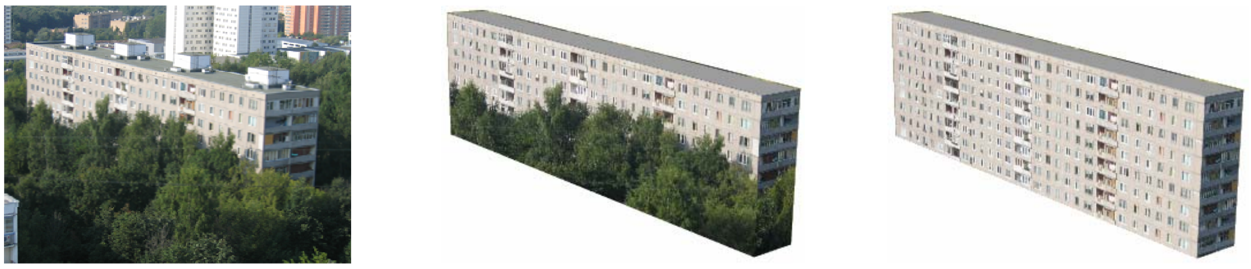
Automatic building texture completion

Vadim Konushin, Vladimir Vezhnevets
Department of Computational Mathematics and Cybernetics, Graphics and Media Lab
Moscow State Lomonosov University, Moscow, Russia
vadim@graphics.cs.msu.ru, vvp@graphics.cs.msu.ru



Original image. Reconstructed model without image completion. Reconstructed model with image completion.

Fig.1. Example of image completion applied to a reconstructed model.

Abstract

In this paper a new image completion algorithm, specifically designed to work with building textures, is proposed. It uses high periodicity, inherent to building textures, to estimate a floor size. The usage of this estimate speeds up image completion process, allowing dealing with high texture resolutions, and helps to capture the macrostructure of the texture. Results and some ideas for further improvement are also provided.

Keywords: *image completion, graph cuts.*

1. INTRODUCTION

Image completion task consists in filling in missing pixels in unknown region of an image in a visually plausible way. One of its important applications is a reconstruction of building textures in building reconstruction tasks. Buildings are often partially occluded by other buildings, road signs, trees, people or other objects. Choosing a right perspective with no occlusion is not always possible. And even when it is possible, a viewing angle can be very small, which leads to a poor texture quality.

This image completion application has some specific properties:

1. As the building's geometry is reconstructed before the texture, we already know correct perspective transformation and therefore can deal only with orthographic views.
2. Large texture resolutions and large missing areas.
3. High image periodicity.

In this paper, we propose an image completion algorithm, specifically designed for building texture reconstruction tasks. It searches for a building's floor height and then uses it in a patch cloning process. This strategy helps to capture the macrostructure of a building and substantially speeds up the total completion time.

Our algorithm is primarily designed for automatic reconstruction systems, like Automatic Photo Popup [4], though it can also be used for other systems.

The rest of the paper is organized as follows: section 2 gives an overview of the related works, section 3 describes proposed algorithm in details, section 4 presents some results of the algorithm, and in section 5 we draw a conclusion and discuss future works.

