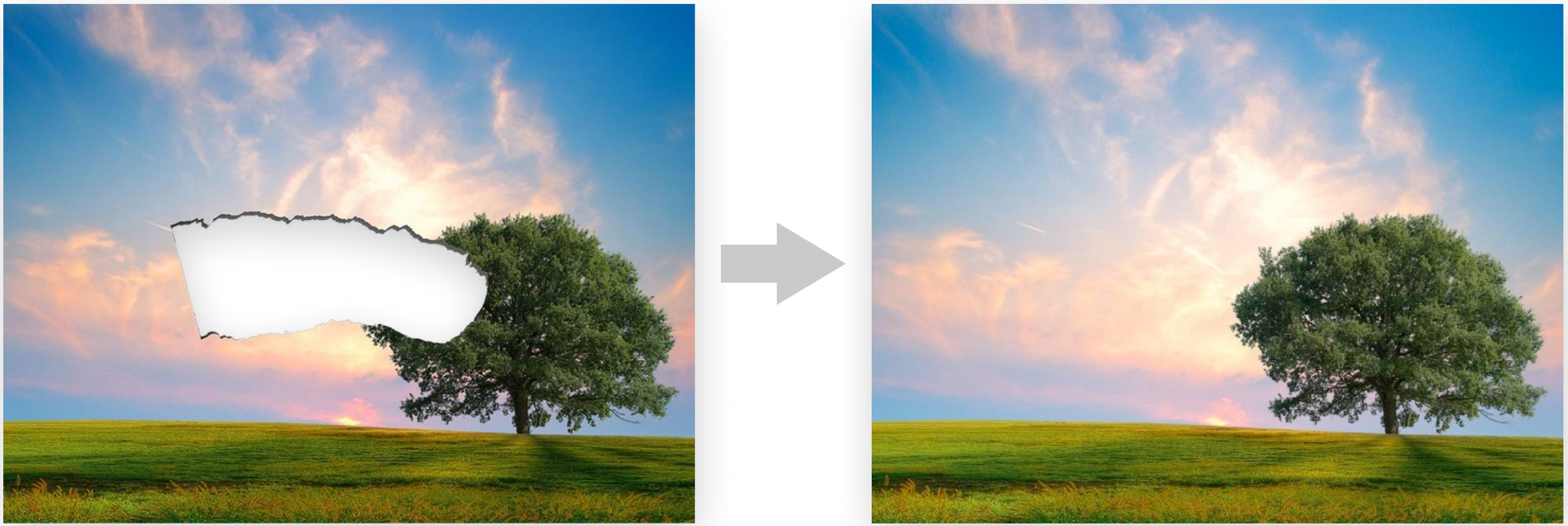
IMAGE COMPLETION

deceivingly simple, impressive results

WHAT IS IT?

- fill in a missing part of an image



not an actual result

MANY NAMES

- image completion
- in-painting
- object removal
- image restoration
- filling-in
- error concealment
- image extrapolation

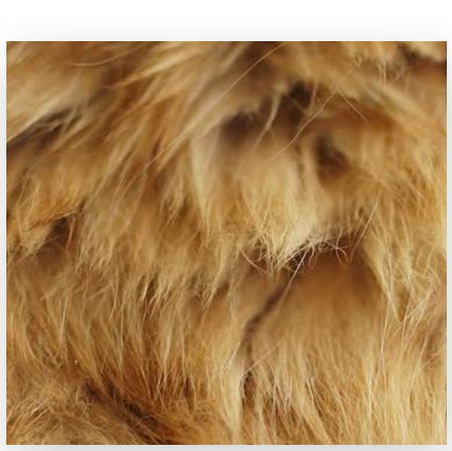
- we'll see why in later examples

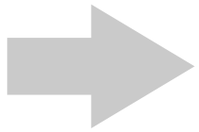
TEXTURE SYNTHESIS

a detour?

WHAT IS IT?

- produce a similar texture starting from a sample





not an actual result

IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

Texture Synthesis by Non-parametric Sampling

Alexei A. Efros and Thomas K. Leung
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776, U.S.A.
{efros,leungt}@cs.berkeley.edu

Abstract

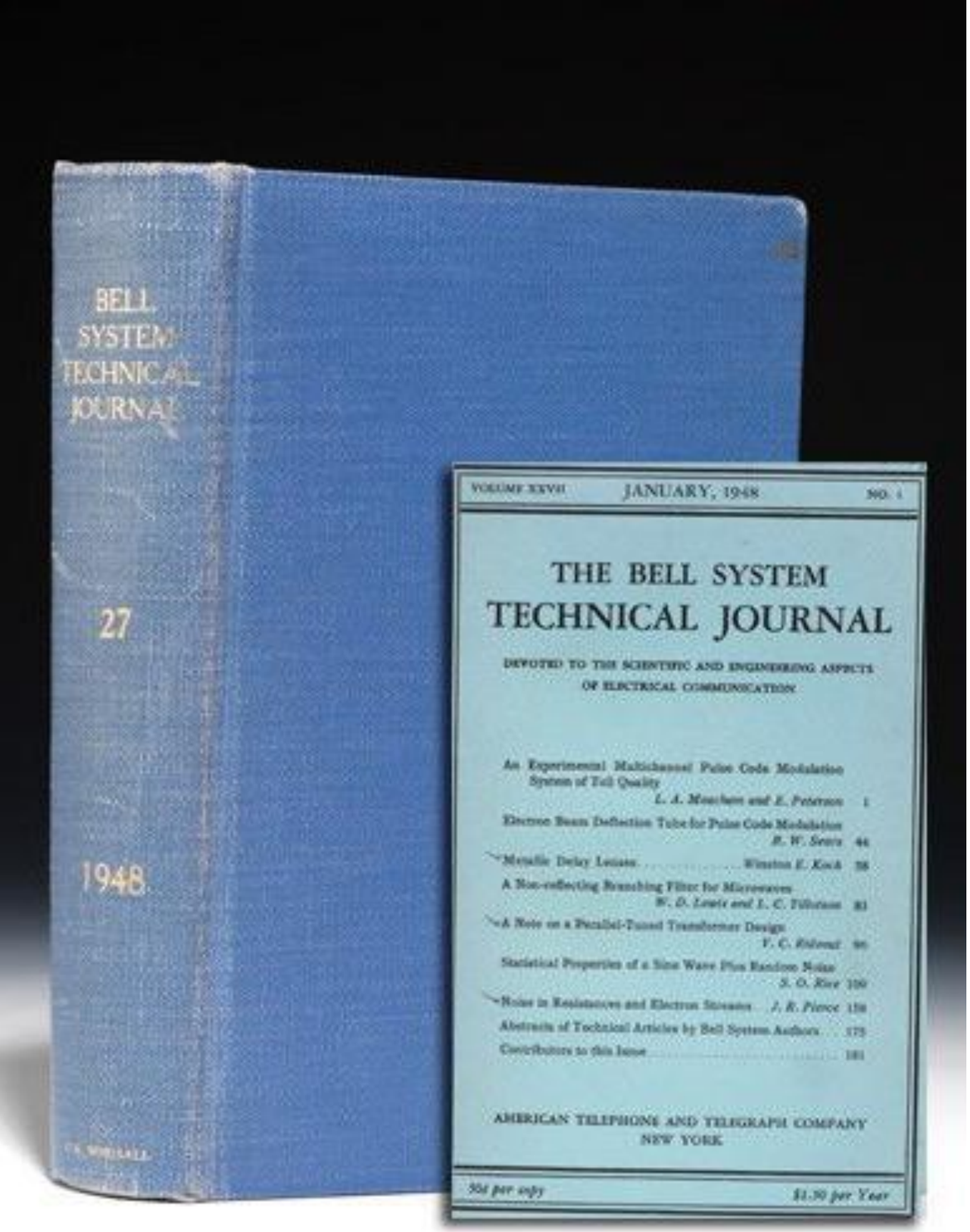
A non-parametric method for texture synthesis is proposed. The texture synthesis process grows a new image outward from an initial seed, one pixel at a time. A Markov random field model is assumed, and the conditional distribution of a pixel given all its neighbors synthesized so far is estimated by querying the sample image and finding all similar neighborhoods. The degree of randomness is controlled by a single perceptually intuitive parameter. The method aims at preserving as much local structure as possible and produces good results for a wide variety of synthetic and real-world textures.

1. Introduction

Texture synthesis has been an active research topic in computer vision both as a way to verify texture analysis methods, as well as in its own right. Potential applications of spatial locality. The result is a very simple texture synthesis algorithm that works well on a wide range of textures and is especially well-suited for constrained synthesis problems (hole-filling).

1.1. Previous work

Most recent approaches have posed texture synthesis in a statistical setting as a problem of sampling from a probability distribution. Zhu et. al. [12] model texture as a Markov Random Field and use Gibbs sampling for synthesis. Unfortunately, Gibbs sampling is notoriously slow and in fact it is not possible to assess when it has converged. Heeger and Bergen [6] try to coerce a random noise image into a texture sample by matching the filter response histograms at different spatial scales. While this technique works well on highly stochastic textures, the histograms are not pow-



INSPIRATION

- *A Mathematical Theory of Communication – Shannon 1948*
- build n-grams probabilities table from a book
- eg:
  - ***go** is often followed by **home***
  - ***how**, **are**, **you** are frequently encountered together*
- sample repeatedly to create sentences

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	**53**	-	14	-	-

will was followed by **bake** **53** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	**14**	-	-

will was followed by **bake 53** times

will was followed by **go 14** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
-

will was followed by **bake 17** times

will was followed by **go 64** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
-

will was followed by **bake 17** times

will was followed by **go 64** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
- ***bake***

will was followed by **bake 17** times

will was followed by **go 64** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
- ***bake***
-

will was followed by **bake 17** times

will was followed by **go 64** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
- ***bake***
- ***cake***

will was followed by **bake 17** times

will was followed by **go 64** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
- ***bake***
- ***cake***

will was followed by **bake 17** times

will was followed by **go 64** times

GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

- finish this:
  *Today I **will** ...*
- ***bake***
- ***cake***
- ***home***

will was followed by **bake 17** times

will was followed by **go 64** times

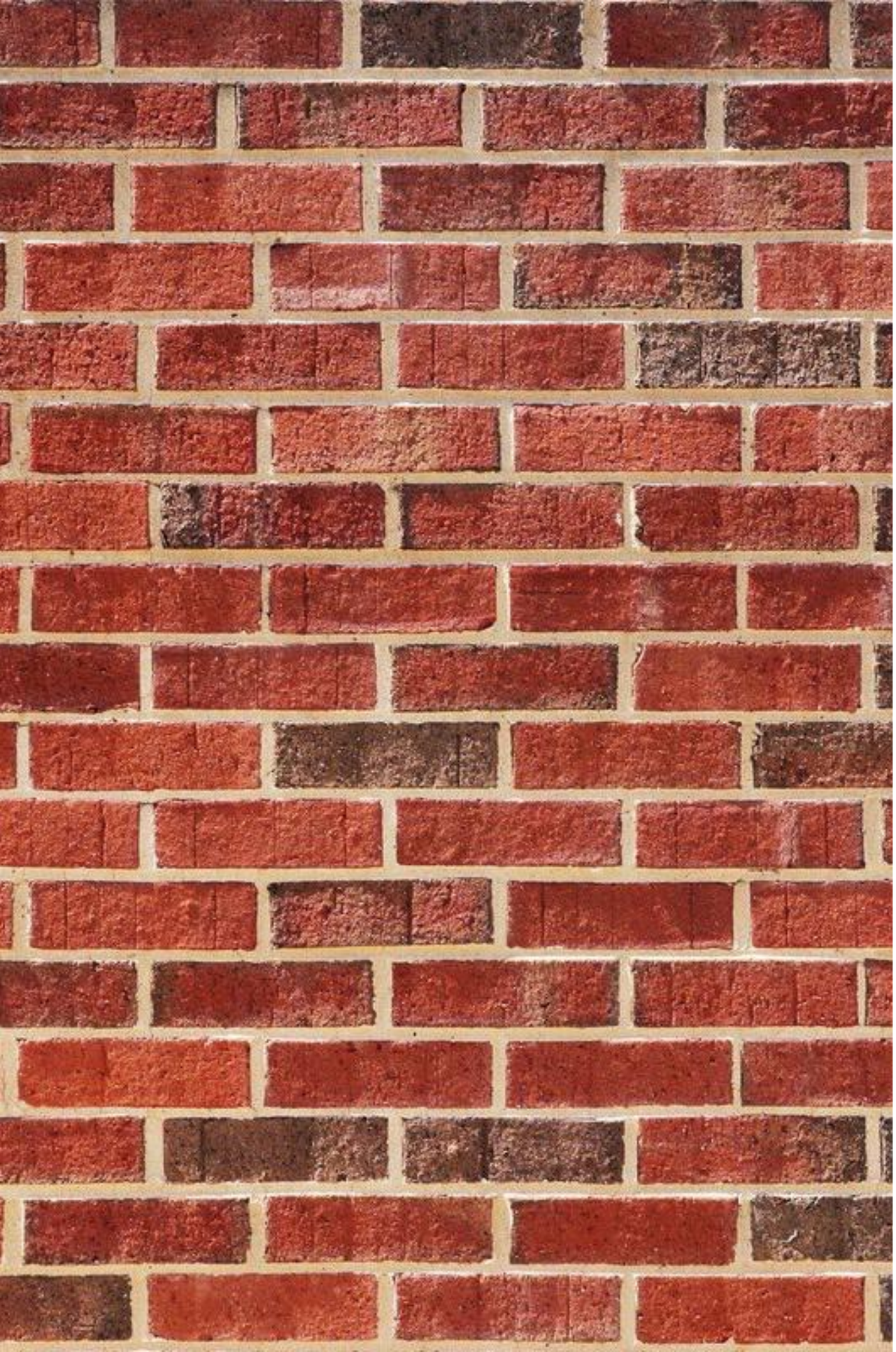
GENERATE SENTENCES

	bake	cake	go	home	will
bake	-	71	-	-	-
cake	-	-	-	12	-
go	6	-	3	49	-
home	7	24	-	-	-
will	53	-	14	-	-

will was followed by **bake 17** times

will was followed by **go 64** times

- finish this:
  *Today I **will** ...*
- ***bake***
- ***cake***
- ***home***
- *Today I will bake a cake at home.*
- alternative:
  Today I will go home.

