

CS 180 Image Quilting

Project Overview

This project implements the image quilting algorithm for texture synthesis and transfer, as described in the SIGGRAPH 2001 paper by Efros and Freeman. The key idea is to generate textures by sampling overlapping patches from a given sample texture. This process is extended to texture transfer, where the texture of a sample is applied to a target image.

Randomly Sampled Texture

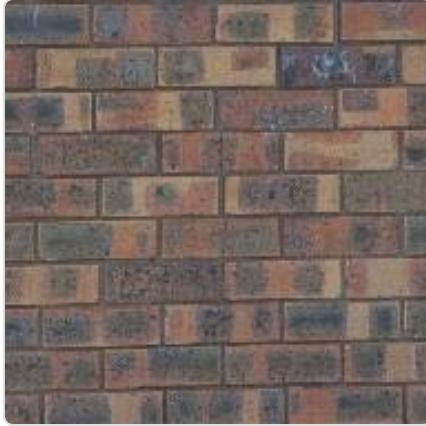
The `quilt_random` function generates an output image by randomly sampling patches of size `patch_size` from a given sample image. Each patch is placed without any overlap, leading to visible seams. This approach serves as a baseline for more advanced methods.



Grass



Grass Randomly Sampled



Brick



Brick Randomly Sampled

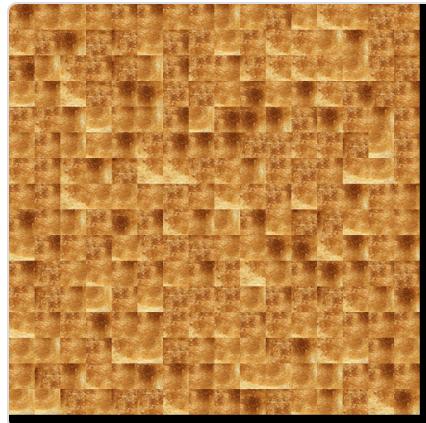
ut it becomes harder to lau-
ound itself, at "this daily
wing rooms," as House De-
scribed it last fall. He fail-
ut he left a ringing question:
ore years of Monica Lewi-
inda Tripp?" That now seem-
Political comedian Al Frac-
ext phase of the story will

Text

Text Randomly Sampled



Toast



Toast Randomly Sampled

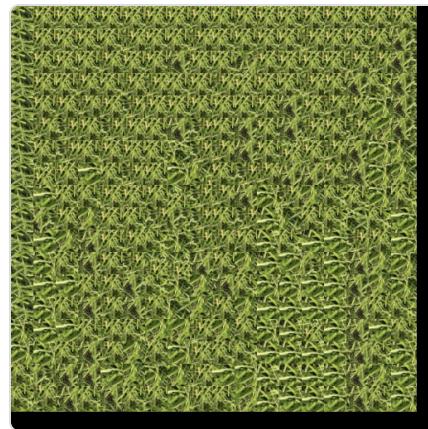
Overlapping Patches

The `quilt_simple` function improves upon the random sampling method by overlapping patches.

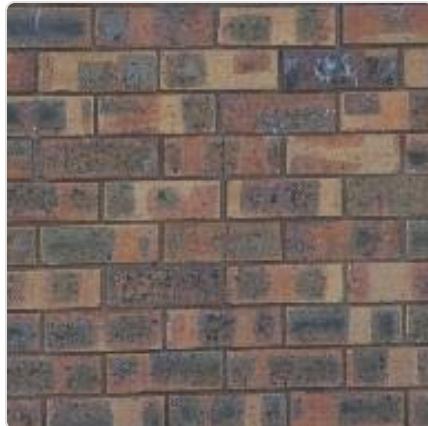
Using the `ssd_patch` and `choose_sample` helper functions, the algorithm ensures better alignment between adjacent patches, reducing visible seams.



