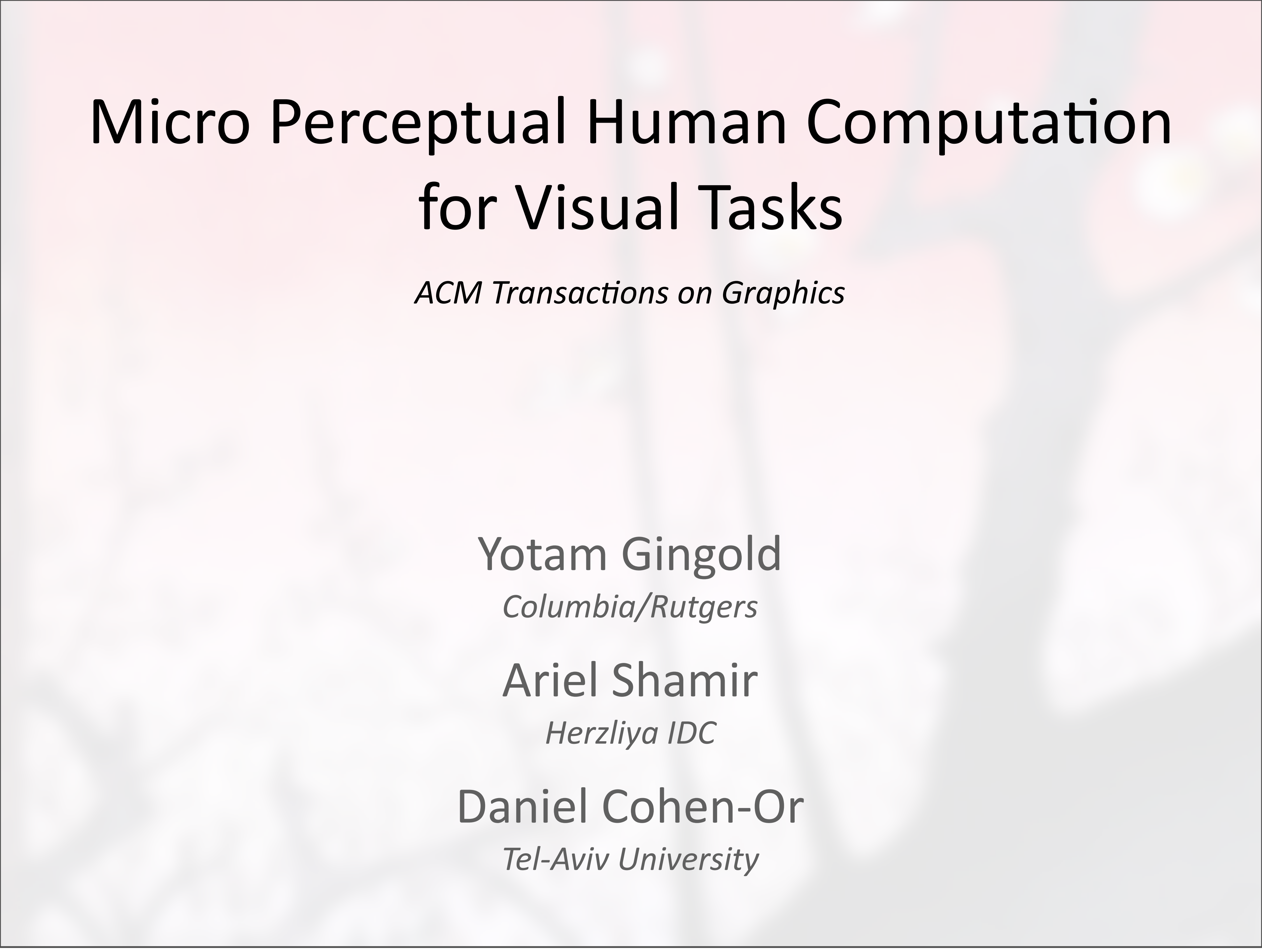


Micro Perceptual Human Computation for Visual Tasks

ACM Transactions on Graphics

Yotam Gingold
Columbia/Rutgers

Ariel Shamir
Herzliya IDC

Daniel Cohen-Or
Tel-Aviv University

This is joint work with Ariel Shamir and Daniel Cohen-Or when I was in Israel.

