V_p(x_p) - \sum_{r:(r,p) \in \mathcal{E}} m_{rp}(x_p). \quad (4)$$

Intuitively, the belief $b_p(x_p)$ expresses the probability of assigning label x_p to p , and for estimating it, node p must first gather all messages (*i.e.* all opinions) coming from its neighboring nodes (which is accounted by the sum $\sum_{r:(r,p) \in \mathcal{E}} V_{rp}(x_p)$ in equation (4), see also Figure 7(b)). Based on the above observations,

once all beliefs have been computed, each node is then assigned the label having the maximum belief:

$$\hat{x}_p = \arg \max_{x_p \in \mathcal{L}} b_p(x_p). \quad (5)$$

Strictly speaking, beliefs actually approximate the so-called max-marginals², *i.e* each belief $b_p(x_p)$ approximates the maximum conditional probability that can be obtained, given the fact that node p has already been assigned the label x_p .

It can be proved that, in the case of a tree structured graph, belief propagation is exact, *i.e* the exact global optimum of the MRF energy is computed. Not only that, but it can be actually shown that this global optimum may be computed in just one iteration. However, if the MRF graph contains cycles, then no such guarantee can be provided. Moreover, many iterations are then needed for the algorithm to converge (this also explains why the algorithm has been given the name *loopy belief propagation* in this case). Experimentally, however, it has been proved that Belief Propagation typically produces strong local minimum (*i.e* solutions that are close to the optimum) for a wide variety of tasks in computer vision.

At this point, it is also important to note that, for the Belief Propagation algorithm to work properly, one must ensure that all the messages get transmitted during any of the algorithm’s iterations. This practically means that, for each edge (p, q) in \mathcal{E} , one must ensure that all the messages from p to q , as well as all the messages from q to p (*i.e* all the messages in both directions) are transmitted at each iteration.

Finally, we should note that there are actually two versions of the Belief Propagation method: the “*max-product*” and the “*sum-product*” algorithm. The difference is that “*sum-product*” computes the marginal posterior of each node, while “*max-product*” maximizes the posterior of each node, *i.e* it computes the max-marginals. The algorithm that has been presented above corresponds to the “*max-product*” version, but, due to the fact that we are using negative log probabilities, we are left with the task of minimizing sums of terms, instead of maximizing products of terms (as normally done in the “*max-product*” algorithm).

B. Priority-based message scheduling

In the form presented above, however, BP is impractical for problems with a large number of labels like ours. In particular, if $|\mathcal{L}|$ is the total number of labels (which, in our case, can be many many thousands) then just the basic operation of updating the messages from one node p to another node q takes $O(|\mathcal{L}|^2)$ time. In fact, the situation is much more worse for us. The huge number of labels also implies that for

²Actually, as we are working in the $-\log$ domain, beliefs approximate the min-marginals, *e.g* $b_p(x_p)$ approximates the minimum energy that can be obtained given that node p has already been assigned label x_p .

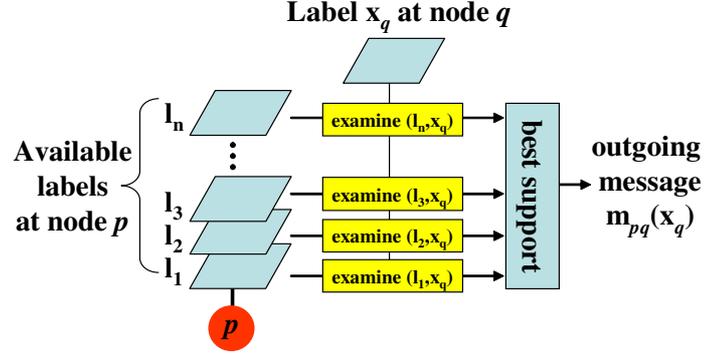


Fig. 8: If node p wants to send a message to node q about label x_q , he must then examine each one of its own labels and see which one provides the best support for label x_q . Hence, the cost of any outgoing message from p will be $O(n)$, where n is the number of available labels at p .

any pair of adjacent nodes p, q their matrix of pairwise potentials $V_{pq}(\cdot, \cdot)$ is so large that cannot fit into memory and therefore cannot be precomputed. That matrix therefore must be reestimated every time node p sends its messages to node q , meaning that $|\mathcal{L}|^2$ SSD calculations (between image patches) are needed for each such update.

To deal with this issue we will try to reduce the number of labels by exploiting the beliefs calculated by BP. However not all nodes have beliefs which are adequate for this purpose in our case. To see that, it suffices to observe that the label costs at all interior nodes are all equal to zero. This in turn implies that the beliefs at an interior node will initially be all equal as well, meaning that the node is “unconfident” about which labels to prefer. No label pruning may therefore take place and so any message originating from that node will be very expensive to calculate *i.e* it will take $O(|\mathcal{L}|)$ time. On the contrary, if we had a node whose labels could be pruned (and assuming that the maximum number of labels after pruning is L_{max} with $L_{max} \ll |\mathcal{L}|$) then any message from that node would take only $O(L_{max})$ time.

Based on this observation we therefore propose to use a specific message scheduling scheme, whose goal will be twofold. On one hand, it will make label pruning possible and favor the circulation of cheap messages. On the other hand, it will speed up BP’s convergence. This issue of BP message scheduling, although known to be crucial for the success of BP, it has been largely overlooked until now. Also, to the best of the authors’ knowledge it is the first time that message scheduling is used in this manner for general graphical models. Roughly, our message scheduling scheme will be based on the notion of priorities that are assigned to the nodes of the MRF. Any such priority will represent a node’s confidence about which labels to prefer and will be dynamically updated throughout the algorithm’s execution. Our message scheduling will then obey the following simple principle:

Algorithm 1 Priority-BP

assign priorities to nodes and declare them uncommitted
for $k = 1$ to K **do** $\{K$ is the number of iterations $\}$
 execute *ForwardPass* and then *BackwardPass*
 assign to each node p its label \hat{x}_p that maximizes $b_p(\cdot)$

ForwardPass:

for time = 1 to N **do** $\{N$ is the number of nodes $\}$
 $p =$ “uncommitted” node of highest priority
 apply “label pruning” to node p
 forwardOrder[time] = p ; $p \rightarrow$ committed = true;
 for any “uncommitted” neighbor q of node p **do**
 send all messages $m_{pq}(\cdot)$ from node p to node q
 update beliefs $b_q(\cdot)$ as well as priority of node q

BackwardPass:

for time = N to 1 **do**
 $p =$ forwardOrder[time]; $p \rightarrow$ committed = false;
 for any “committed” neighbor q of node p **do**
 send all messages $m_{pq}(\cdot)$ from node p to node q
 update beliefs $b_q(\cdot)$ as well as priority of node q

Message-scheduling principle. *The node most confident about its labels should be the first one (i.e it has the highest priority) to transmit outgoing messages to its neighbors.*

The above way of scheduling messages during BP is good for the following reasons:

- The first reason is that the more confident a node is, the more label pruning it can tolerate (before sending its outgoing messages) and therefore the cheaper these messages will be. We recall here that the cost of an outgoing message from a node p will be proportional to the number of available labels for that node (see Figure 8).
- The second reason is that, in this way, we also help other nodes become more amenable to pruning. Intuitively, this happens because the more confident a node is, the more informative its messages are going to be, meaning that these messages can help the neighbors of that node to increase their own confidence and thus these nodes will become more tolerable to pruning as well.
- Furthermore, by first propagating the most informative messages around the graphical model, we also help BP to converge much faster. This has been verified experimentally as well. *E.g* Priority-BP never needed more than a small fixed number of iterations to converge for all of our image completion examples.