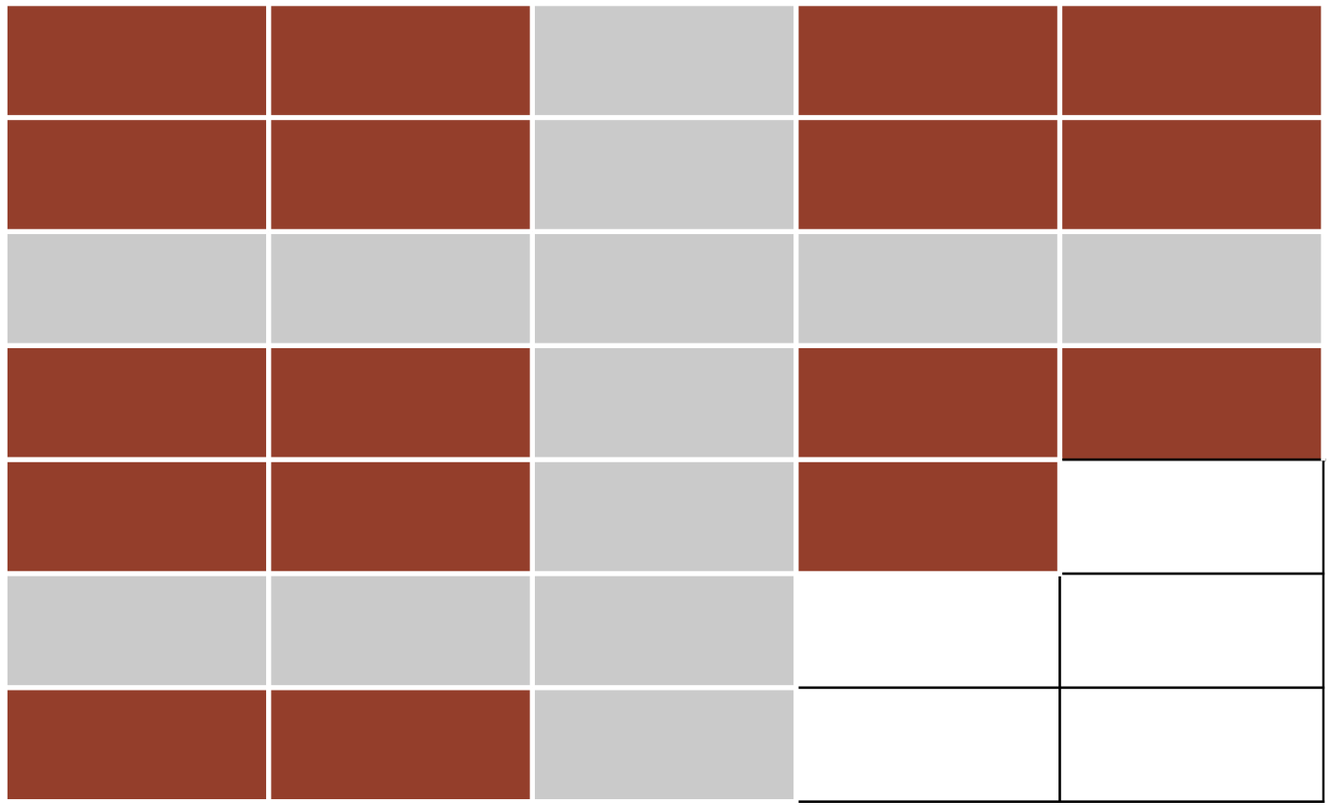


CORE IDEA

- generalize to 2D
  - unit: letter ~ pixel
  - context: n-gram ~ texel
- eg:
  - **red** *often follows* **red**
  - **grey** *seldom after* **red**
- find missing pixel color in probability table, based on neighbors

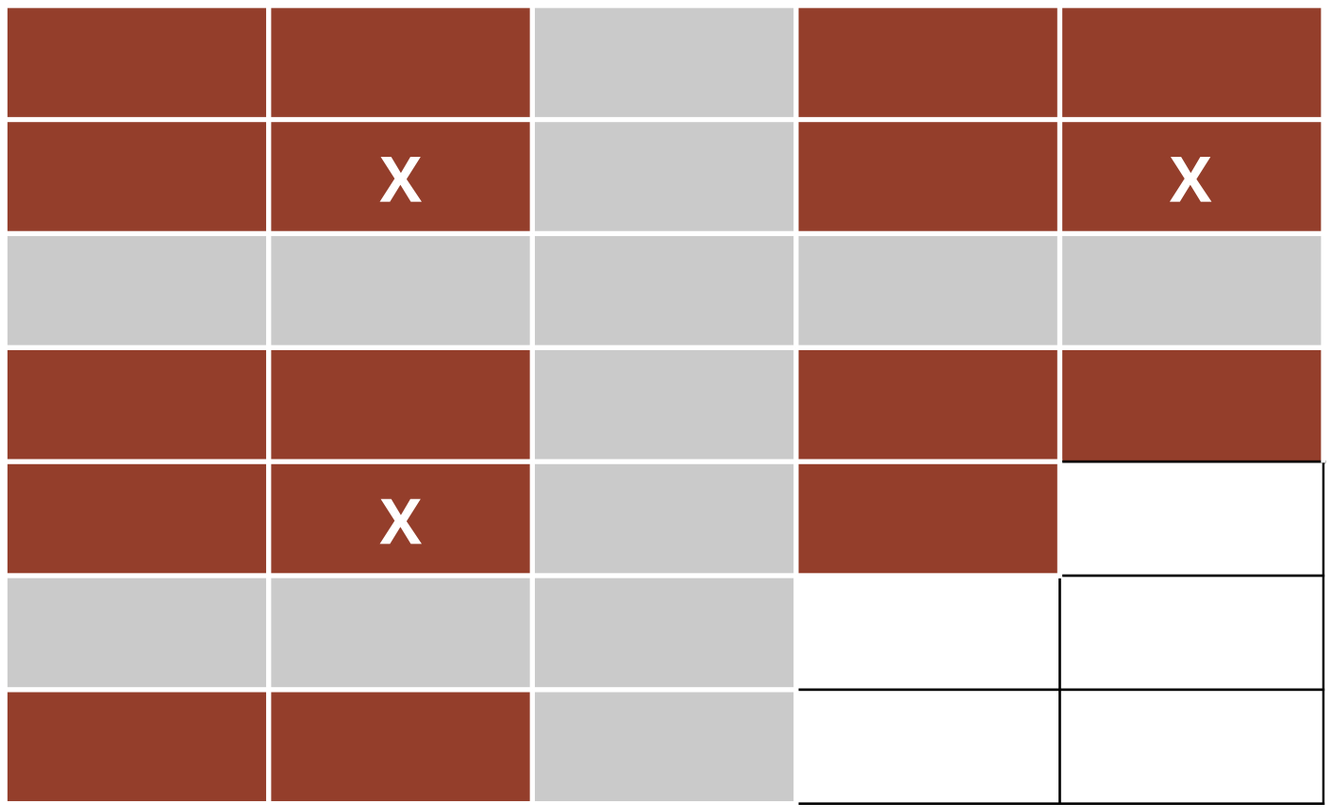
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



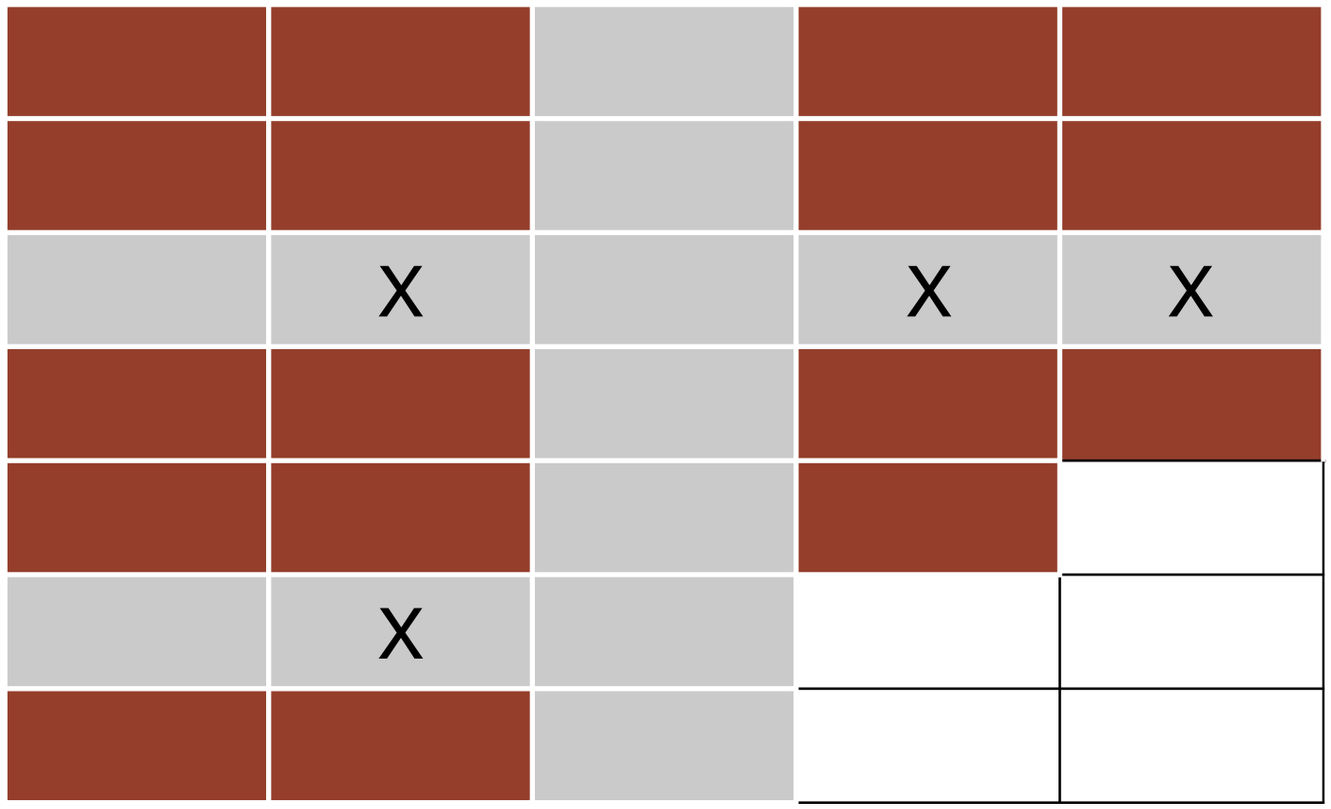
CONTINUE THE TEXTURE

	red	grey
left red, above red	3	0
left red, above grey	3	4
left grey, above red	2	4
left grey, above grey	1	2



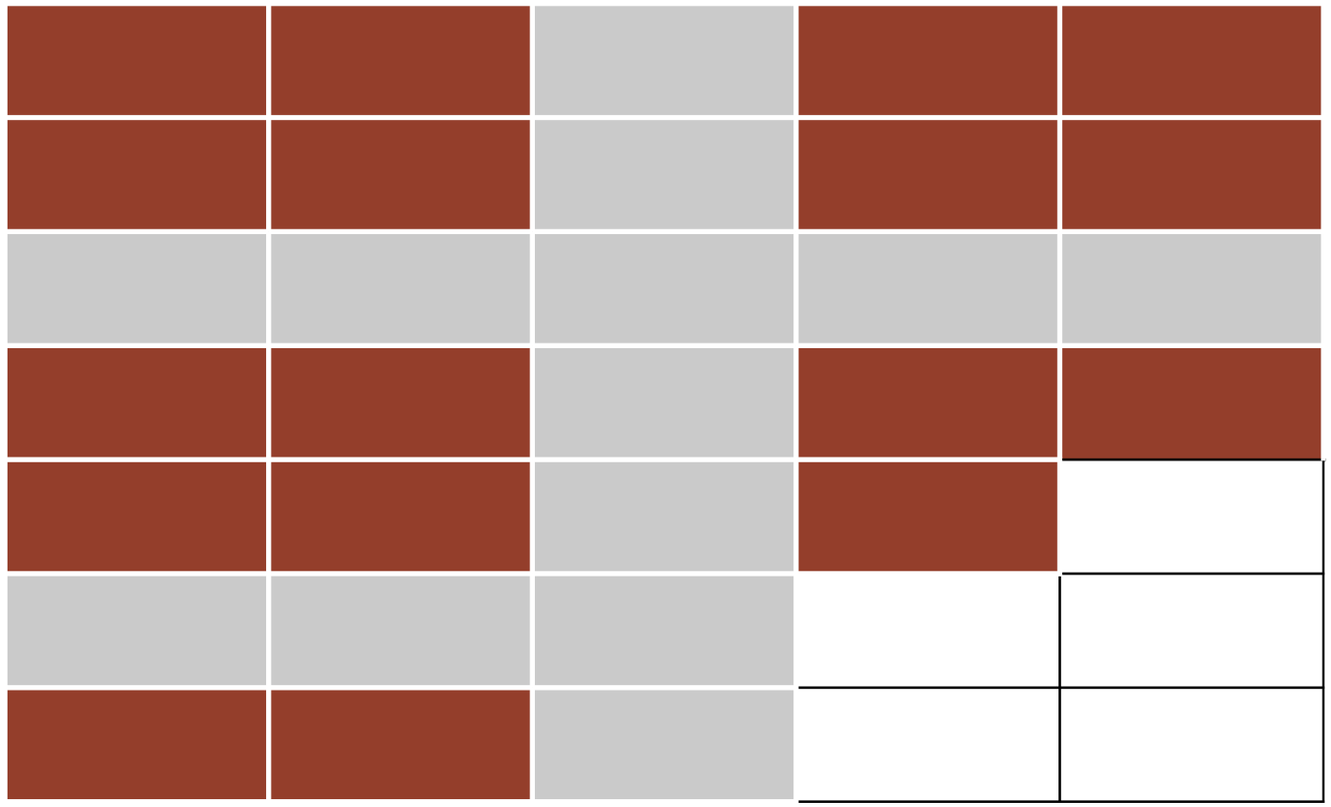
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