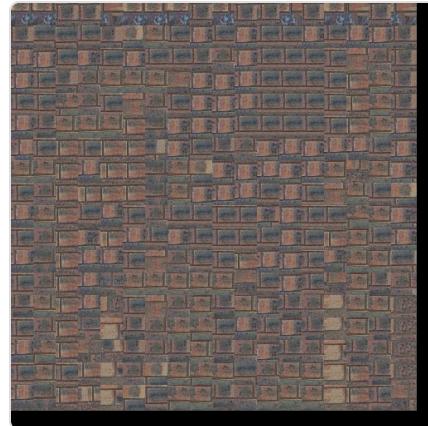
Grass



Grass, Overlapping Patches



Brick



Brick, Overlapping Patches

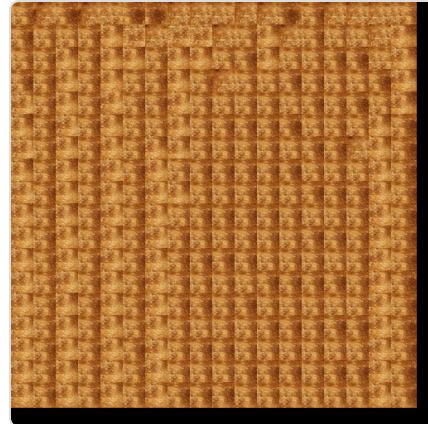
ut it becomes harder to laud sound itself, at "this daily ring rooms," as House described it last fall. He failed he left a ringing question more years of Monica Lewinsky-Tripp?" That now seems Political comedian Al Franken's next phase of the story will

Text

Text, Overlapping Patches



Toast



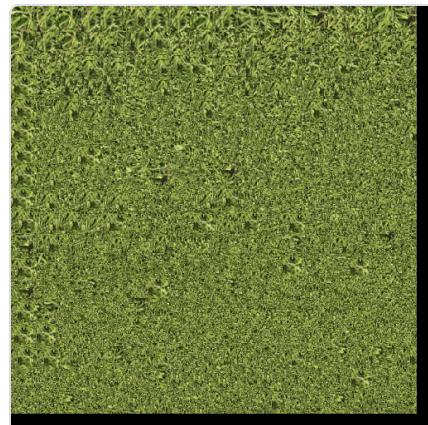
Toast, Overlapping Patches

Seam Finding

The `quilt_cut` function further refines texture synthesis by incorporating seam finding. This minimizes edge artifacts by finding the minimum-cost path through overlapping regions using the `cut` function. The seams are blended to achieve smoother transitions.



Grass



Grass, Seam Finding



Brick



Brick, Seam Finding

ut it becomes harder to lau-
sound itself, at "this daily i-
ving rooms," as House De-
scribed it last fall. He fail-
ut he left a ringing question
more years of Monica Lewin-
inda Tripp?" That now seem
Political comedian Al Frat-
ext phase of the story will

Text

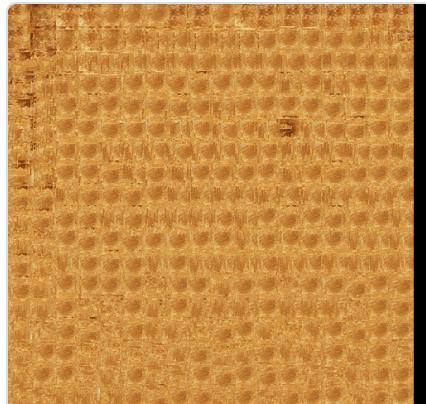
as becomes it? The question is, "How do you find a seam in a noisy image like this?"
It's "Brick" in the original image, so it's hard to tell if there's a seam or not. Let's start with the raw image.
The image is a photograph of a brick wall. The bricks are of different colors, including red, brown, and grey. There is a visible seam between the two rows of bricks.
To find the seam, we can use a seam finding algorithm. One such algorithm is the Beltrami operator, which is a partial differential operator that is used to detect edges and boundaries in images.
The Beltrami operator is defined as follows:
$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

where u is the image intensity and ∇^2 is the Beltrami operator.
We can apply the Beltrami operator to the image to find the edges. The edges are then used to find the seam.
The result of the seam finding algorithm is a red border around the seam in the image.
The final result is a photograph of a brick wall with a red border around the seam.