A pseudocode description of Priority-BP is contained in algorithm 1. Each iteration of Priority-BP is

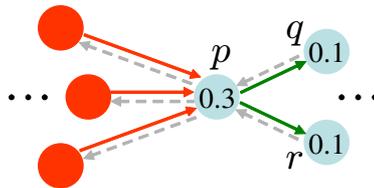


Fig. 9: Message scheduling during the forward pass: currently only red nodes have been committed and only messages on red edges have been transmitted. Among uncommitted nodes (*i.e* blue nodes) the one with the highest priority (*i.e* node p) will be committed next and will also send messages only along the green edges (*i.e* only to its uncommitted neighbors q, r). Messages along dashed edges will be transmitted during the backward pass. Priorities are indicated by the numbers inside uncommitted nodes.

divided into a forward and a backward pass. The actual message scheduling mechanism as well as label pruning takes place during the forward pass. This is also where one half of the messages gets transmitted (*i.e* each MRF edge is traversed in only one of the 2 directions). To this end, all nodes are visited in order of priority. Each time we visit a node, say p , we mark it as “committed” meaning that we must not visit him again during the current forward pass. We also prune its labels and then allow him to transmit its “cheap” (due to pruning) messages to all of its neighbors apart from the committed ones (as these have already sent a message to p during the current pass). The priorities of all neighbors that received a new message are then updated and the process continues with the next uncommitted (*i.e* unvisited) node of highest priority until no more uncommitted nodes exist.

The role of the backward pass is then just to ensure that the other half of the messages gets transmitted as well. To this end, we do not make use of priorities but simply visit the nodes in reverse order (with respect to the order of the forward pass) just transmitting the remaining unsent messages from each node. For this reason, no label pruning takes place during this pass. We do update node priorities, though, so that they are available during the next forward pass.

Also, as we shall see, a node’s priority depends only on the current beliefs at that node. One big advantage out of this is that keeping the node priorities up-to-date can be done very efficiently in this case, since only priorities for nodes with newly received messages need to be updated. The message scheduling mechanism is further illustrated in Figure 9.

C. Assigning priorities to nodes

It is obvious that our definition of priority will play a very crucial role for the success of the algorithm. As already mentioned priority must relate to how confident a node is about the labels that should be assigned to him, with the more confident nodes having higher priority. An important thing to note in

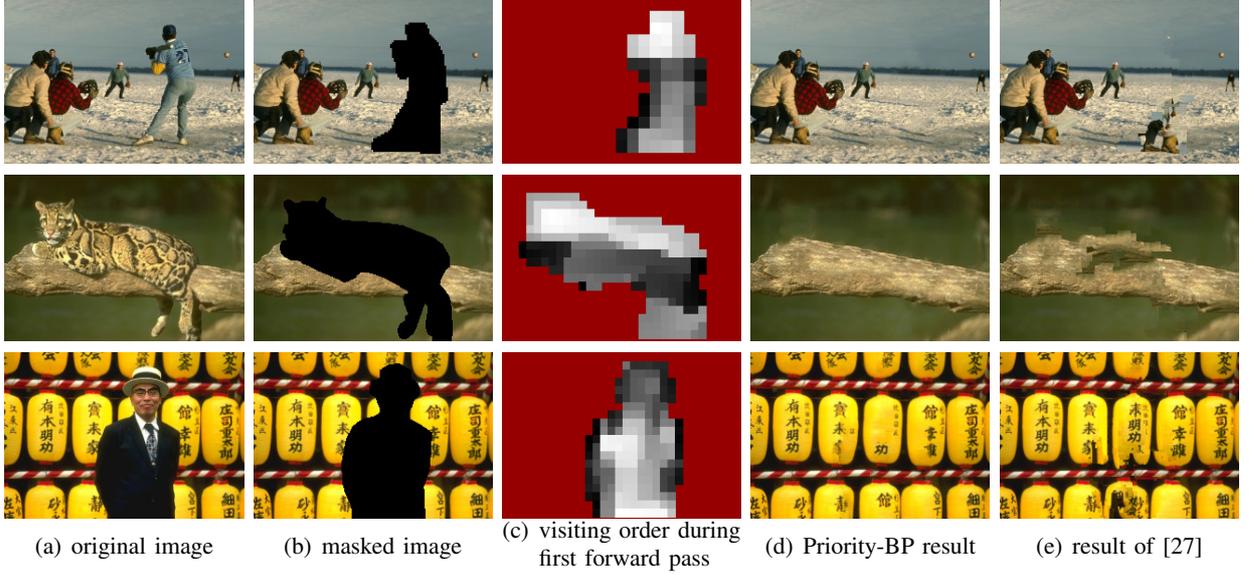


Fig. 10: In column (c) darker patches correspond to nodes that are visited earlier during message scheduling at the first forward pass

our case is that the confidence of a node will depend solely on information that will be extracted by the BP algorithm itself. This makes our algorithm generic (*i.e.* applicable to any MRF energy function) and therefore appropriate for a very wide class of problems.

In particular, our definition of confidence (and therefore priority as well) for node p will depend only on the current set of beliefs $\{b_p(x_p)\}_{x_p \in \mathcal{L}}$ that have been estimated by the BP algorithm for that node. Based on the observation that belief $b_p(x_p)$ is roughly related to how likely label x_p is for node p , one way to measure the confidence of this node is simply by counting the number of likely labels *e.g.* those whose belief exceed a certain threshold b_{conf} . The intuition for this is that the greater this number, the more labels with high probability exist for that node and therefore the less confident that node turns out to be about which specific label to choose. And vice versa, if this number is small then node p needs to choose its label only among a small set of likely labels. Of course, only relative beliefs $b_p^{rel}(x_p) = b_p(x_p) - b_p^{max}$ (where $b_p^{max} = \max_{x_p \in \mathcal{L}} b_p(x_p)$) matter in this case and so by defining the set $\mathcal{CS}(p) = |\{x_p \in \mathcal{L} : b_p^{rel}(x_p) \geq b_{conf}\}|$ (which we will call the *confusion set* of node p hereafter) the priority of p is then inversely related to the cardinality of that set:

$$\text{priority}(p) = \frac{1}{|\mathcal{CS}(p)|} \quad (6)$$

This definition of priority also justifies why during either the forward or the backward pass we were

allowed to update priorities only for nodes that had just received new incoming messages: the reason is that the beliefs (and therefore the priority) of a node may change only if at least one incoming message to that node changes as well (this is true due to the way beliefs are defined *i.e.* $b_p(x_p) = -V_p(x_p) - \sum_{r:(r,p) \in \mathcal{E}} m_{rp}(x_p)$). Although we tested other definitions of priority as well (*e.g.* by using an entropy-like measure on beliefs), the above criterion for quantifying confidence gave the best results in practice by far.