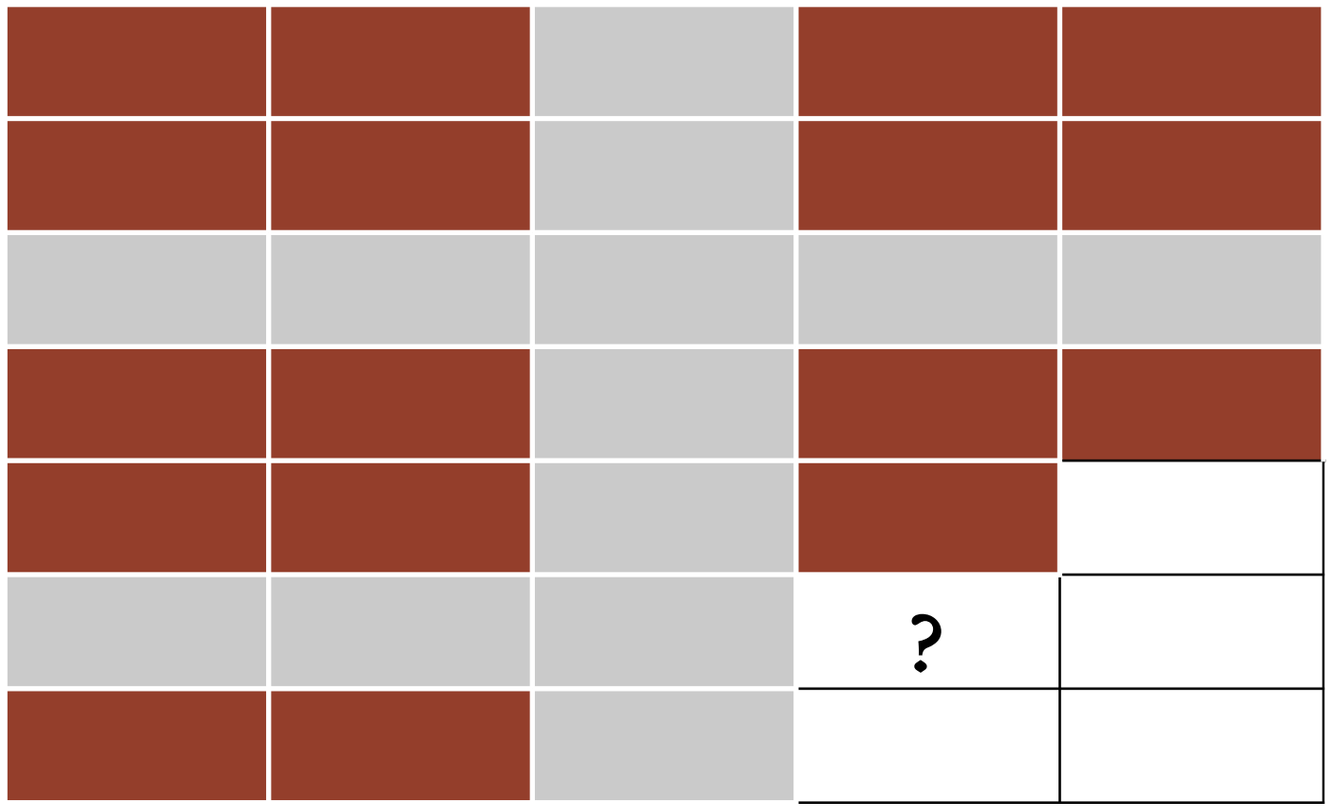
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



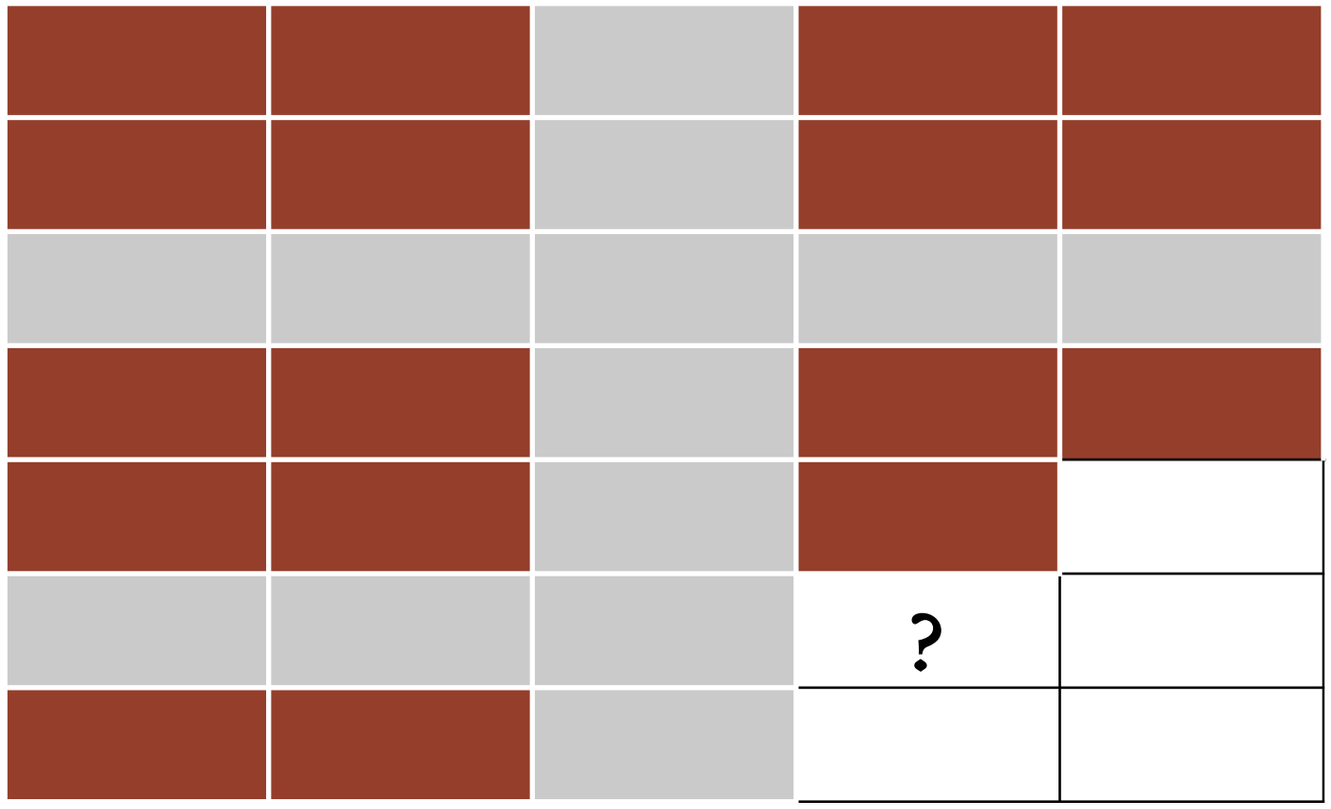
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



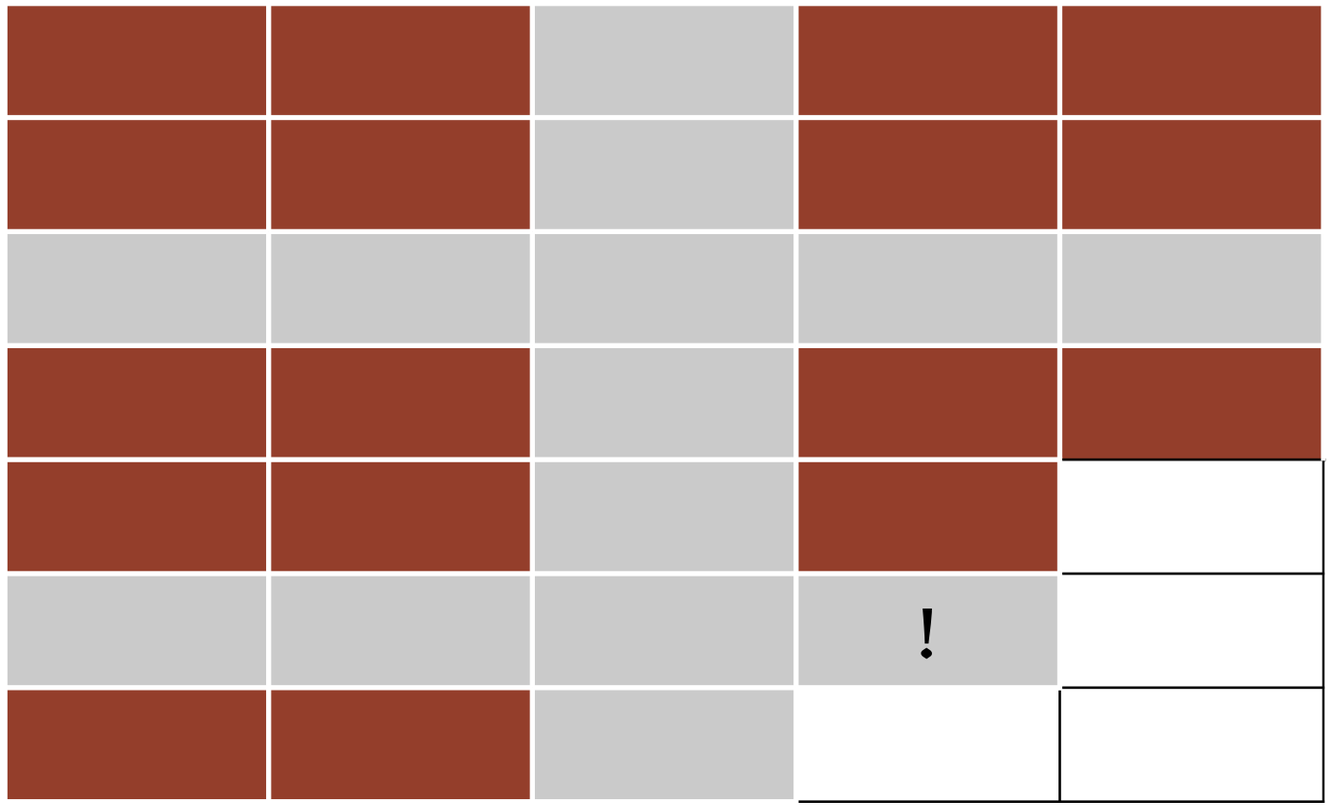
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



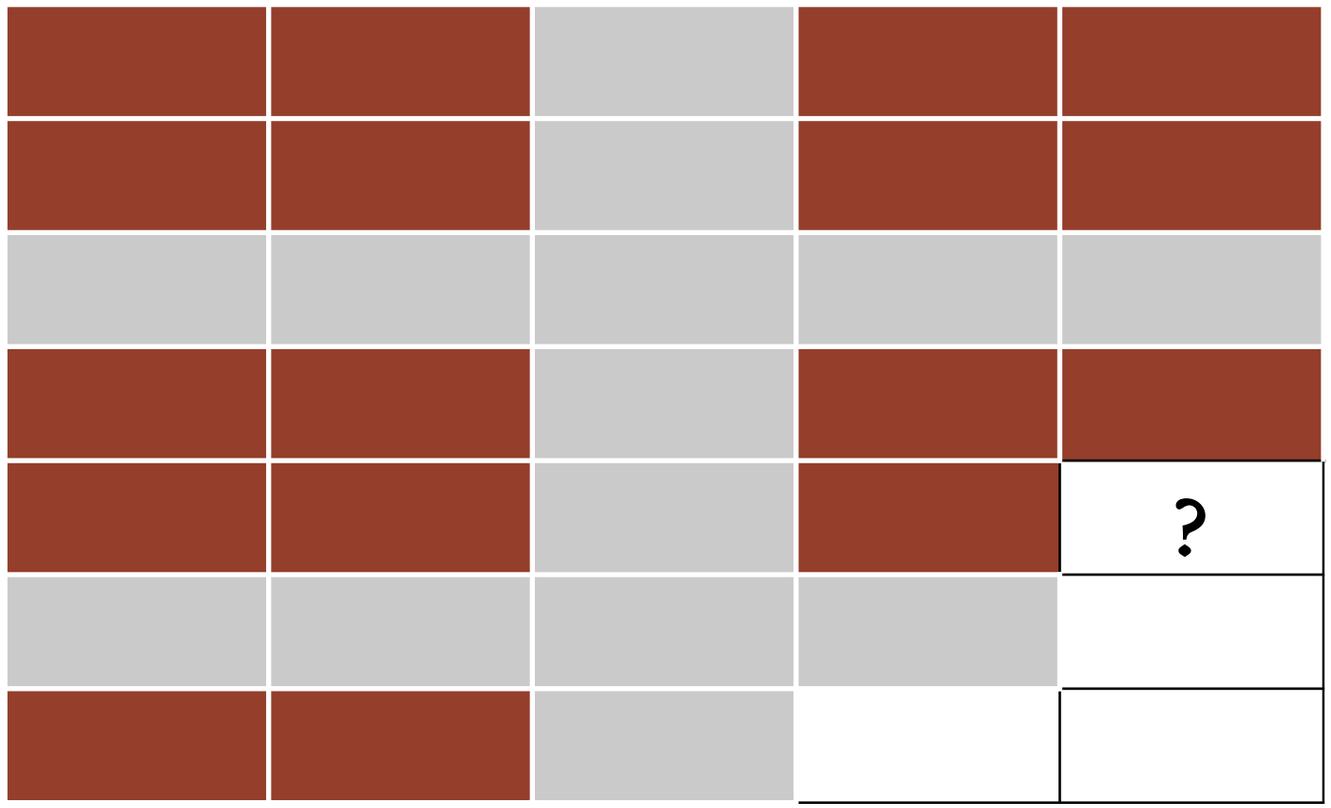
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