2. RELATED WORK

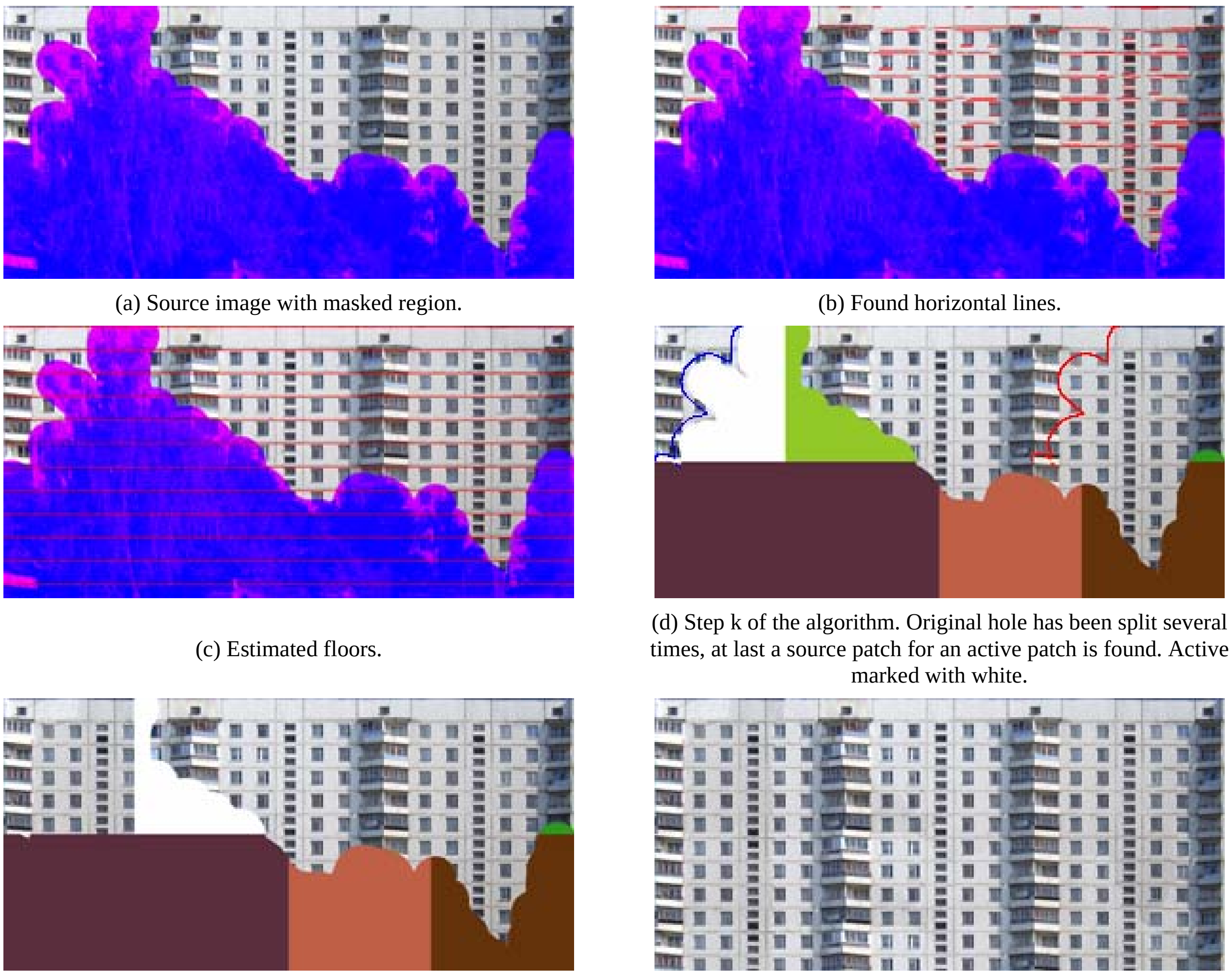
Here we consider only exemplar-based methods, as they are best suited for our task. Inpainting techniques work only with small missing areas, like scratches, and when applied to large holes, produce blurry results, that lack texture.

Drori *et. al.* [3] use pyramid image approximation for patch cloning guidance, adaptive patches sizes and a very large search space: patches can be translated, scaled, flipped or rotated. Results a very impressive, but the computation time is prohibitive even for small size images.

Criminisi *et. al.* [2] use a special patch filling order to encourage isophotes propagation inward a hole. This enables to reconstruct simple linear structures.

In Sun *et. al.* [7] additional user input is used to extend some curves or line segments inward an unknown area. At first texture along these lines is reconstructed using Belief Propagation. Remaining unknown regions are filled by a common patch cloning technique.

In Shen *et. al.* [6] patch based filling algorithm is used in a gradient domain. At first gradient map in an unknown region is reconstructed, and after that image itself is reconstructed by solving a Poisson equation.



(a) Source image with masked region.

(b) Found horizontal lines.

(c) Estimated floors.

(d) Step k of the algorithm. Original hole has been split several times, at last a source patch for an active patch is found. Active marked with white.

(e) Step k + 1 of the algorithm. Found source patch has been cloned to the destination patch. New destination patch is found (again marked with white).

(f) Resulting image.

Fig. 2. Algorithm's stages.

All these algorithms are able to capture only microstructure, as the patches sizes are generally small. In addition, their processing time is too large for typical size of occluded regions in building textures.

In contrast, Hole Filling through Photomontage [8], tries to capture the macrostructure of an image by searching for one large patch to clone into the hole. But the algorithm is still rather slow, and the authors haven't provided any solutions in case, when there is not enough known texture for a patch, that can cover the hole. Our work is based on this algorithm and alleviates these problems.

In real urban modeling projects, that we are aware of, mostly only general-purpose image editors, like Adobe Photoshop [1], are used. Image completion is performed either by tools like a cloning brush, which require a long, meticulous work, or by manually cloning many times a simple patch over the whole texture, which results in synthetic-looking textures. Such projects can also benefit from our algorithm, though its main application is for automatic reconstruction programs, like Automatic Photo Popup [4].