D. Applying Priority-BP to image completion

We pause here for a moment (postponing the description of label pruning to the next section) in order to stress the advantages of applying our algorithm to image completion, while also showing related results.

First of all, we should mention that, although confidence has already been used for guiding image completion in other works as well [27], [23], our use of confidence differs (with respect to these approaches) in two very important aspects. The first is that we use confidence in order to decide upon the order of BP message passing and not for greedily deciding which patch to fill next. These are two completely different things: the former is part of a principled global optimization procedure, while the latter just results in patches that cannot change their appearance after they have been filled.

The second aspect is that in all of the previous approaches the definition of confidence was mostly based either on heuristics or on ad hoc principles that were simply making use of application-specific knowledge about the image completion process. On the contrary, as we saw, our definition of confidence is generic and therefore applicable to any kind of images. Moreover, this way our method is placed on firm theoretical grounds.

Three examples of applying Priority-BP to image completion are shown in Figure 10. As can be seen, the algorithm has managed to fill the missing regions in a visually plausible way. The third column in that figure shows the visiting order of the nodes during the first forward pass (based on our definition of priority). The darker a patch is in these images, the earlier the corresponding node was visited. Notice how the algorithm learns by itself how to propagate first the messages of the nodes containing salient structure, where the notion of saliency depends on each specific case. *E.g.* the nodes that are considered salient for the first example of Figure 10 are those lying along the horizon boundary. On the contrary, for the second example of that figure the algorithm prefers to propagate information along the MRF edges at the interior of the wooden trunk first. *The remarkable thing is that in both cases such information was not explicitly encoded but was, instead, inferred by the algorithm.*

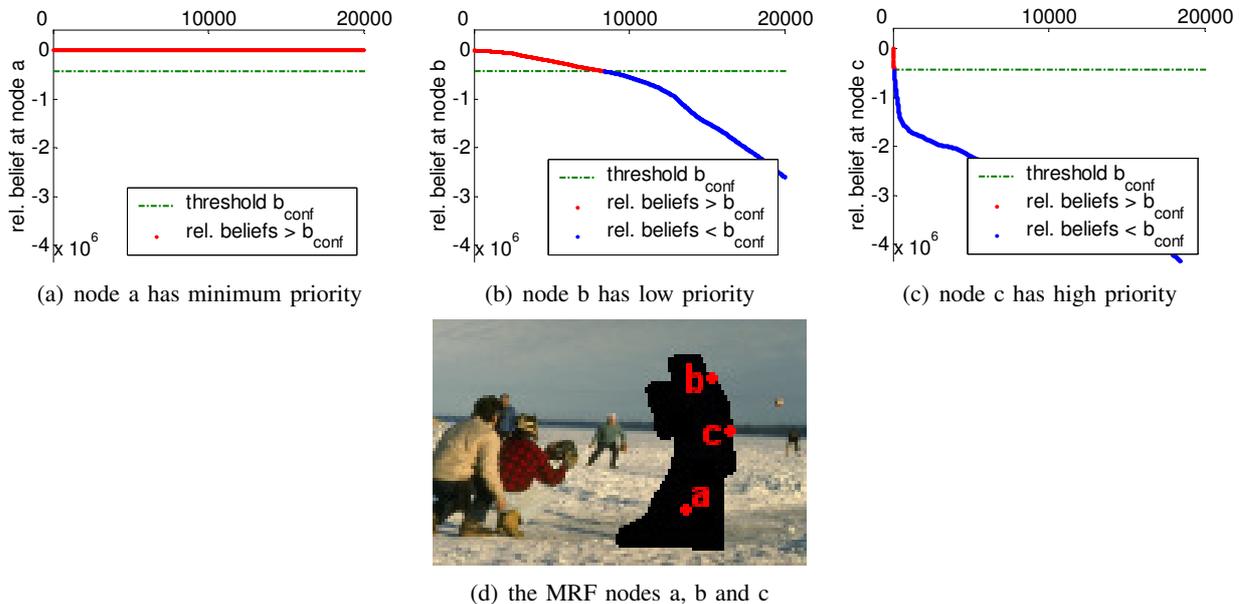


Fig. 11: The plots in (a), (b) and (c) show the sorted relative beliefs for the MRF nodes a, b and c in figure (d) at the start of Priority-BP. Relative beliefs plotted in red correspond to labels in the confusion set. This set determines the priority of the corresponding node.

This is in contrast to the state-of-the-art method in [27], where the authors had to hardwire isophote-related information into the definition of priority (*i.e* a measure which is not always reliably extracted or even appropriate *e.g* in images with texture). The corresponding results produced by that method are shown in the last column of Figure 10. In these cases, only one label (*i.e* patch) is greedily assigned to each missing block of pixels and so any errors made early cannot be later backtracked, thus leading to the observed visual inconsistencies. On the contrary, due to our global optimization approach, any errors that are made during the very first iterations can be very well corrected later, since our algorithm always maintain not one but many possible labels for each MRF node. A characteristic case for this is the third example in Figure 10, where unless one employs a global optimization scheme it is not easy to infer the missing structure.

Also, the plots in Figures 11(a), 11(b), 11(c) illustrate our definition of priority in (6). They display the largest 20000 relative beliefs (sorted in ascending order) that are observed at the very beginning of the algorithm for each of the MRF nodes a, b, c in Figure 11(d) respectively. Relative beliefs plotted in red correspond to labels in the confusion set. Node a , being an interior node, has initially all the labels in its confusion set (since their relative beliefs are all zero) and is therefore of lowest priority. Node b still has too many labels in its confusion set due to the uniform appearance of the source region around

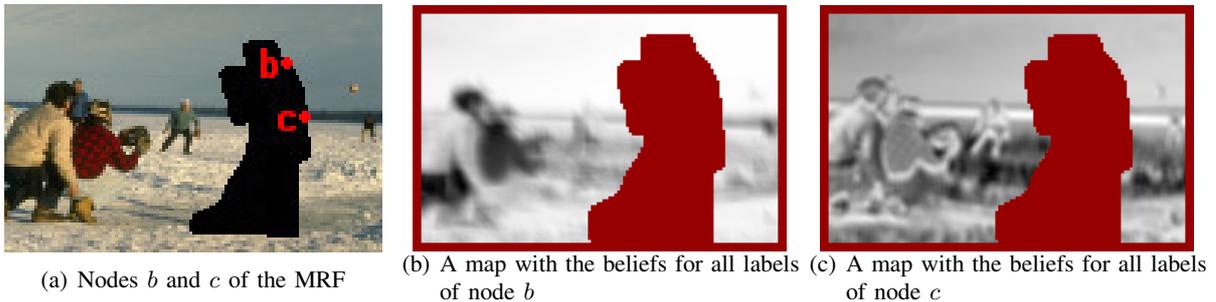


Fig. 12: Figure (b) visualizes the beliefs (as estimated at the start of Priority-BP) for all the labels of node b . Figure (c) does the same thing for all the beliefs of node c . In these visualizations, the whiter a pixel, the higher the belief of the corresponding label. We recall that the label corresponding to a pixel is the patch having that pixel as its center.

that node. On the contrary, node c is one of the nodes to be visited early during the first forward pass, since only very few labels belong to its confusion set. Indeed, even at the very beginning, we can easily exclude (*i.e.* prune) many source patches from being labels of that node, without the risk of throwing away useful labels. This is why Priority-BP prefers to visit him early.