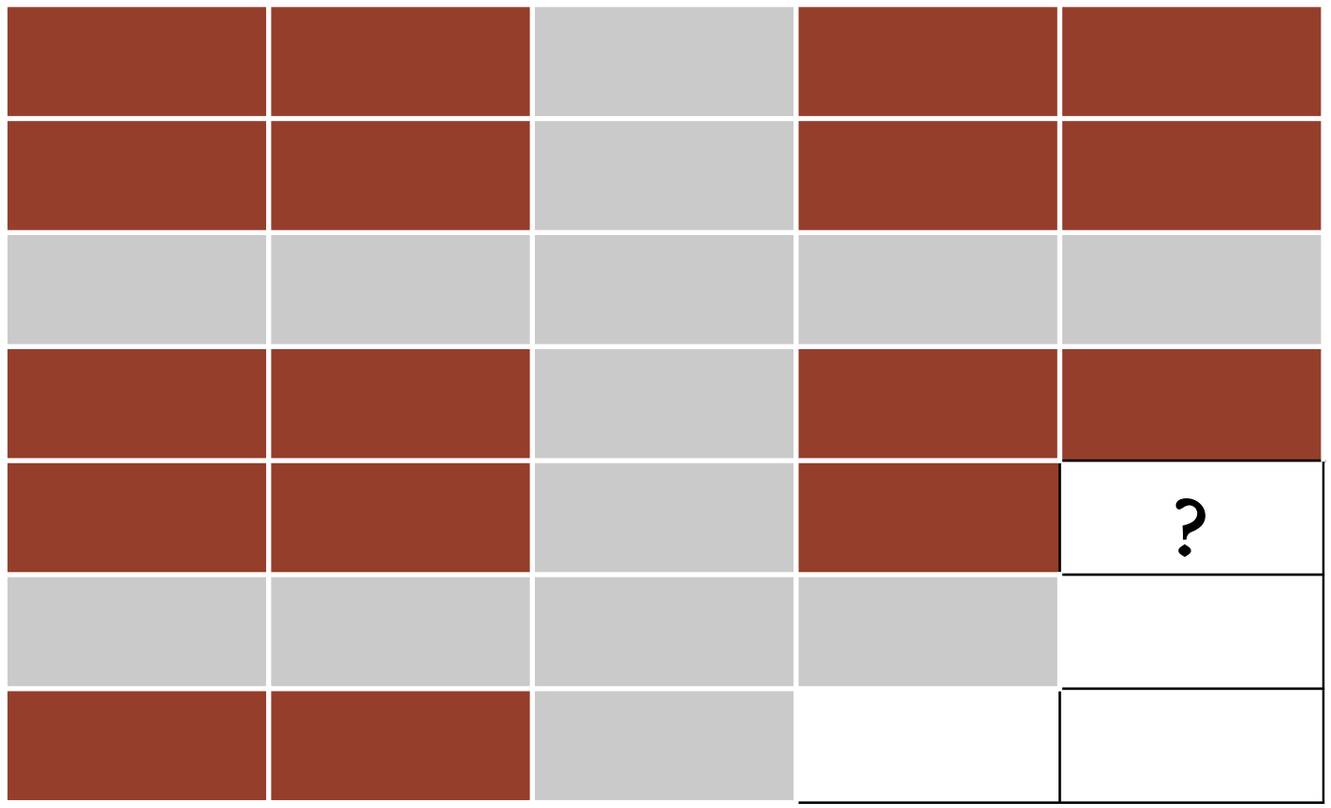
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



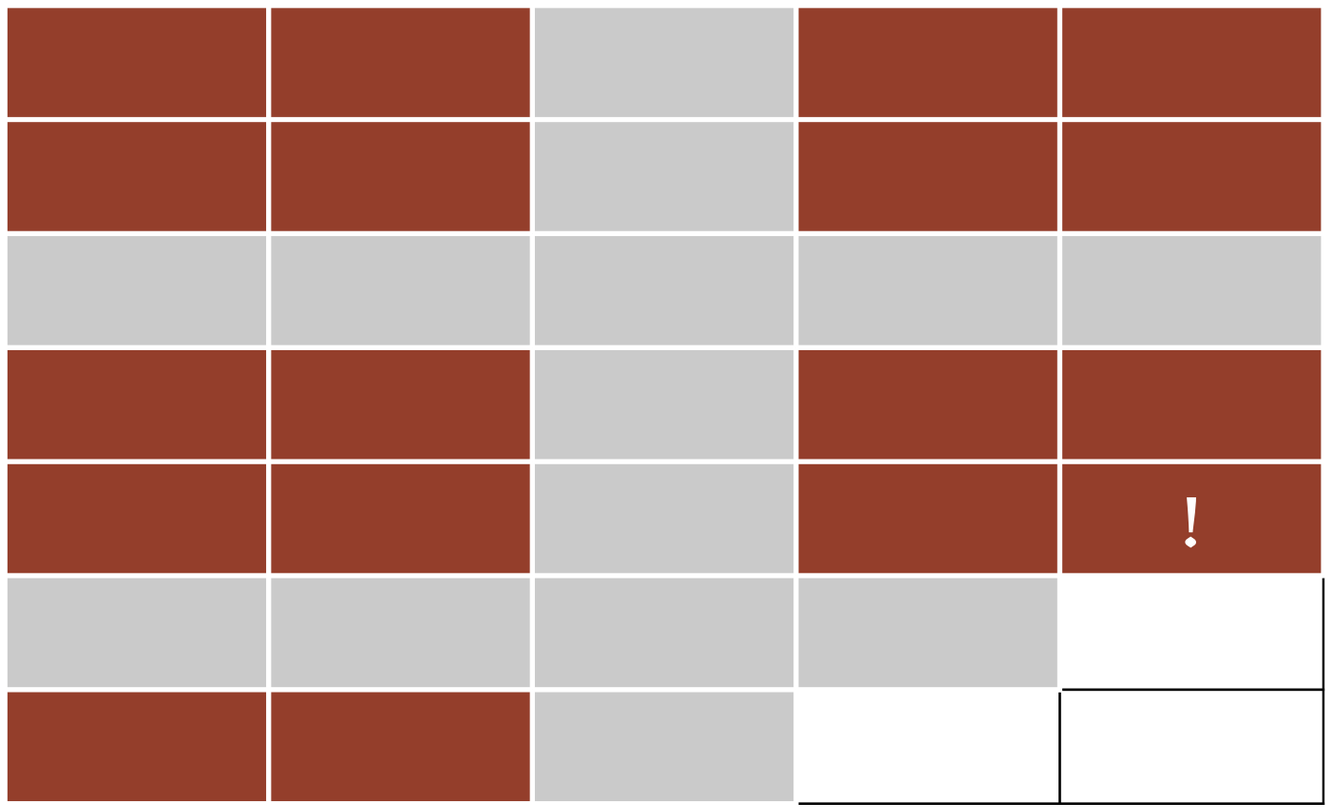
CONTINUE THE TEXTURE

	red	grey
left red, above red	3	0
left red, above grey	3	4
left grey, above red	2	4
left grey, above grey	1	2



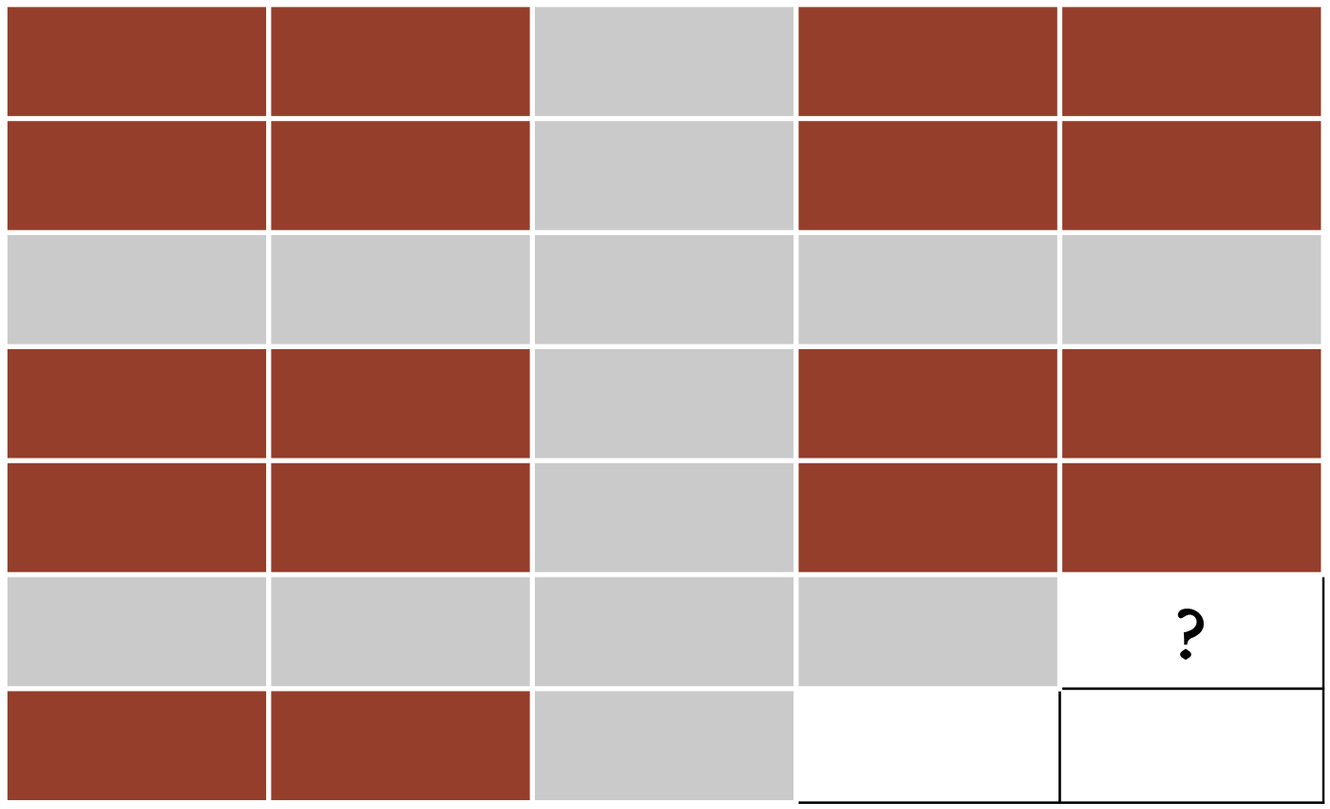
CONTINUE THE TEXTURE

	red	grey
left red, above red	3	0
left red, above grey	3	4
left grey, above red	2	4
left grey, above grey	1	2



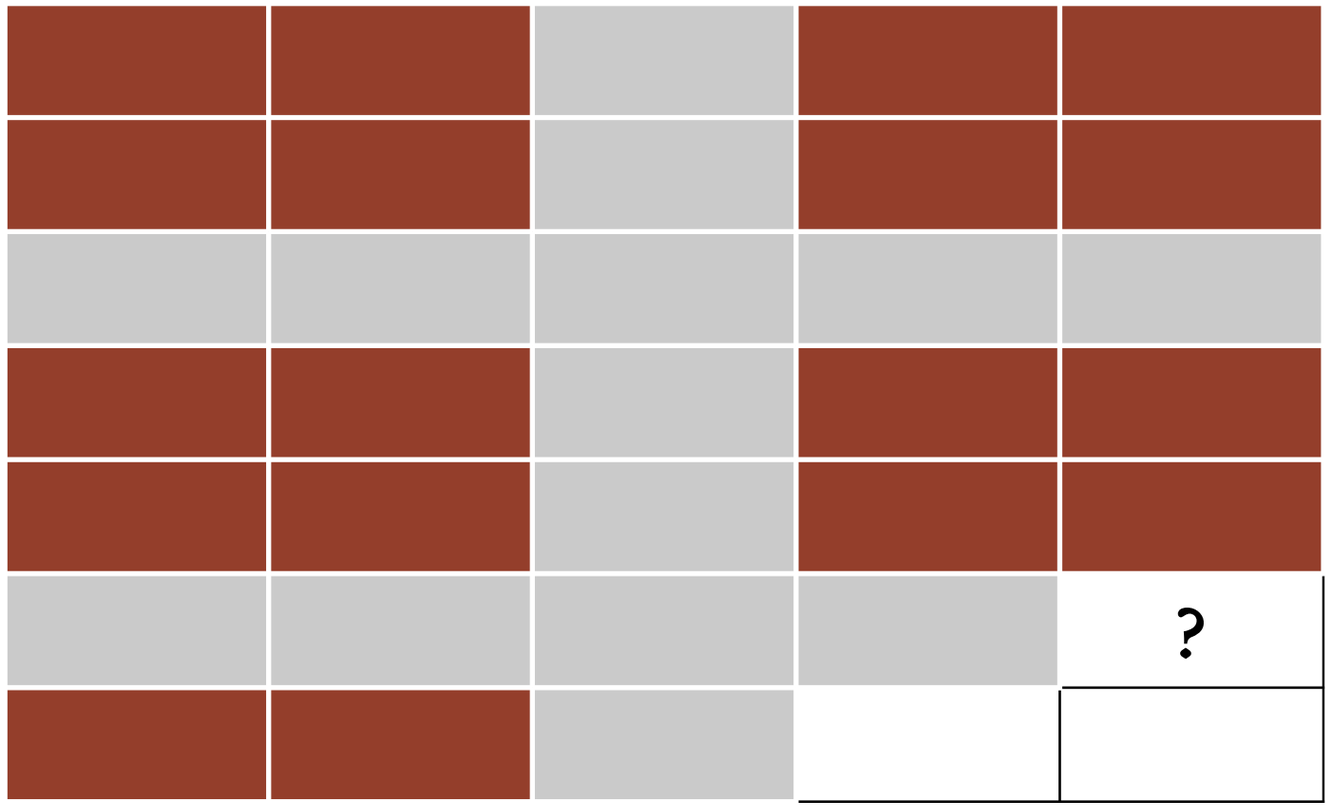
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