Text, Seam Finding



Toast

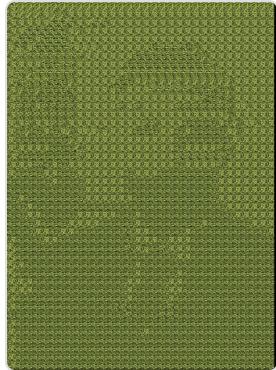


Toast, Seam Finding

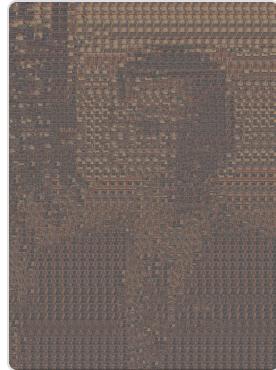
Texture Transfer

The `texture_transfer` function applies texture synthesis to a target image. By balancing overlap costs

and target alignment costs, the algorithm overlays the texture while preserving the target's structural features.



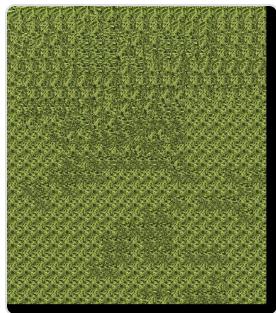
Grass Texture on
Obama



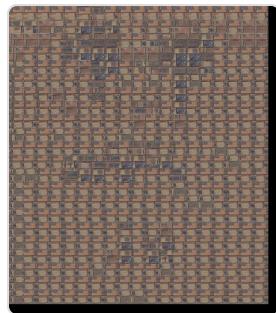
Brick Texture on
Obama



White Texture on
Obama



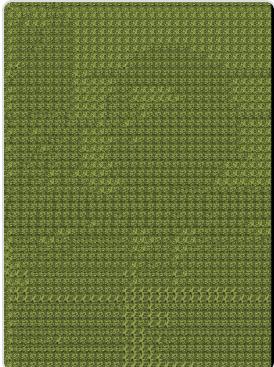
Grass Texture on
Mickey Mouse



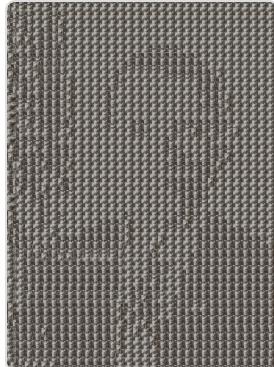
Brick Texture on
Mickey Mouse

Bells and Whistles: Iterative Texture Transfer

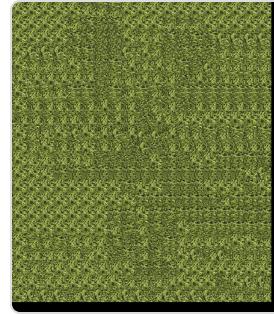
For the B&W portion of the project, I implemented the `texture_transfer_iterative` function. This method progressively refines the transfer result over multiple iterations by iteratively increasing the weight of the target image alignment term. This ensures a more accurate texture alignment with the target structure while retaining the visual characteristics of the texture sample.



Grass Texture on
Obama, Iteratively



White Texture on
Obama, Iteratively



Grass Texture on
Mickey Mouse,
Iteratively

Improvements

Some of the challenges faced during the implementation included:

- Balancing the overlap cost (`alpha`) and target alignment cost.
- Implementing efficient seam finding to avoid misaligned patches, since I kept running into misalignment-related errors.

Future improvements could involve advanced blending techniques (e.g., Laplacian pyramids) and extending the algorithm to handle arbitrary shapes for image completion tasks.

Acknowledgements

This project is a course project for CS 180 at UC Berkeley. Part of the starter code is provided by course staff. This website template is referenced from Bill Zheng.