This is also clearly illustrated in Figures 12(b) and 12(c), where we visualize the beliefs that have been estimated at the start of Priority-BP for all the labels of nodes b and c respectively. In these figures, the whiter a pixel appears, the higher the belief for the corresponding label is (we recall that the label corresponding to a pixel is the patch having that pixel as its center).

E. Label pruning

The main idea of “label pruning” is that, as we are visiting the nodes of the MRF during the forward pass (in the order induced by their priorities), we dynamically reduce the number of possible labels for each node by discarding labels that are unlikely to be assigned to that node. In particular, after committing a node, say p , all labels having a very low relative belief at p , say less than b_{prune} , are not considered as candidate labels for p thereafter. The remaining labels are called the “*active labels*” for that node. An additional advantage we gain in this way is that after all MRF nodes have pruned their labels at least once (*e.g.* at the end of the first forward pass) then we can precompute the reduced matrices of pairwise potentials (which can now fit into memory) and thus greatly enhance the speed of our algorithm. The important thing to note is that “label pruning” relies only on information carried by the Priority-BP algorithm itself as well. This keeps our method generic and therefore applicable to any energy function. A key observation, however, relates to the fact that label pruning is a technique not meant to be used on its own. Its use is allowed only in conjunction with our priority-based message scheduling scheme of

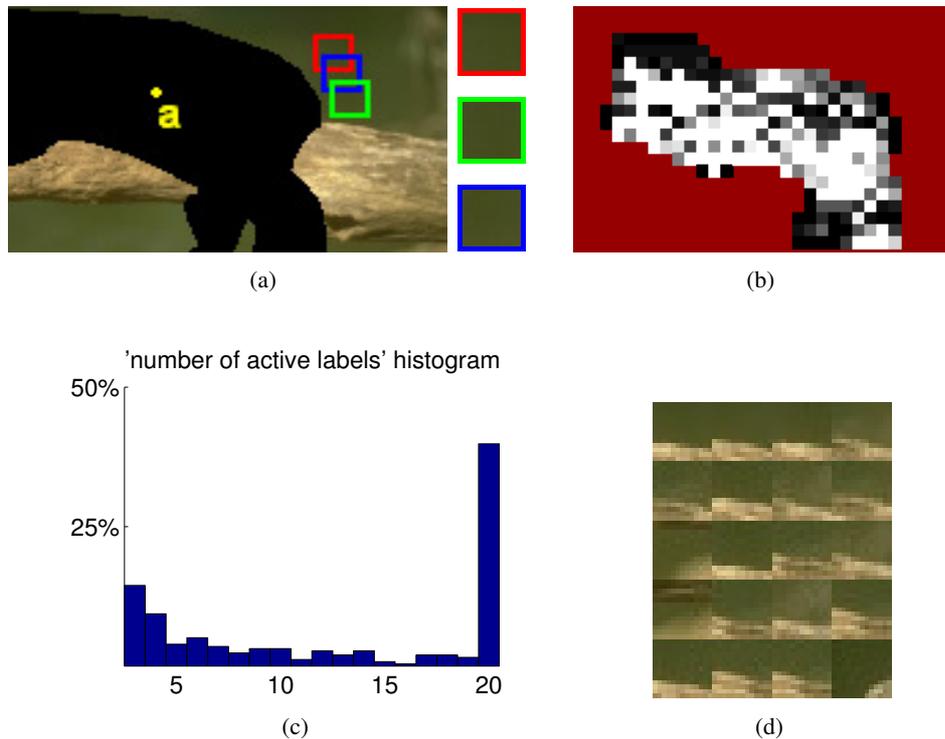


Fig. 13: (a) Although the red, green and blue patches correspond to distinct labels, they are very similar and so only one of them has to be used as an active label for an MRF node. (b) A map with the number of active labels per node (for the 2nd example of Figure 10). Darker patches correspond to nodes with fewer labels. As can be seen, interior nodes often require more labels. (c) The corresponding histogram showing the percentage of nodes that use a certain number (in the range from $L_{min} = 3$ to $L_{max} = 20$) of active labels. (d) The active labels for node a in Fig. (a).

visiting most confident nodes first (*i.e.* nodes for which label pruning is safe and does not throw away useful labels). This is exactly the reason why label pruning does not take place during the backward pass.

In practice, we apply label pruning only to nodes whose number of active labels exceeds a user specified number L_{max} . To this end, when we are about to commit a node, we traverse its labels in order of belief (from high to low) and each such label is declared active until either no more labels with relative belief greater than b_{prune} exist or the maximum number of active labels L_{max} has been reached. In the case of image completion, however, it turns out that we also have to apply an additional filtering procedure as part of label pruning. The problem is that otherwise we may end up having too many active labels which are similar to each other, thus wasting part of the L_{max} labels we are allowed to use. This issue is further illustrated in Figure 13(a). To this end, as we traverse the sorted labels, we declare a label as active only if it is not similar to any of the already active labels (where similarity is measured by calculating the SSD between image patches), otherwise we skip that label and go to the next one. Alternatively, we apply a

clustering procedure to the patches of all labels beforehand (*e.g.* cluster them into textons) and then never use more than one label from each cluster while traversing the sorted labels. Finally, we should note that for all nodes a (user-specified) minimum number of active labels L_{min} is always kept.

The net result of label pruning is thus to obtain a compact and diverse set of active labels for each MRF node (all of them having reasonably good beliefs). *E.g.* Figure 13(b) displays the number of active labels used by each of the nodes in the second example of Figure 10. The darker a patch is in that figure, the fewer are the active labels of the corresponding node. As it was expected, interior nodes often require more active labels to use. The corresponding histogram, showing the percentage of nodes that use a certain number of active labels, is displayed in Figure 13(c). Notice that more than half of the MRF nodes do not use the maximum number of active labels (which was $L_{max} = 20$ in this case). Also, Fig. 13(d) displays the active labels that have been selected by the algorithm for node a in Fig. 13(a).