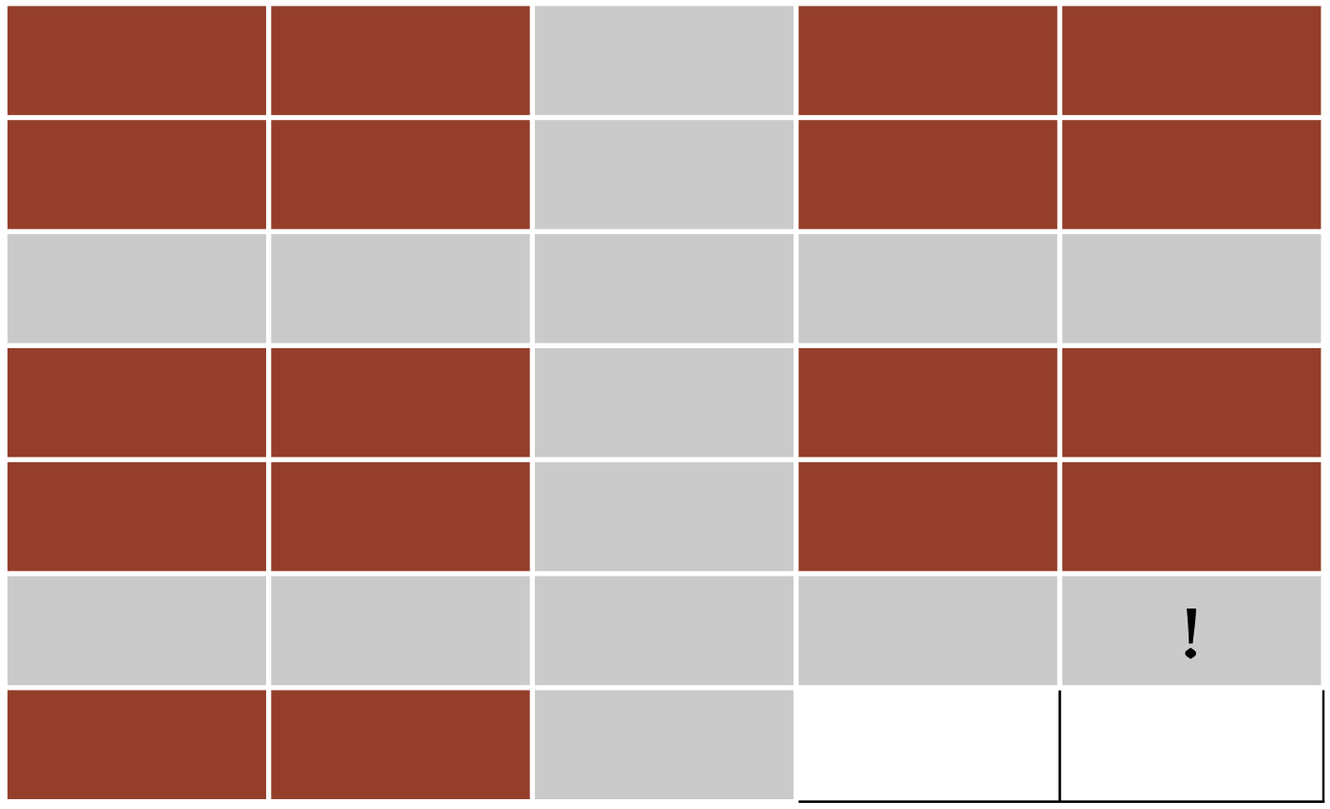
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



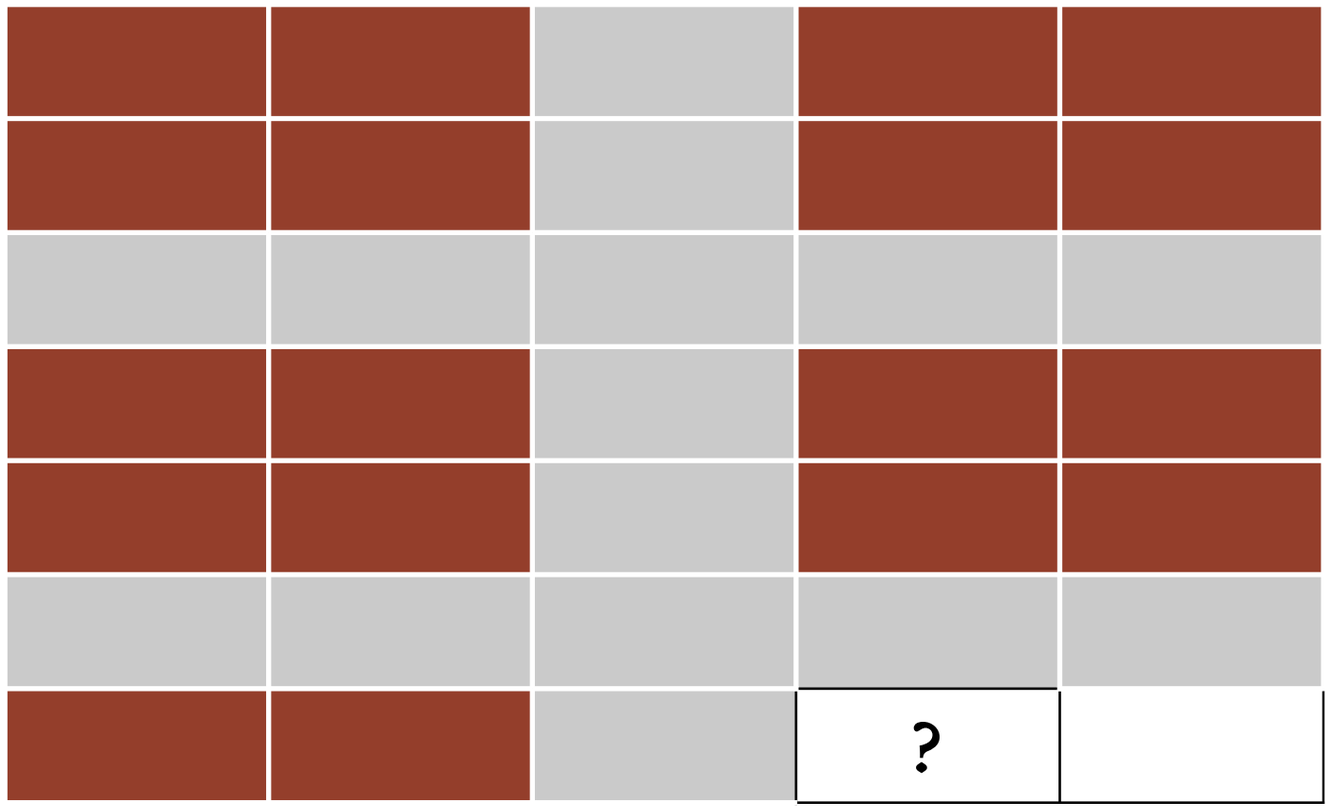
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



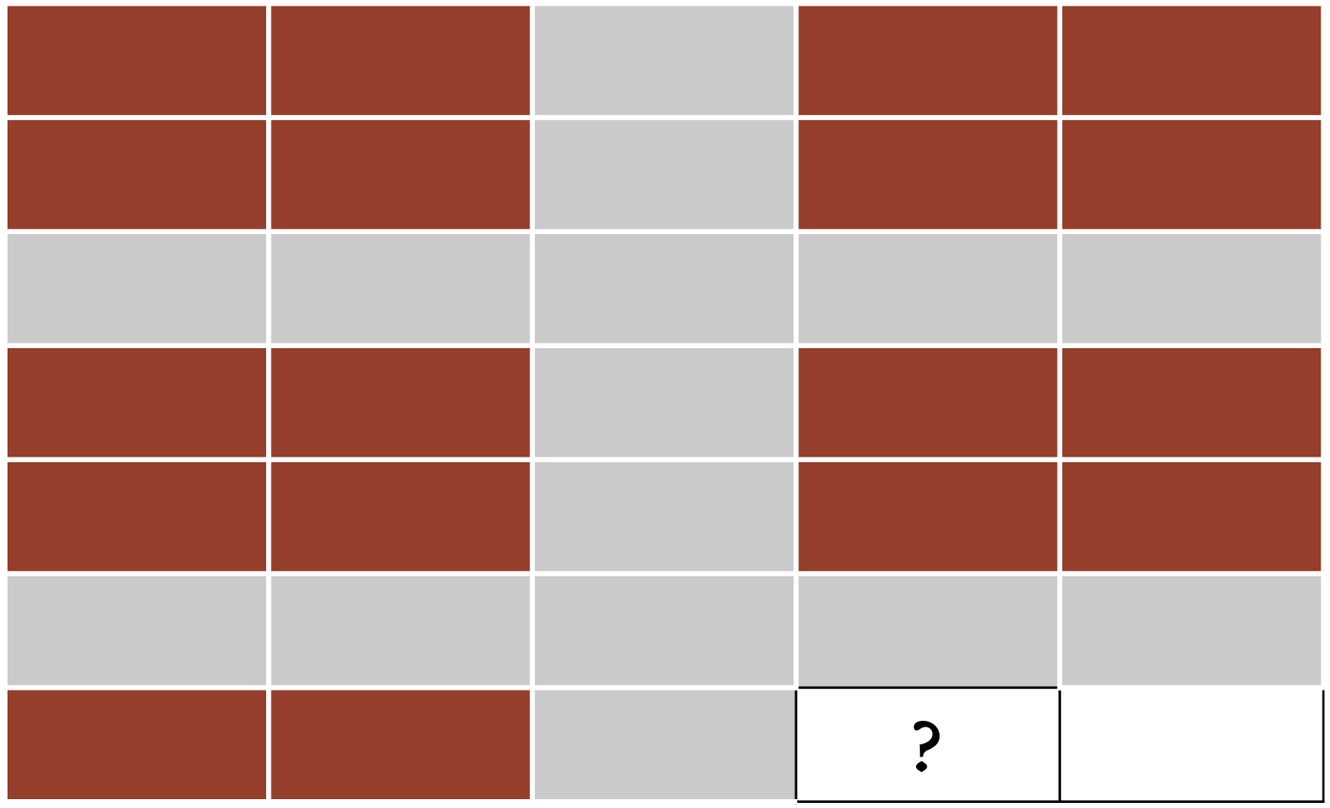
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



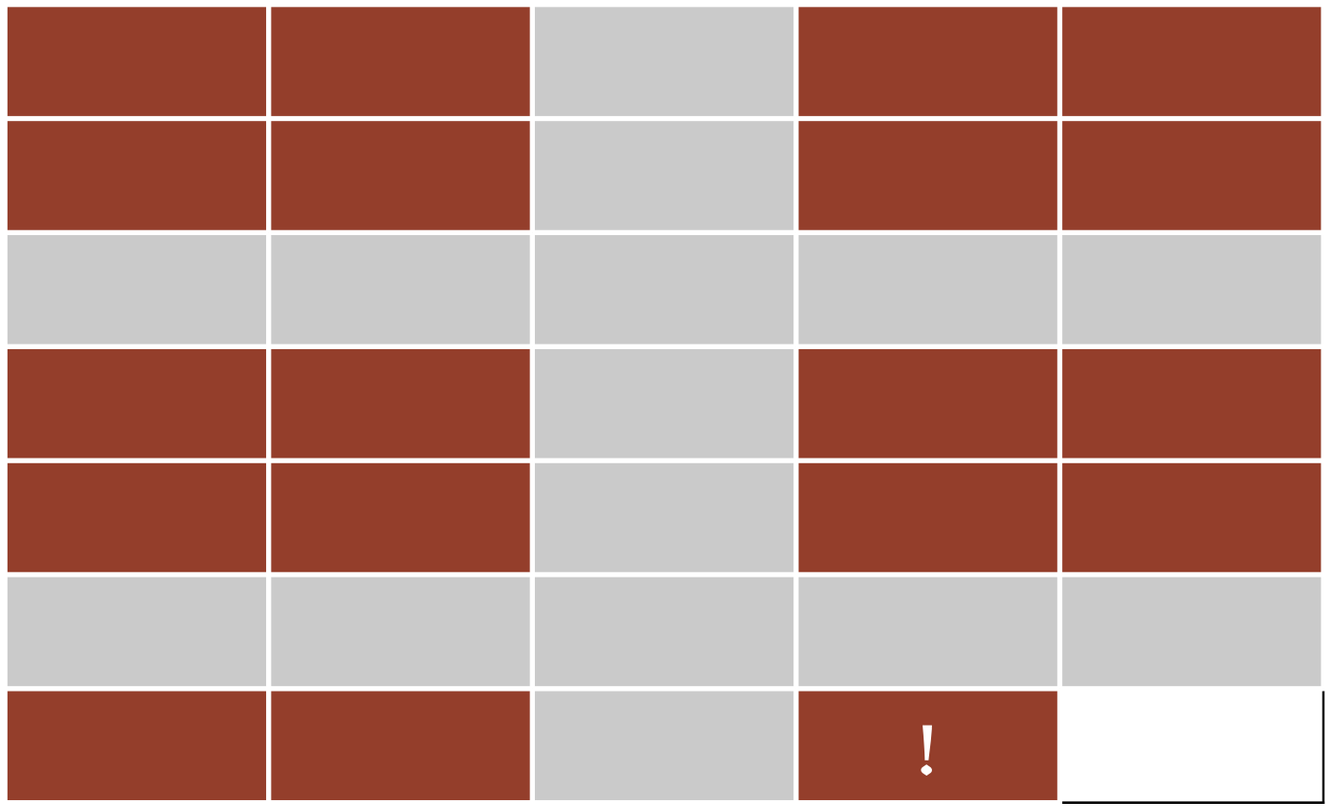
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