3. OUR APPROACH

3.1 Floor height estimation

At first, we estimate a floor height of the building. This process consists of 3 steps:

- Line detection
- Horizontal lines extraction
- Ransac-based floor height estimation

For a line detection algorithm we use the one, described in [Kosecka]. It applies Canny edge detector, then quantizes edge pixels into several bins by their gradient directions, after which a connected component algorithm is applied to group edge pixels with the same label. Groups, which are smaller than a certain threshold, are rejected. For other groups line parameters are determined, and the ones with quality of the line fit below another threshold are also rejected.

Horizontal lines extraction stage keeps lines, whose slope angle is close to a horizontal. An example of found lines is shown in Fig. 2. b).

To estimate a floor height from these horizontal lines, we use the RANSAC-based algorithm: for a set number of iterations, we choose 2 random lines, compute a vertical translation between them, and shift all horizontal lines by this distance. Then a number of intersected lines between the original lines and the shifted ones is counted. A translation, that gives the largest intersection number, is taken as floor height estimation. This algorithm is presented in the following pseudo code:

Code 1.

```
$MaxNum = 0$;
for $i = 1 : IterNum$
  $L_1 = Lines[rand(1..LinesNum)]$;
  $L_2 = Lines[rand(1..LinesNum)]$;
  if $y(L_1) == y(L_2)$ continue; end if;
  $ShiftVal = y(L_1) - y(L_2)$;
  $ShiftedLines = shift(Lines, ShiftVal)$;
  $Count = \mathrm{intersect}(Lines, ShiftedLines)$;
  if $Count > MaxNum$
    $FloorHeight = abs(ShiftVal)$;
    $MaxNum = Count$;
  end if;
end;
```

Results of floor height estimation can be seen in Fig. 2. c).

3.2 Cloning process

At this step, we try to fill the unknown area with as large patches, as possible. This strategy helps to keep the macrostructure of the image.

Let I be the whole original image, R be an unknown area, P be a current destination patch and S be a corresponding source patch. The cloning process is summarized in Code 2. In short, we are trying to find a source patch to cover the whole unknown area at once. If we find it, then we clone it into the hole, otherwise we split the hole in two, and do the same with each of them.

3.2.1 Find a source patch (P)

We take the same approach, as in [8]. Let I_d be a distance image for the patch P. We define its band mask J as following:

$$J_{i,j} = \begin{cases} 0 & if \quad I_d(i,j) > h \vee (i,j) \in R \\ 1 & otherwise \end{cases}$$

where h - is a predefined width.

Code 2.

```
$P = R$;
while $P \neq 0$
  $S = FindASourcePatch(P)$;
  if $S \neq 0$
    $Clone(S, P)$;
  else
    $Split(P)$;
  end if
  $P = FindActivePatch()$;
end while
```

A translation (x_0, y_0) to a source patch is searched by minimizing the following sum-of-squared-differences (SSD):

$$SSD(x_0, y_0) = \sum_{(x,y)\in I} J(x,y)(I(x,y) - I(x + x_0, y + y_0))^2$$ All shifted points $(x + x_0, y + y_0)$ must come from known area $I \setminus R$. As candidate vertical translations y_0 only those, that are divisible by the floor height, are considered.

3.2.2 Clone (S, P)

Cloning is performed into an extended target region, which is computed by a graph cut algorithm, again as in [8]. We use only a smoothing term in the minimized energy (formula (7) of [8]), as we are only interested in absence of visible seams, but the seams are not required to be close to the not extended target region.

3.2.3 Split (P)

If the current destination patch is too large, and there is no source patch, large enough to cover it, we split this destination patch in two. If a width of the destination patch is greater, than its height, we split it with a vertical line passing through its middle; otherwise we split it with a horizontal line.

3.2.4 Find active patch ()

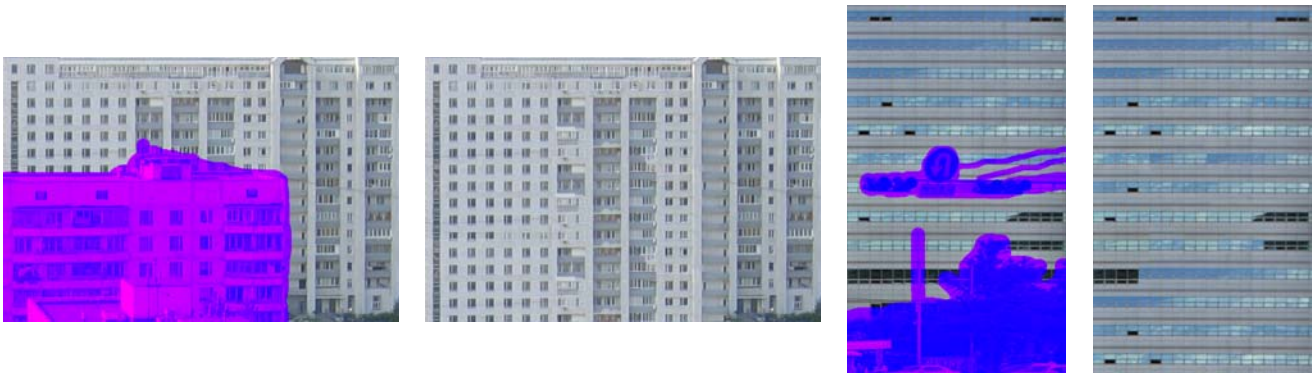
As a next active destination patch we take a patch with a maximum number of known neighbors, so that SSD minimization will be based on enough number of pixels.