IV. EXTENSIONS & FURTHER RESULTS

Completion via energy refinement: One advantage of posing image completion as an optimization problem is that one can now refine completion simply by refining the energy function (*i.e.* adding more terms to it). *E.g.* to favor spatial coherence during image completion (*i.e.* fill the target region with large chunks of the source region) one simply needs to add the following “*incoherence penalty terms*” V_{pq}^0 to our energy function: $V_{pq}^0(x_p, x_q) = w_0$ if $x_p - x_q \neq p - q$, while in all other cases $V_{pq}^0(x_p, x_q) = 0$. These terms simply penalize (with a weight w_0) the assignment of non-adjacent patches (with centers x_p, x_q) to adjacent nodes p, q and have proved useful in texture synthesis problems (*e.g.* see Figure 20). Thanks to the ability of Priority-BP to handle effectively any energy function, we intend to explore the utility (with respect to image completion) of many other refinement terms in the future. We believe that this will also be an easy and effective way of applying prior knowledge or imposing user specified constraints on the image completion process.

Pyramid-based image completion: Another advantage of our method is that it can also be used in multi-scale image completion, where a Gaussian pyramid of images $\mathcal{I}_k, \mathcal{I}_{k-1}, \dots, \mathcal{I}_0$ is provided as input. In this case, we begin by applying Priority-BP to the image at the coarsest scale \mathcal{I}_k . The output of this procedure is then up-sampled and the result, say \mathcal{I}'_k , is used for guiding the completion of the image \mathcal{I}_{k-1} at the next finer scale. To this end, the only part of our algorithm that needs to be modified is that of how label costs are computed. In particular, instead of using equation (1), the following formula will



Fig. 14: The result of a multi-scale image completion for a very high resolution input image of size 2816×2112 . In this example, we wanted to remove the price tags from the original image (shown on the left). In this case, a 4-level image pyramid has been used during the completion process.

be applied for computing the label costs at level $k - 1$ of the completion process:

$$V_p(x_p) = \sum_{dp \in [-\frac{w}{2}, \frac{w}{2}] \times [-\frac{h}{2}, \frac{h}{2}]} (\mathcal{I}'_k(p + dp) - \mathcal{I}_{k-1}(x_p + dp))^2. \quad (7)$$

Put another way, the up-sampled image \mathcal{I}'_k from level k is being used as an additional constraint for the completion at level $k - 1$. The rest of the algorithm remains the same and this process is repeated until we reach the image at the finest scale \mathcal{I}_0 . An advantage we thus gain is that features at multiple scales are captured during completion. On the other hand, a disadvantage is that if an error occurs at a coarse scale, this may carry through the finer levels of the completion process. Fig. 14 shows a result produced by our method for a multi-scale completion of a very high resolution input image.

More experimental results: Figures 15, 16 contain further results on image completion. These results along with those in Figure 10 demonstrate the effectiveness of our method, which was tested on a wide variety of input images. As can be seen from the presented examples, Priority-BP was able to handle the completion of smooth regions, textured areas, areas with structure, as well as any combinations of the above. Also, figure 17 contains some examples on texture synthesis. These were again produced by utilizing our exemplar-based framework. For each of these examples, we placed the small input texture into the upper left corner of a large empty area that should contain the output texture, and then let the algorithm fill the rest of that area, thus synthesizing a larger texture. In addition, Figure 18 demonstrates another possible application of Priority-BP. In this case, it has been used for accomplishing the task of removing text from images, whereas, in Figure 19, Priority-BP has been employed as an image inpainting tool for the restoration of a destroyed digital photograph. Our method had no problem of handling these tasks as well. At this point, it is important to emphasize the fact that, in all of the above cases, exactly

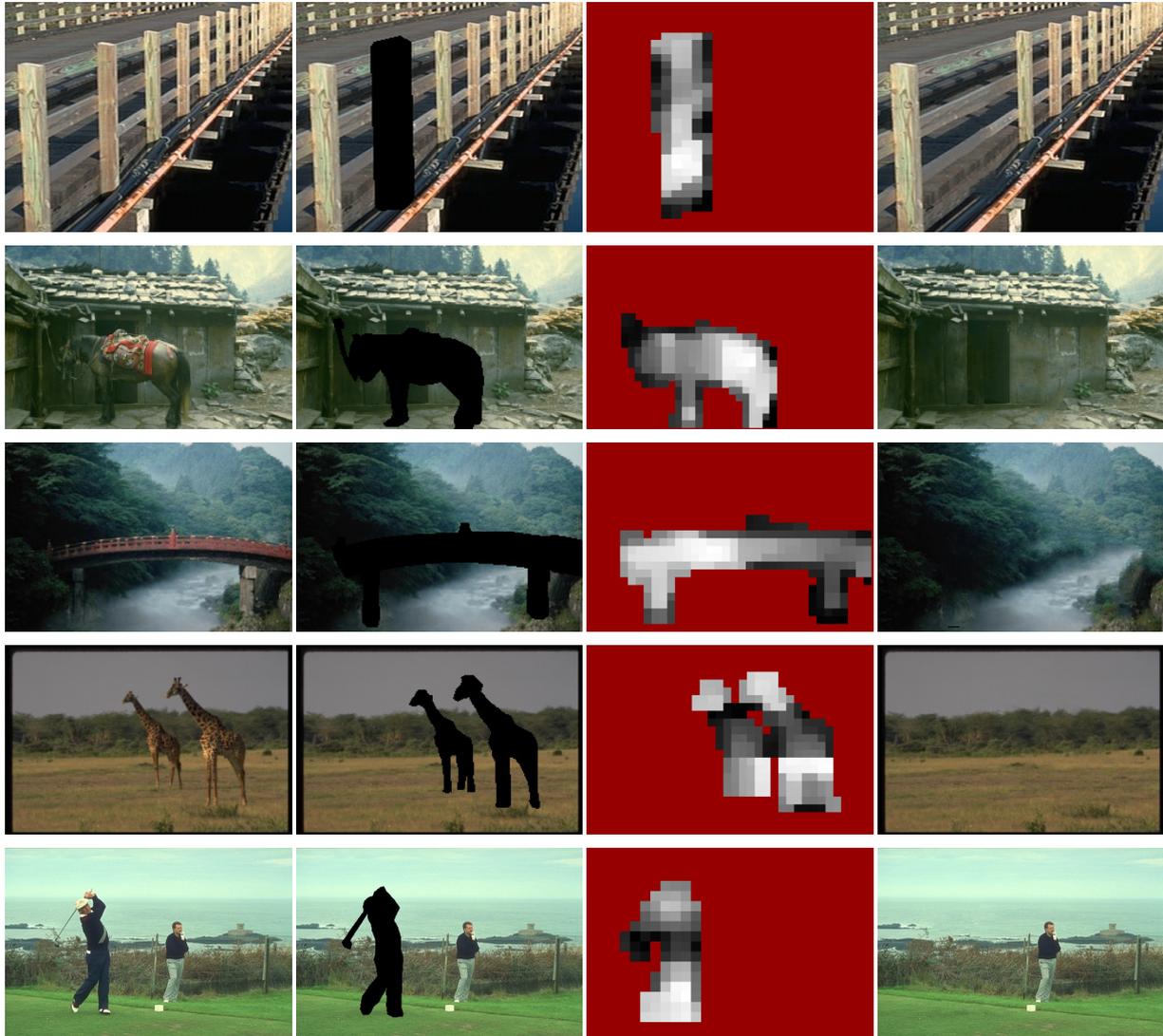


Fig. 15: Image completion results. From left to right: original images, masked images, visiting order at 1st forward pass, Priority-BP results

the same algorithm has been used.