CONTINUE THE TEXTURE

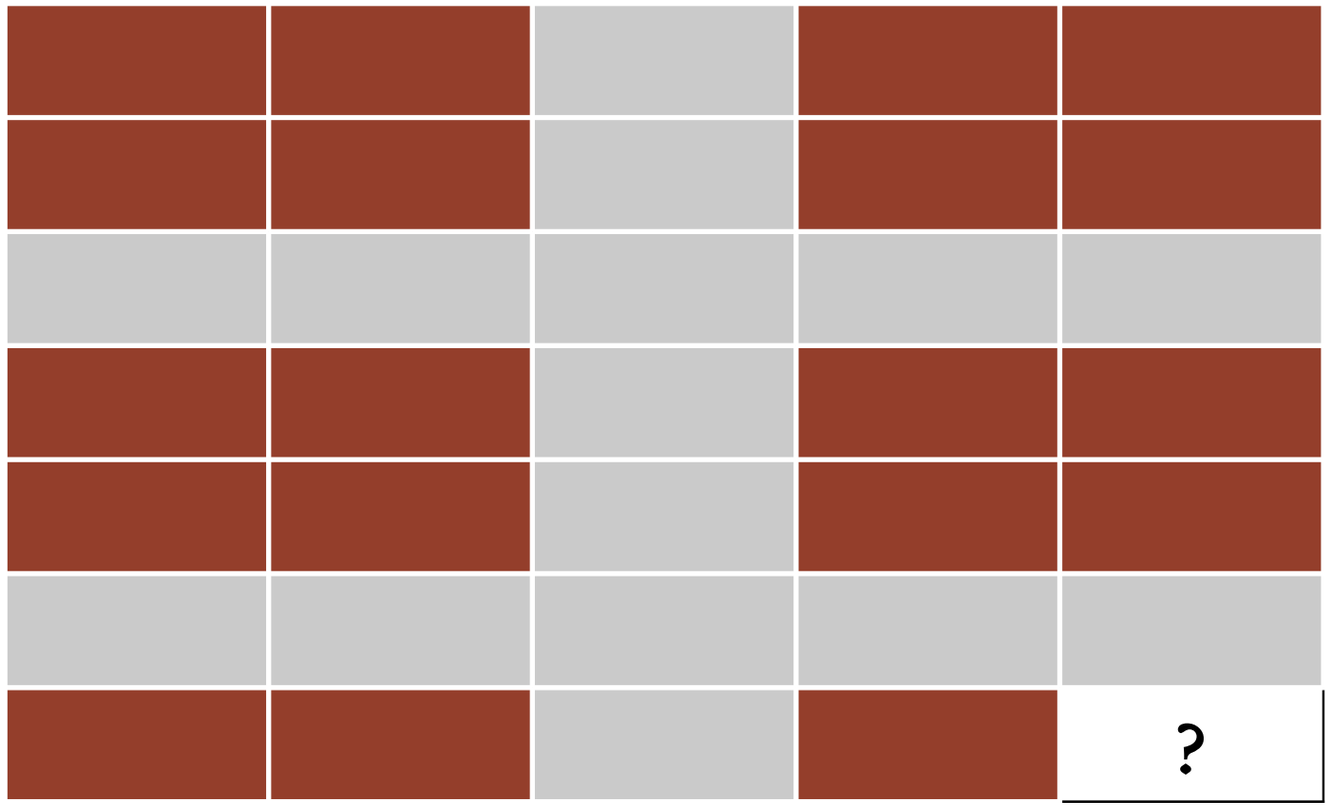
	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



with some stochasticity

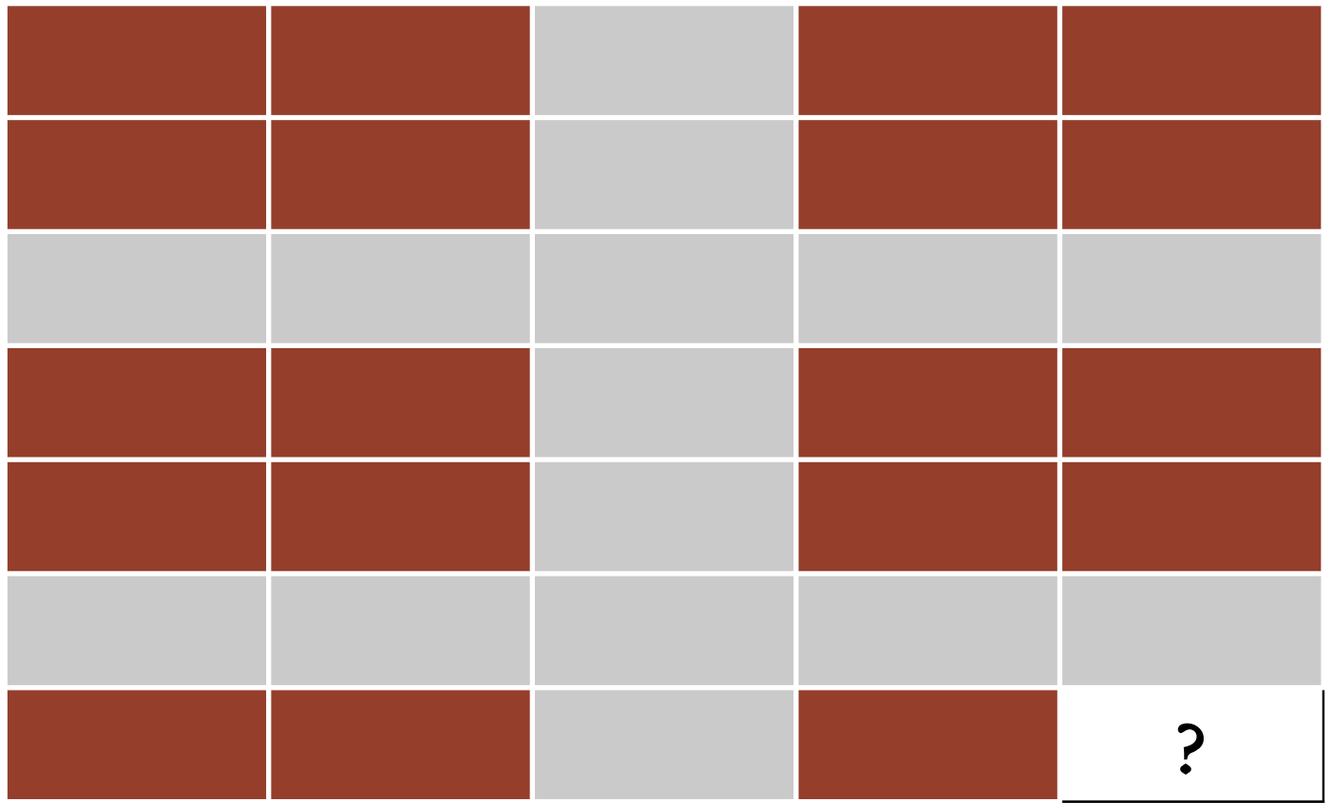
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



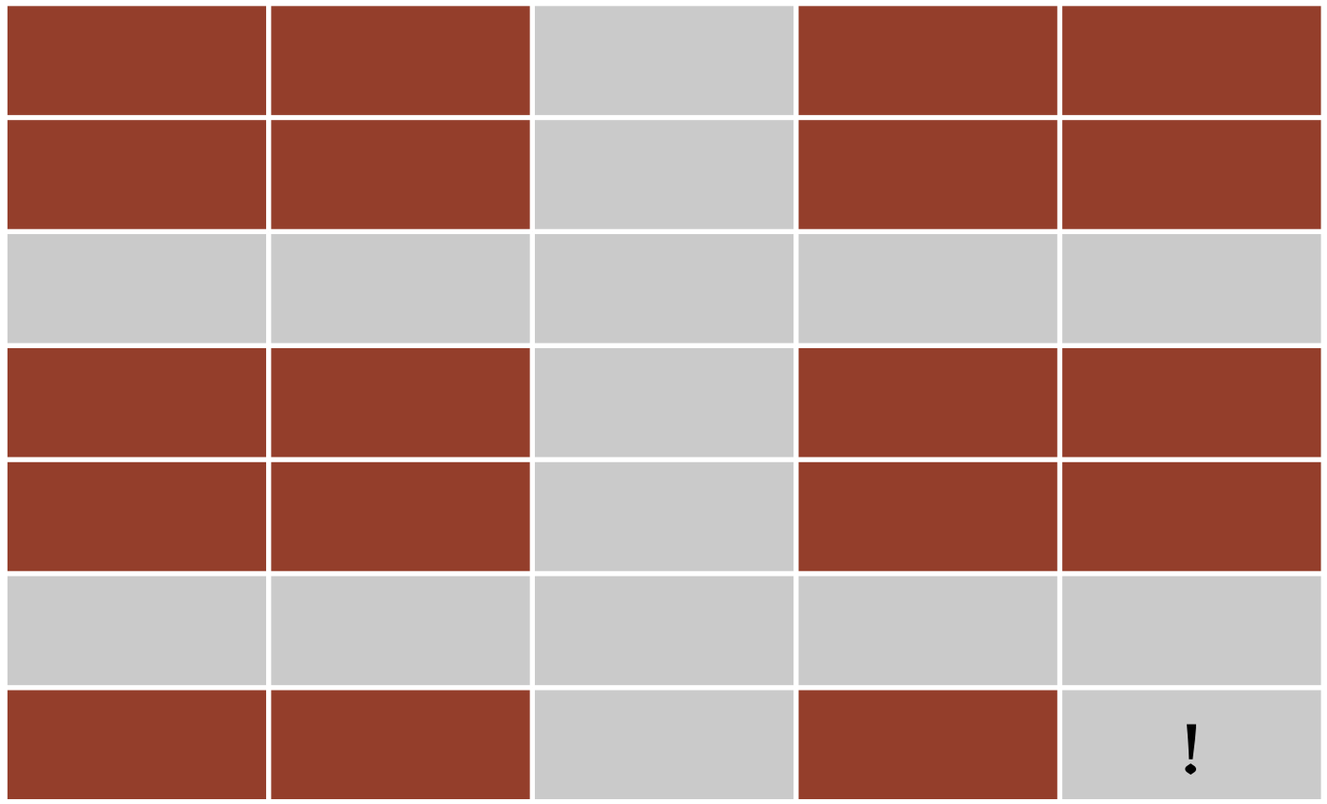
CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



CONTINUE THE TEXTURE

	red	grey
left **red**, above **red**	3	0
left **red**, above **grey**	3	4
left **grey**, above **red**	2	4
left **grey**, above **grey**	1	2



small mistake

BUT THERE'S A PROBLEM!

- words are categorical (finite & unordered), pixels are continuous
  - discretization leads to millions of colors
  - info about **red** should also (partially) apply to **dark red**
- building a probability table is unfeasible
  - gets exponentially larger when considering more neighbors
  - one image is not exhaustive, exact match might not be present
- **solution:** mimic table by looking up directly in the image
  - don't store anything, search every time

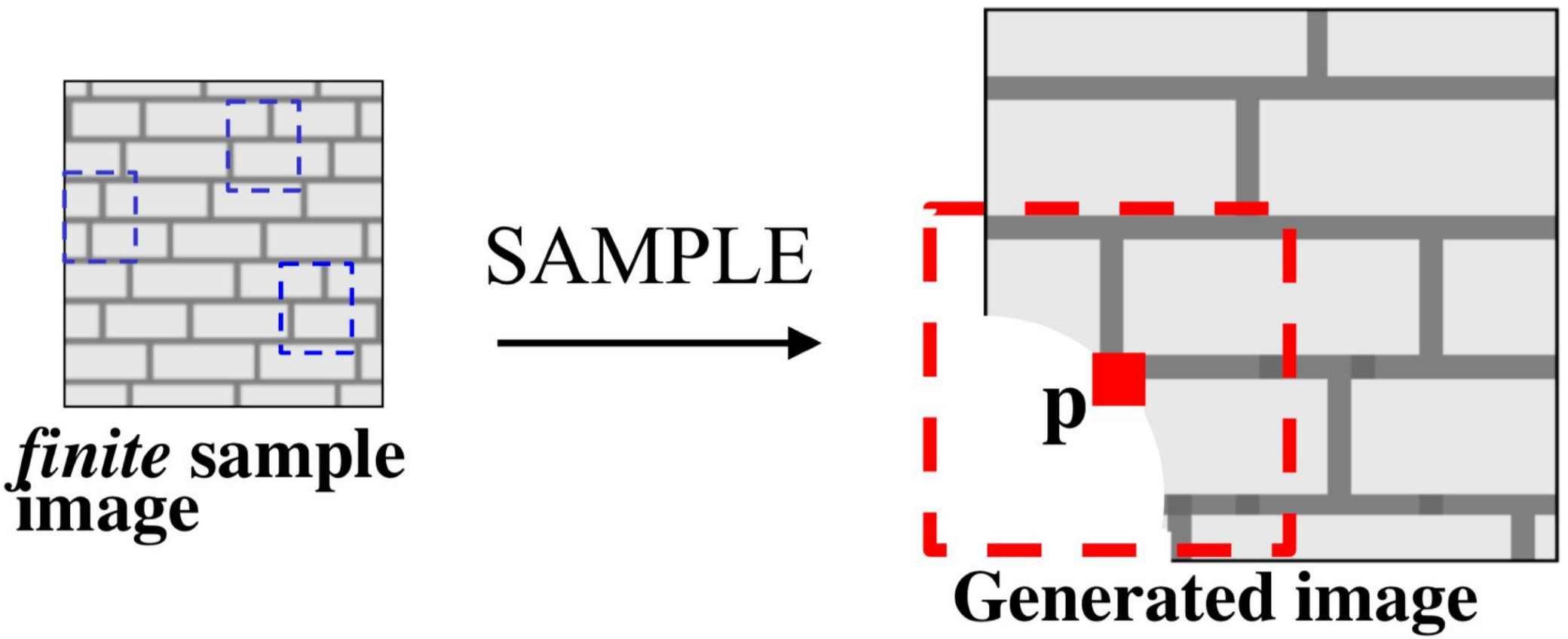
ALGORITHM

- input: image and hole position

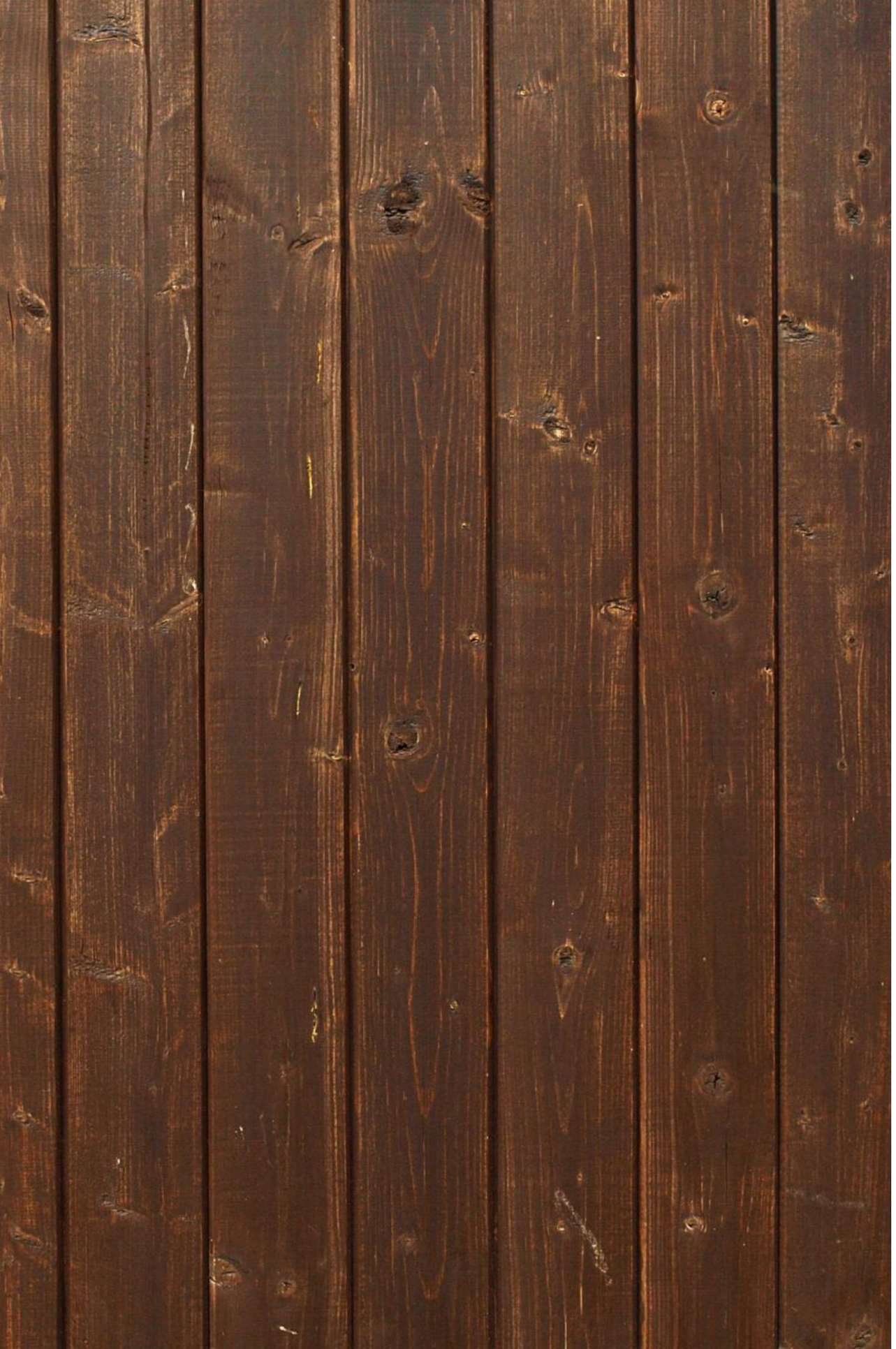
    1. pick a pixel on the hole's edge
       empty pixels having at least one filled neighbor
    2. select a patch around it
    3. find the most similar patch to it
       using sum of squared distances
    4. replace pixel with found patch center
    5. repeat until the hole is filled