Main stages of proposed algorithms are depicted in Fig. 2.

4. IMPLEMENTATION AND RESULTS

Some results are shown in Fig. 1, 2. More results can be seen in Fig. 3.

We have tested our algorithm on more, than 30 building of Moscow and Seoul. Most apartment buildings of Moscow had high vertical repetitiveness and low horizontal one due to alternating windows and balconies. So in our current implementation we changed a source patch search algorithm in order to encourage pure vertical translation. Specifically, if there



(a) Source image 1 with masked region. (b) Resulting image 1. (c) Source image 2 with masked region. (d) Resulting image 2.

Fig. 3. Some results of the proposed algorithm.

is known texture above or below current destination patch of at least one floor height, we prohibit horizontal cloning, and split the destination patch, until we can clone to it a source patch with no horizontal translation.

Besides significant algorithm's speed increase (average floor height in our tests was 30-40 pixels), explicit use of floor height preserves periodical structure of building textures even in cases of SSD minimization errors, so that final texture is plausibly looking (see Fig 3 b.)

We had no other algorithms implementations, and we couldn't compare our method on pictures, presented in other papers, as the proposed algorithm works only with building textures (in orthographic view). So we used computational statistics, presented in Table 1 of [6] with the assumption of linear time dependency on inpainting area size (real dependency is higher and there is also a time dependency on image size). Our comparison results are presented in Table 1. We couldn't compare our time with the time of [8], which is supposed to be faster than other compared algorithms, as the authors didn't present any processing time.

Examples	Fig 3 a), b)	Fig 3 c), d)
Image Size	522 x 875	1000 x 667
Inpainting area	162482	286674
Time of [2]	8,72 min	15,38 min
Time of [3]	8,59 hr	15,16 min
Time of [6]	9,21 min	16,25 min
Time of ours	21 sec	55 sec

Table 1. Algorithms time comparison.

5. SUMMARY AND FUTURE WORK

In this paper we have presented a new image completion algorithm, designed to work with building textures. It estimates a floor height and uses it to guide a patch cloning process. Our algorithm is able to keep a macrostructure of the building and deals with large resolution textures.

In future we plan a more sophisticated building structure analysis, including automatic detection of windows and other structural elements. This additional information can enable a correct reconstruction of occluded building texture in more complex cases and may further increase computation time.

6. REFERENCES

[1] Adobe System Incorp. Adobe Photoshop User Guide, 2002

[2] Criminisi, A., Perez, P., and Toyama, K. Object removal by exemplar-based inpainting. In *Proc. Conf. Comp. Vision Pattern Rec*., *pp. 417–424*, 2003

[3] Drori, I., Cohen-Or, D., and Yeshurun, H. 2003. Fragment-based image completion. In *Proceedings of ACM SIGGRAPH*, *pp. 303–312*, 2003

[4] Hoiem D., Efros A.A., and Hebert M., Automatic Photo Pop-up. In *ACM SIGGRAPH, pp. 577 – 584*, 2005

[5] Kosecka J. and Zhang W.. Video compass. In *European Conference on Computer Vision, pp. 657 – 673*, 2002.

[6] Shen J., Jin X., Zhou C. Gradient Based Image Completion by Solving Poisson Equation. In *PCM (1), pp. 257-268*, 2005

[7] Sun J, Yuan L, Jia J, Shum H.-Y. Image completion with structure propagation. In *ACM Transactions on Graphics 24(3): pp. 861-868*, 2005

[8] Wilczkowiak M., Brostow G., Tordoff B. and Cipolla R..Hole Filling Through Photomontage. In *the proceedings of the 16th British Machine Vision Conference*, *pp. 492-501*, 2005

About the authors

Vadim Konushin is a 5th year student at Moscow State University, Department of Computational Mathematics and Cybernetics, Graphics and Media Lab.

His contact email is vadim@graphics.cs.msu.ru.

Vladimir Vezhnevets is a Ph.D. at Moscow State University, Department of Computational Mathematics and Cybernetics, Graphics and Media Lab.

His contact email is vvp@graphics.cs.msu.ru.