In Figure 20, we demonstrate an example of using the “incoherence penalty terms” in texture synthesis. As one can observe, the output texture does contain large chunks of the input texture as desired. We should also note that, for all of the presented examples (whether they are about image completion, texture synthesis or image inpainting), the visiting order of the nodes during the first forward pass is shown as well. This contributes to illustrating how the algorithm initially chooses to propagate information (*i.e* messages) for each one of the input images. In our tests the patch size ranged between 7×7 and 27×27 . The running time on a 2.4GHz CPU varied from a few seconds up to 2 minutes for $256 \times$



Fig. 16: Some more results on image completion, produced using the Priority-BP algorithm. From left to right: original images, masked images, visiting order at 1st forward pass, Priority-BP results

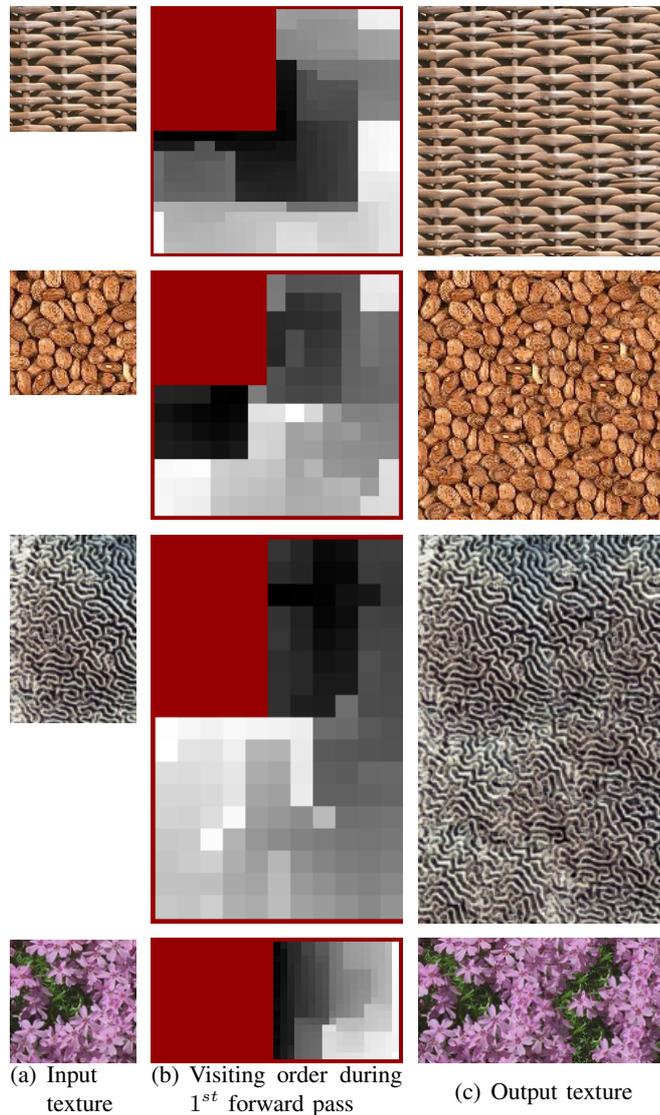


Fig. 17: Texture synthesis results produced with the Priority-BP algorithm

170 images, while the maximum number of labels L_{max} was set between 10 and 50 (depending on the input's difficulty). For all of the examples, the belief thresholds were set equal to $b_{conf} = -SSD_0$ and $b_{prune} = -2 \cdot SSD_0$, where SSD_0 represents a predefined mediocre SSD score between $w \times h$ patches.

Composition of final patches: After the Priority-BP algorithm has converged and the final labels (*i.e.* patches) have been selected for each MRF node, we then need to compose them to produce the final result. This can be achieved in many ways. One simple solution is the following: we visit again the nodes of the MRF in the same order as they were visited during the last iteration of Priority-BP. Then, as we are traversing the nodes, we compose the corresponding final patches by simply blending them with weights