- output: filled image

ALGORITHM



source: Texture Synthesis Paper presentation

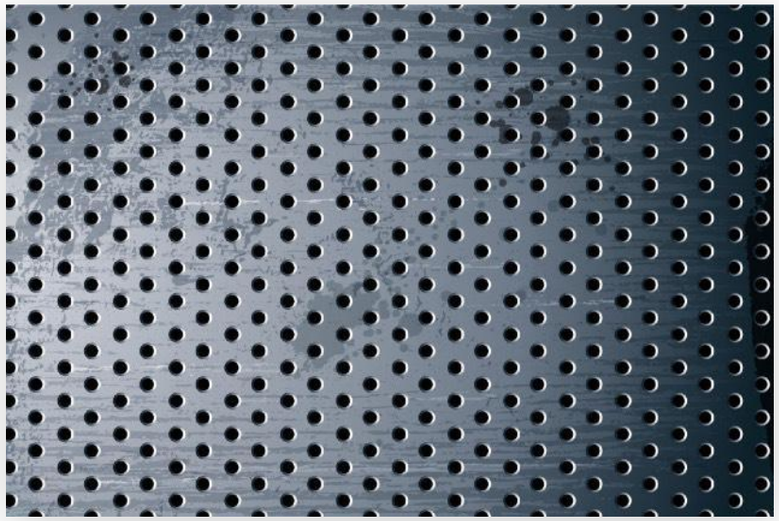


ONE LAST CONSIDERATION

- texture types:
  - regular (repeating)
  - stochastic
  - combination
- add stochasticity by not always picking the closest match

ALGORITHM PARAMETERS

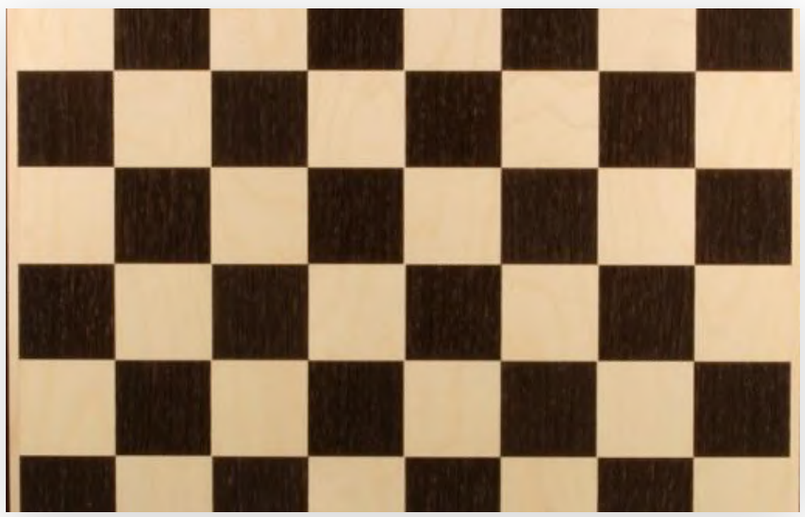
- *patch size*: texel size (how often texture repeats)



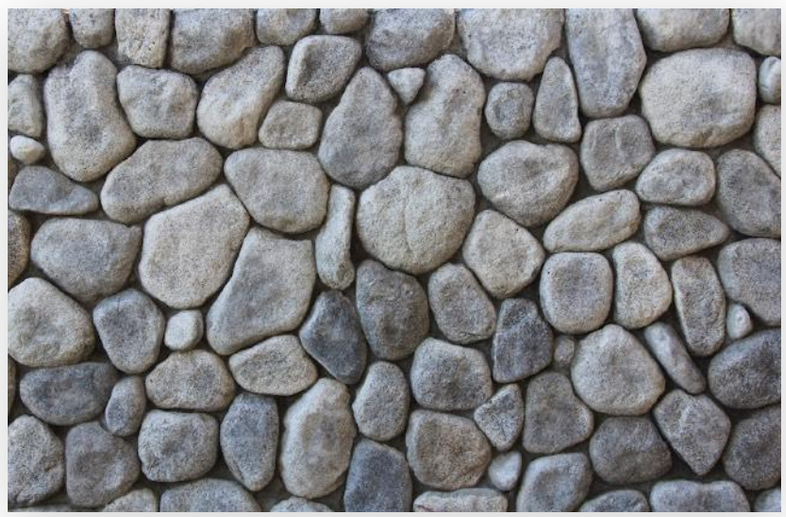
SMALL

LARGE

- *selection std*: texture randomness (picking most similar)



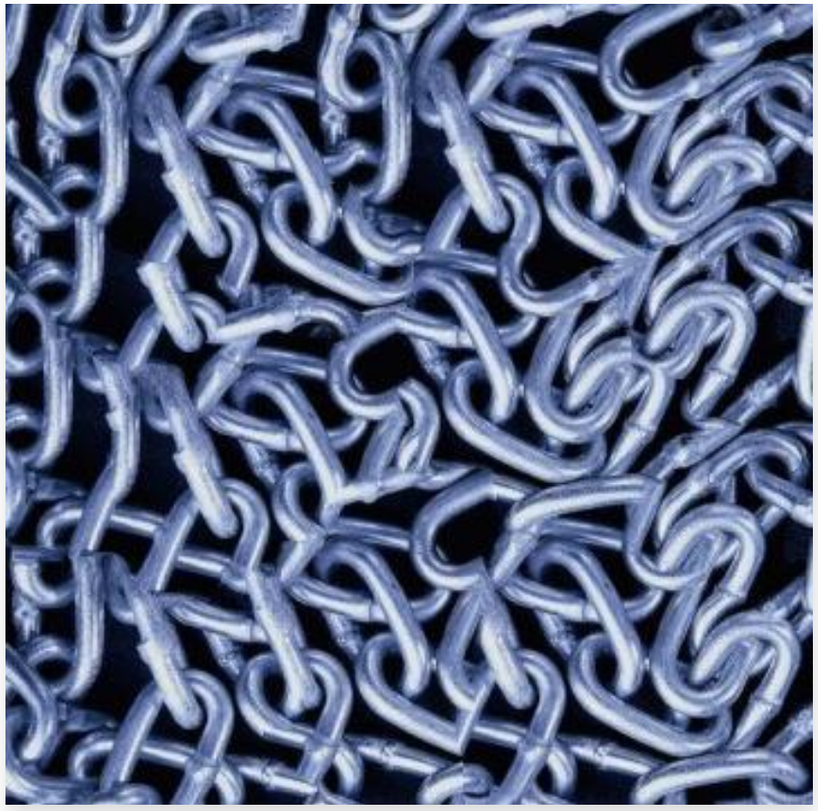
LITTLE

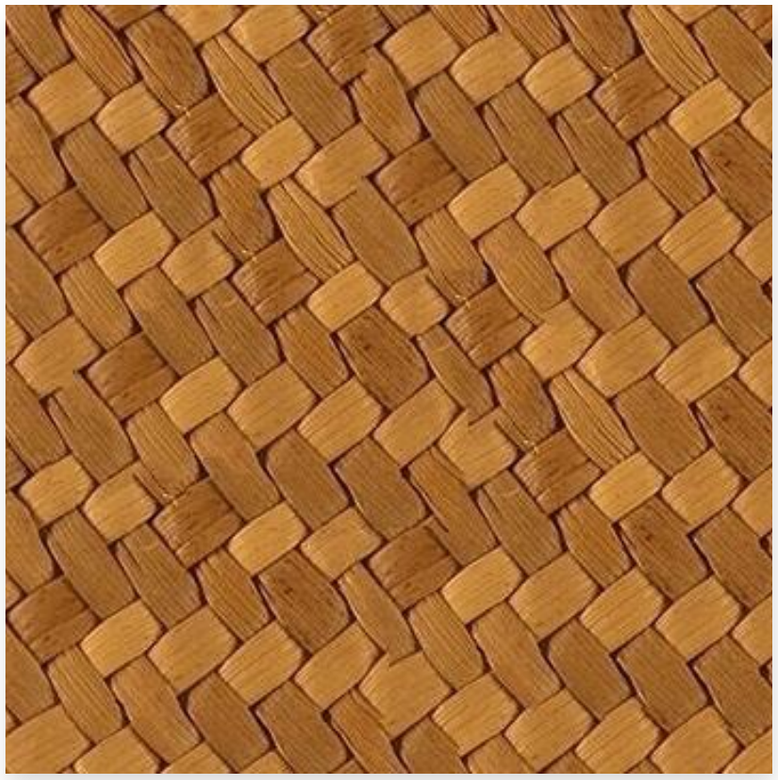


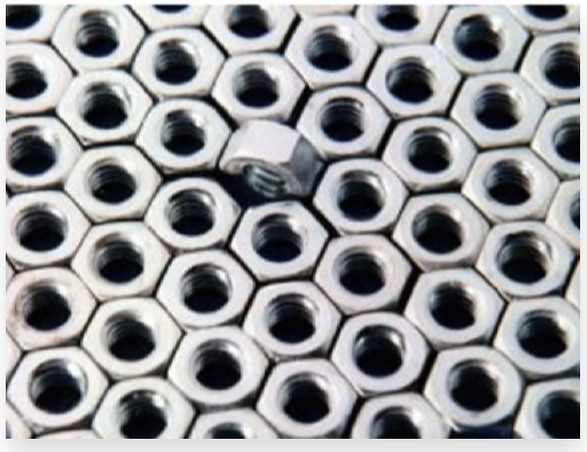
A LOT

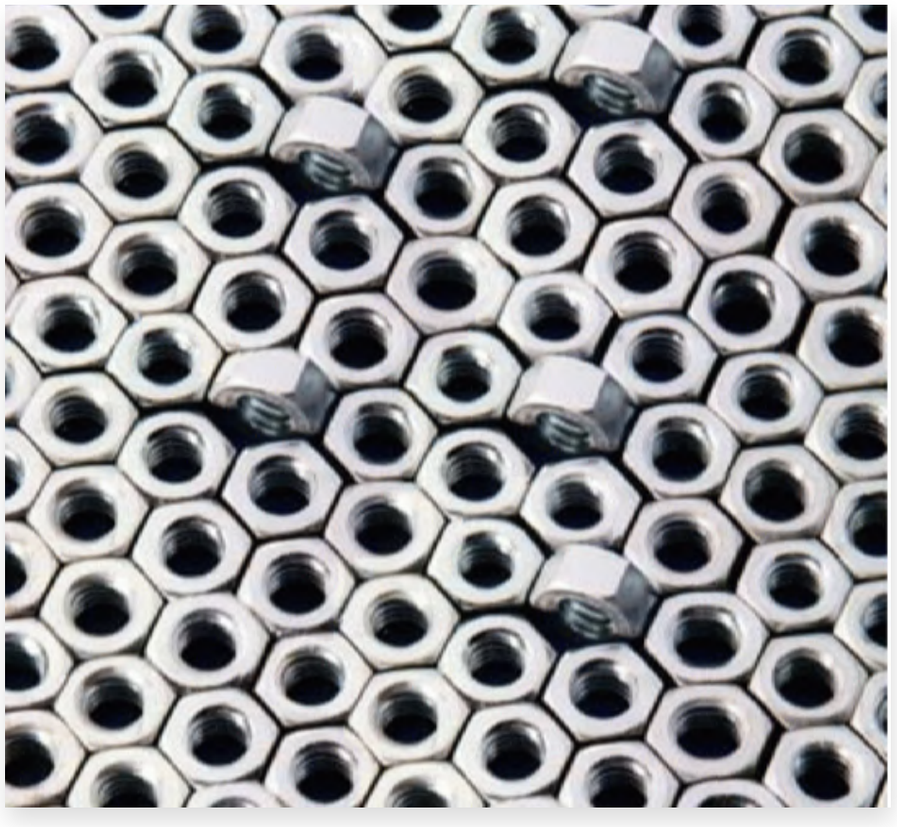
EXAMPLES

not run by me









DEMO

PRACTICAL CONSIDERATIONS

- to speed up computation:
  - replace the whole patch instead of the center pixel
  - restrict searching from whole image to a smaller area

ORIGINAL

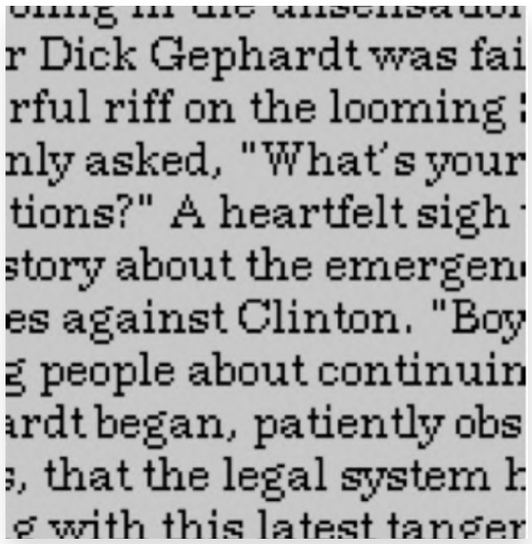
REMOVED

THAT WAS NOT A TEXTURE!