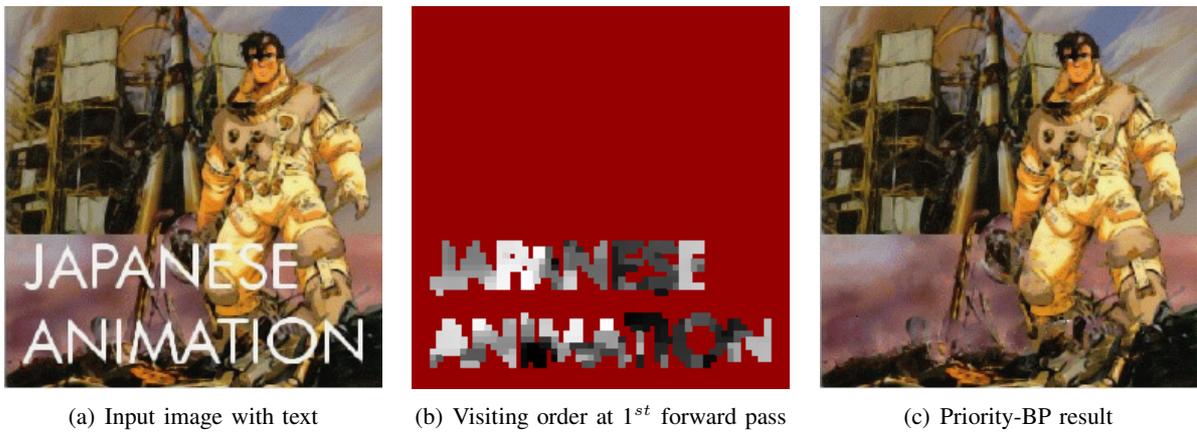


Fig. 18: An example of text removal

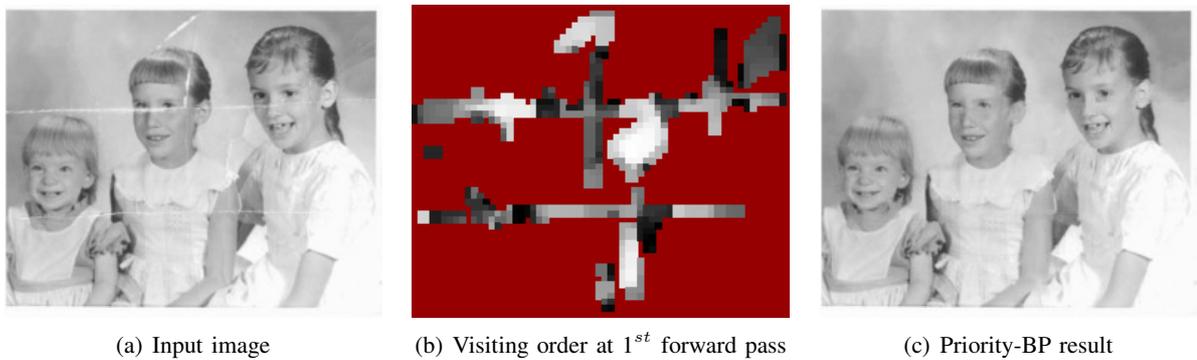


Fig. 19: An image inpainting example

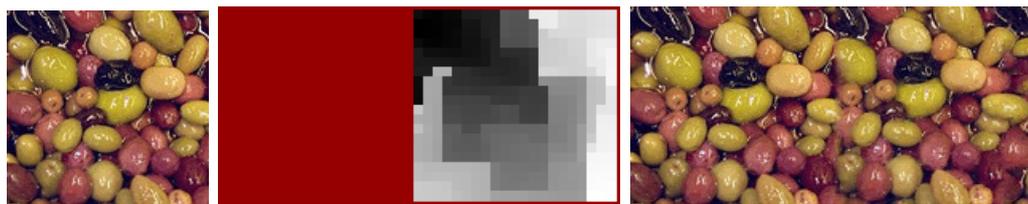


Fig. 20: Texture synthesis using the “incoherence penalty terms”. Notice that, in this case, the output texture has been synthesized by copying large chunks from the input texture.

which are proportional to the nodes’ confidence (as estimated by the Priority-BP algorithm). This simple technique has proved effective in practice and gave good results in most cases.

Another, more elaborate, way for performing the composition is to use an image blending technique (as in [33]), or to apply a composition method similar to the one described in [20], [15]. Both of these methods have been tested as well, and in both cases it was again important that the order by which

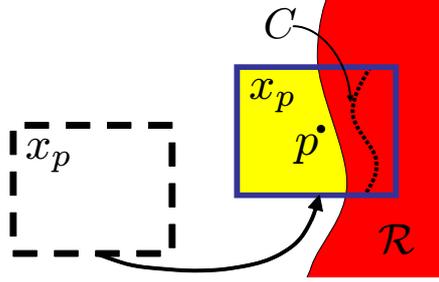


Fig. 21: Only region $\mathcal{R} \subset \mathcal{T}$ has been filled so far by the composition of the final patches. We want to extend \mathcal{R} by seamlessly pasting the next selected patch x_p onto the neighborhood of node p . One way for this is by finding a curve C along which there is a seamless transition between the patch x_p and the region \mathcal{R} . In addition, one may apply a smooth correction to the patch x_p so that it matches seamlessly with the region \mathcal{R} . The smooth correction is obtained by solving a poisson equation over the neighborhood of node p , *i.e.* over the blue rectangle.

the final patches were composed coincided with the visiting order of the nodes during the last iteration of Priority-BP. Let us, for instance, assume that we want to seamlessly compose the patch x_p (*i.e.* the final patch for node p) onto the region \mathcal{R} , which is the region that has been constructed so far by the composition of patches from all previously visited nodes (see Fig. 21). One way to achieve this is by finding an optimal boundary C inside the region $x_p \cap \mathcal{R}$, so that the transition from \mathcal{R} to x_p along that boundary is as seamless as possible. Depending on whether C is an open or closed curve, this problem can be formulated either as a problem of dynamic programming [20], or as a minimum-cut problem [15] respectively. Alternatively, another way for compositing x_p with \mathcal{R} is to apply a smooth correction to patch x_p so that a new patch x'_p is obtained, which seamlessly matches with region \mathcal{R} . The image blending technique in [33] can be used for this purpose, in which case the corrected patch x'_p is obtained merely by solving a Poisson differential equation over the neighborhood region of node p . Other techniques, such as multi-resolution splining [34], have been tried as well, and have also given good results.

Computing the MRF potential functions in the frequency domain: Finally, another point, that is worth mentioning (as it brings a great reduction in the computational time), is the use of the Fast Fourier Transform [35] for performing all the SSD calculations that are needed by the algorithm. More specifically, the estimation of all label costs, as well as all pairwise potentials requires many SSD computations. *E.g.*, as indicated by equation (1), for estimating the label costs for a node p we need to calculate the SSD between a local neighborhood around p , say $\mathcal{I}_0(p + dp)$, and every other source patch, say $\mathcal{I}_0(x_p + dp)$, with the result being multiplied by a mask $\mathcal{M}(p + dp)$, *i.e.*:

$$V_p(x_p) = \sum_{dp} \mathcal{M}(p + dp) (\mathcal{I}_0(p + dp) - \mathcal{I}_0(x_p + dp))^2. \quad (8)$$

By defining, however, the following identities:

$$\begin{aligned} t &\triangleq x_p, \\ \mathcal{I}_1(dp) &\triangleq \mathcal{I}_0(p + dp), \\ \mathcal{M}_1(dp) &\triangleq \mathcal{M}(p + dp), \end{aligned}$$

and substituting them into equation (8), that equation reduces to:

$$\begin{aligned} V_p(t) &= \sum_{dp} \mathcal{M}_1(dp) (\mathcal{I}_1(dp) - \mathcal{I}_0(t + dp))^2 \\ &= \sum_{dp} \mathcal{M}_1(dp) \mathcal{I}_1(dp)^2 - 2 \sum_{dp} \mathcal{M}_1(dp) \mathcal{I}_1(dp) \mathcal{I}_0(t + dp) + \sum_{dp} \mathcal{M}_1(dp) \mathcal{I}_0(t + dp)^2, \end{aligned}$$

which can be also written as an expression involving just correlations between functions, *i.e.*:

$$V_p(t) = \langle \mathcal{M}_1, \mathcal{I}_1^2 \rangle(0) - 2 \langle \mathcal{M}_1 \cdot \mathcal{I}_1, \mathcal{I}_0 \rangle(t) + \langle \mathcal{M}_1, \mathcal{I}_0^2 \rangle(t).$$

In the above expression, the symbol $\langle \cdot, \cdot \rangle$ denotes the correlation operation, which is defined as:

$$\langle f, g \rangle(t) = \sum_x f(x) g(x + t). \quad (9)$$

Furthermore, the first term (*i.e.* $\langle \mathcal{M}_1, \mathcal{I}_1^2 \rangle(0)$) is independent of t , which means that it can be precomputed, and so we can estimate all values of the function $V_p(\cdot)$ by using just two correlation operations. However, the advantage we gain in this manner is that these correlations can now be computed very efficiently, simply by moving to the frequency domain and using the Fast Fourier Transform (FFT) therein [36], [37]. This way of performing the computations greatly accelerates the whole process and is applied for estimating the pairwise potentials $V_{pq}(\cdot, \cdot)$ as well.