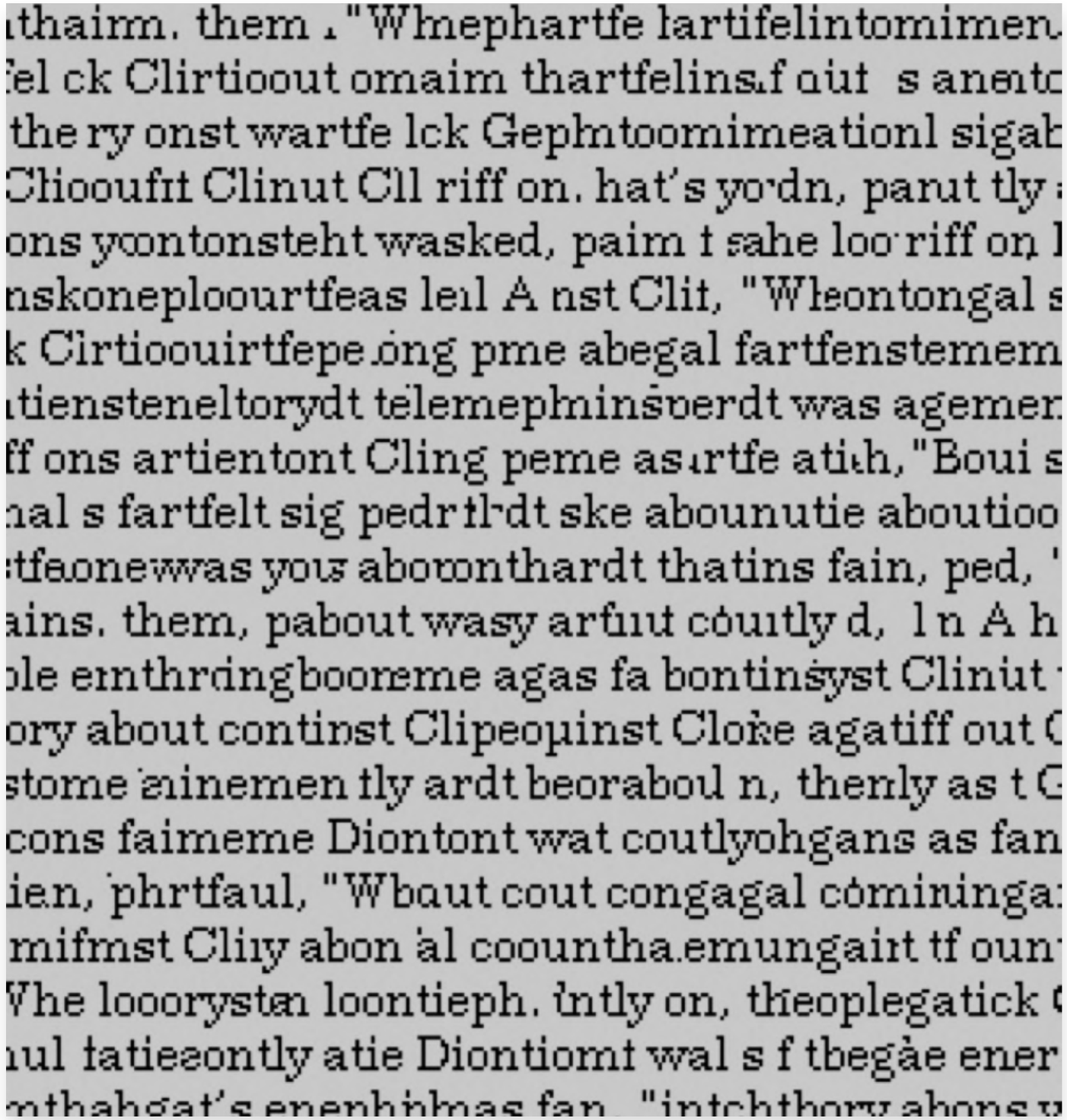
- turns out texture synthesis was not a detour
- photos are made up of many "textures"
- the method can be used for many tasks

MORE EXAMPLES

not run by me, various methods



WORDS



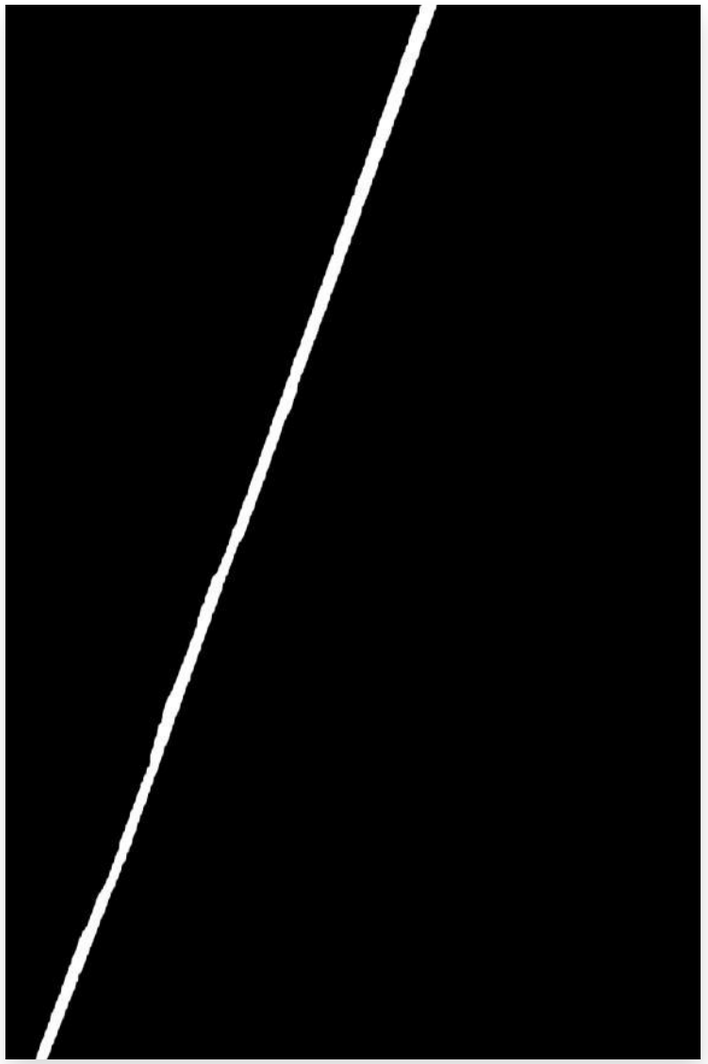
NOT ACTUAL WORDS!

ORIGINAL

EXPANDED

IMAGE RESTORATION

DAMAGED



MASK

RESTORED

WRINKLE REMOVAL

ORIGINAL

MASK

REMOVED

OBJECT REMOVAL

ORIGINAL

MASK

REMOVED

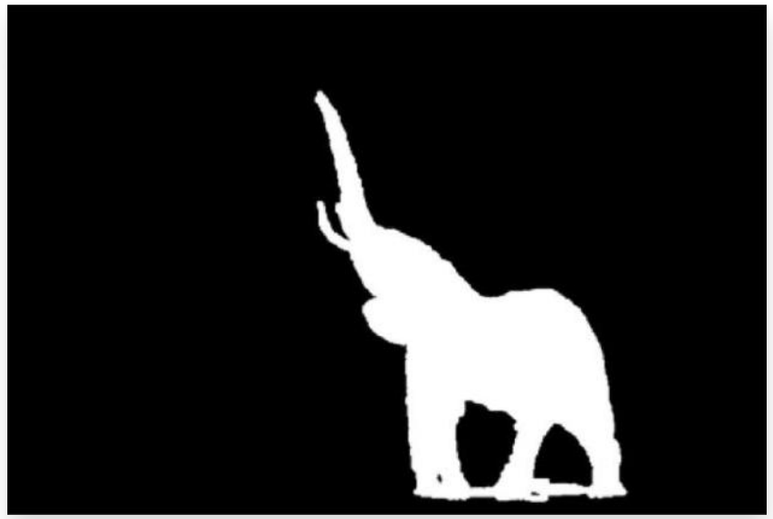
OBJECT REMOVAL

ORIGINAL

MASK

REMOVED

OBJECT REMOVAL

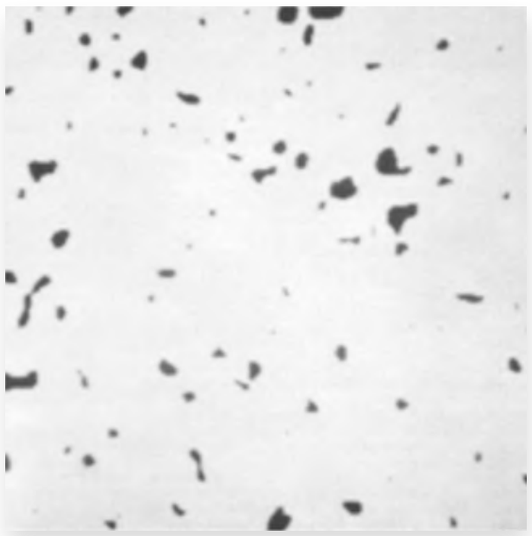





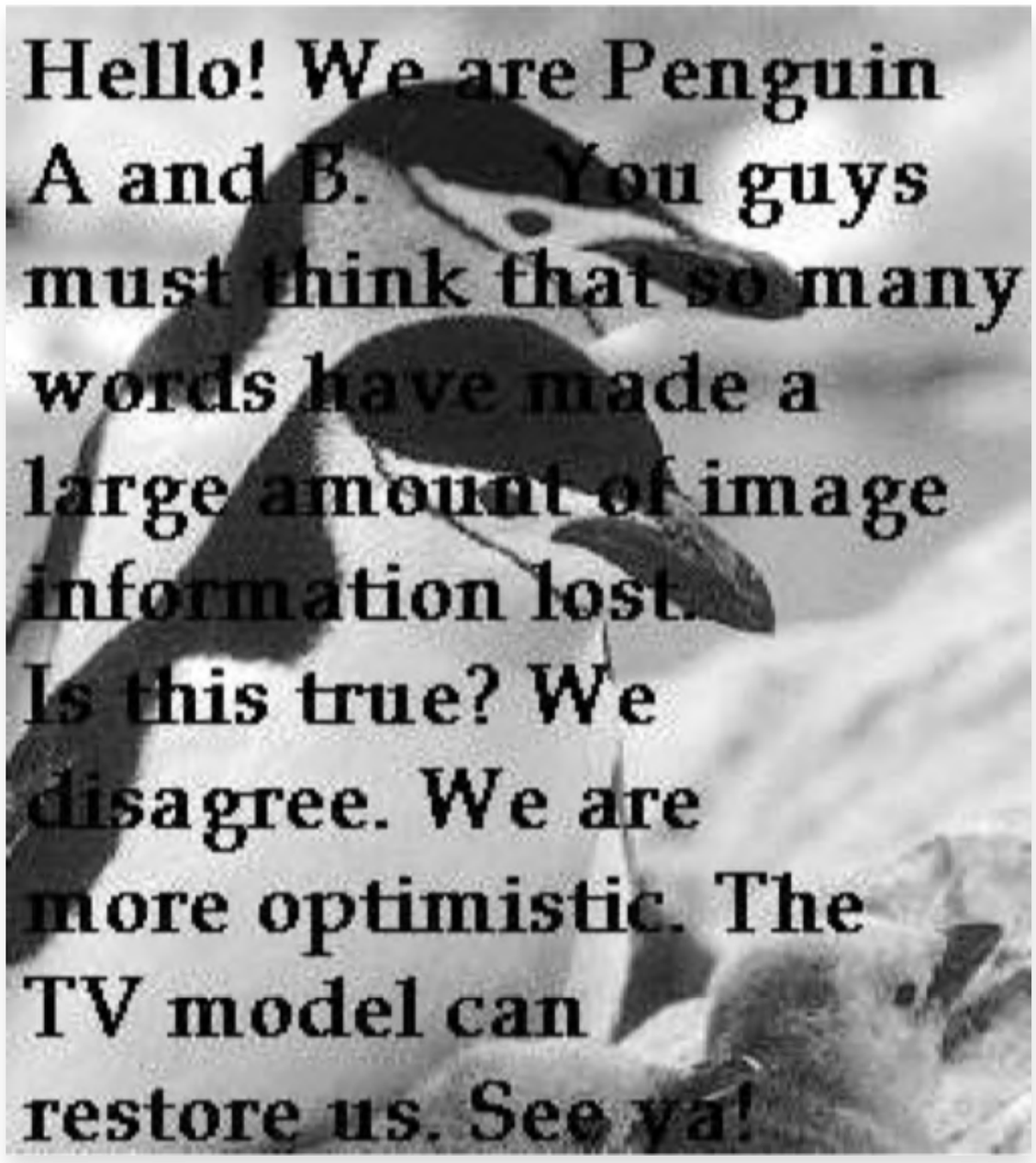

有本明功
館幸雄
庄司重太郎
細田


有本明功
館幸雄
庄司重太郎
細田


有本明功
館幸雄
庄司重太郎
細田



TEXT REMOVAL



METHOD COMPARISON

ORIGINAL

REMOVED

CONTENT-AWARE RESIZE

source: FMI UB CV lecture 4

IN-PAINTING

RESIZING

LIMITATIONS

ORIGINAL

FAILURE

SUMMARY

TEXTURE SYNTHESIS

- fundamentally sound
- conceptually simple
- widely applicable
- not very robust
- pretty slow

REFERENCES

- *Texture Synthesis by Non-Parametric Sampling*, A. Efros & T. Leung
- *Inpainting Methods Survey*, M. Bertalmio et al
- *Assignment 4 – CPSC 425: Computer Vision*, University of British Colombia

- many published example runs on various image inputs

THANK YOU!