V. CONCLUSIONS

A novel approach unifying image completion, texture synthesis and image inpainting has been presented in this paper. In order to prohibit visually inconsistent results, we try to avoid greedy patch assignments, and instead pose all of these tasks as a discrete labeling problem with a well defined global objective function. To solve this problem, a novel optimization scheme, Priority-BP, has been proposed, that carries two very important extensions over standard BP: priority-based message scheduling and dynamic label pruning. This optimization scheme does not rely on any image-specific prior knowledge and can thus be applied to all kinds of images. Furthermore, it is generic (*i.e.* applicable to any MRF energy) and thus

cope with one of the main limitations of BP: its inefficiency to handle problems with a huge number of labels. Also, experimental results on a wide variety of images have verified the effectiveness of our method.

One interesting avenue of future work would be to extend our framework so that it can be used for other types of completion problems as well. *E.g.* it would be interesting to test our framework on problems such as video completion or geometric completion. Also, in the future, we plan to allow the inclusion of more “refinement terms” into our energy function. We believe that this will be a very elegant and easy-to-apply way for allowing the user to impose either high-level or low-level constraints onto the completion process. In this manner, the user could gain a finer control over the final result. Furthermore, this would make our method suitable for problems such as, *e.g.* constrained texture synthesis. Finally, besides image completion, we also plan to test our Priority-BP algorithm, which is a generic MRF optimization scheme, to other labeling problems as well, for which the large cardinality of their state-space causes them to have a very high computational cost.

REFERENCES

- [1] J. Portilla and E. P. Simoncelli, “A parametric texture model based on joint statistics of complex wavelet coefficients.” *IJCV*, vol. 40, no. 1, pp. 49–70, 2000.
- [2] D. J. Heeger and J. R. Bergen, “Pyramid-based texture analysis/synthesis,” in *SIGGRAPH*, 1995, pp. 229–238.
- [3] Y. W. S. Soatto, G. Doretto, “Dynamic textures,” in *Intl. Conf. on Computer Vision*, pp. 439–446.
- [4] A. W. Fitzgibbon, “Stochastic rigidity: Image registration for nowhere-static scenes,” in *ICCV*, 2001, pp. 662–669.
- [5] M. Szmummer and R. W. Picard, “temporal texture modeling,” in *Proc. of Int. Conference on Image Processing*, vol. 3, 1996, pp. 823–826.
- [6] G. Doretto and S. Soatto, “Editable dynamic textures.” in *CVPR (2)*, 2003, pp. 137–142.
- [7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in *Siggraph 2000, Computer Graphics Proceedings*. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 417–424.
- [8] M. Bertalmío, A. L. Bertozzi, and G. Sapiro, “Navier-stokes, fluid dynamics, and image and video inpainting.” in *CVPR (1)*, 2001, pp. 355–362.
- [9] C. Ballester, M. Bertalmío, V. Caselles, G. Sapiro, and J. Verdera, “Filling-in by joint interpolation of vector fields and gray levels.” *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1200–1211, 2001.
- [10] M. Bertalmío, L. A. Vese, G. Sapiro, and S. Osher, “Simultaneous structure and texture image inpainting.” in *CVPR (2)*, 2003, pp. 707–712.
- [11] T. Chan and J. Shen, “Non-texture inpaintings by curvature-driven diffusions,” *J. Visual Comm. Image Rep.*, vol. 12(4), pp. 436–449, 2001.
- [12] A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling.” in *ICCV*, 1999.
- [13] L.-Y. Wei and M. Levoy, “Fast texture synthesis using tree-structured vector quantization,” in *Siggraph 2000, Computer Graphics Proceedings*. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 479–488.

- [14] J. S. D. Bonet, “Multiresolution sampling procedure for analysis and synthesis of texture images,” *Computer Graphics*, vol. 31, no. Annual Conference Series, pp. 361–368, 1997.
- [15] V. Kwatra and *et al.*, “Graphcut textures: Image and video synthesis using graph cuts,” in *SIGGRAPH*, 2003.
- [16] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, “Real-time texture synthesis by patch-based sampling,” *ACM Trans. Graph.*, vol. 20, no. 3, pp. 127–150, 2001.
- [17] Q. Wu and Y. Yu, “Feature matching and deformation for texture synthesis,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 364–367, 2004.
- [18] M. Ashikhmin, “Synthesizing natural textures,” in *Symposium on Interactive 3D Graphics*, 2001, pp. 217–226.
- [19] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *SIGGRAPH 2001, Computer Graphics Proceedings*. ACM Press / ACM SIGGRAPH, 2001, pp. 327–340.
- [20] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *SIGGRAPH 2001, Computer Graphics Proceedings*. ACM Press / ACM SIGGRAPH, 2001, pp. 341–346.
- [21] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa, “Video textures,” in *Siggraph 2000, Computer Graphics Proceedings*. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 489–498.
- [22] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, “Image completion with structure propagation,” in *SIGGRAPH*, 2005.
- [23] I. Drori, D. Cohen-Or, and H. Yeshurun, “Fragment-based image completion,” in *SIGGRAPH*, 2003.
- [24] J. Jia and C.-K. Tang, “Image repairing: Robust image synthesis by adaptive nd tensor voting,” in *CVPR*, 2003.
- [25] Y. Wexler, E. Shechtman, and M. Irani, “Space-time video completion,” in *CVPR (1)*, 2004, pp. 120–127.
- [26] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, “Texture optimization for example-based synthesis,” *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.
- [27] A. Criminisi, P. Pérez, and K. Toyama, “Object removal by exemplar-based inpainting,” in *CVPR*, 2003.
- [28] N. Komodakis and G. Tziritas, “Image completion using global optimization,” in *CVPR*, 2006.
- [29] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, “Learning low-level vision,” *International Journal of Computer Vision*, vol. 40, no. 1, October 2000.
- [30] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient belief propagation for early vision,” in *CVPR*, 2004.
- [31] O. Boiman and M. Irani, “Detecting irregularities in images and in video,” in *International Conference On Computer Vision*, 2005.
- [32] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.
- [33] P. Pérez, M. Gangnet, and A. Blake, “Poisson image editing,” in *SIGGRAPH*, 2003.
- [34] P. J. Burt and E. H. Adelson, “A multiresolution spline with application to image mosaics,” *ACM Trans. Graph.*, vol. 2, no. 4, 1983.
- [35] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.
- [36] S. L. Kilthau, M. S. Drew, and T. Möller, “Full search content independent block matching based on the fast fourier transform,” in *ICIP (1)*, 2002, pp. 669–672.
- [37] C. Soler, M.-P. Cani, and A. Angelidis, “Hierarchical pattern mapping,” in *SIGGRAPH*, 2002, pp. 673–680.