I’m going to talk about a model of computation that’s very relevant to graphics.
We call it “Micro Perceptual Human Computation” and I will show how it can be used for visual tasks.

The first part of my talk will discuss the nature of human computation and our particular computational model.
The second part will describe three specific algorithms for computing depth layers, normal maps, and symmetry maps.

Before I begin, I would like to offer a historical digression to loosen up our minds a bit.

Historical Digression

[David Alan Grier 2005]

1700's

Historical note:

The term "computer" used to refer to humans who did computation [Grier 2005 "When Computers Were Human"].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1700’s

Clairaut

Historical note:

The term “computer” used to refer to humans who did computation [Grier 2005 “When Computers Were Human”].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1700's

Clairaut

Halley's Comet

Historical note:

The term "computer" used to refer to humans who did computation [Grier 2005 "When Computers Were Human"].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1800's

Historical note:

The term "computer" used to refer to humans who did computation [Grier 2005 "When Computers Were Human"].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

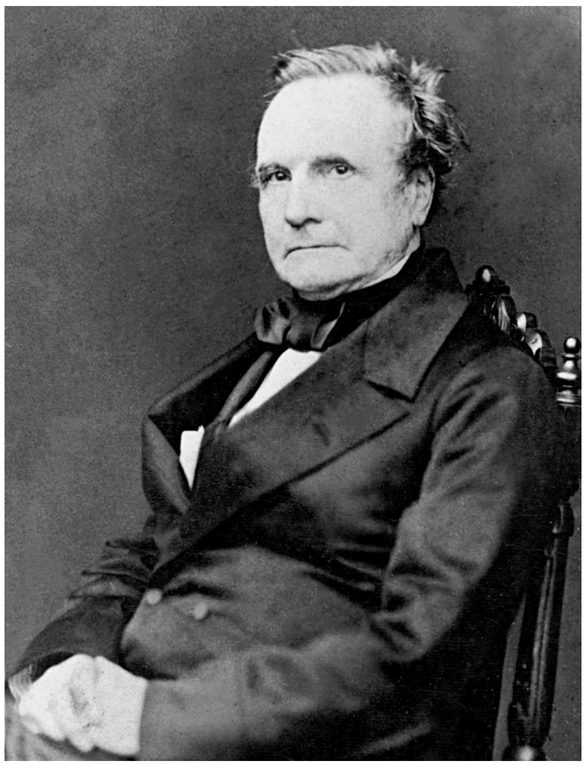
<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1800’s



Babbage

Historical note:

The term “computer” used to refer to humans who did computation [Grier 2005 “When Computers Were Human”].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

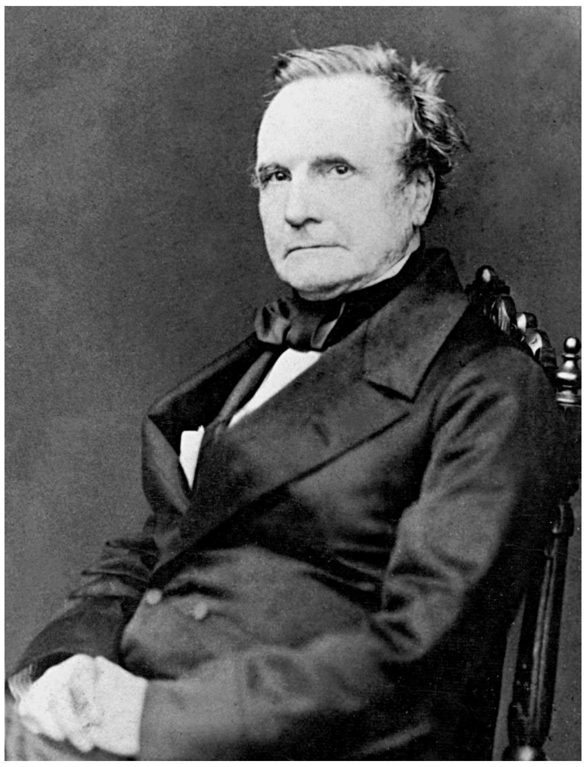
<click> In the 1940's, the ENIAC was developed, the first electronic computer.

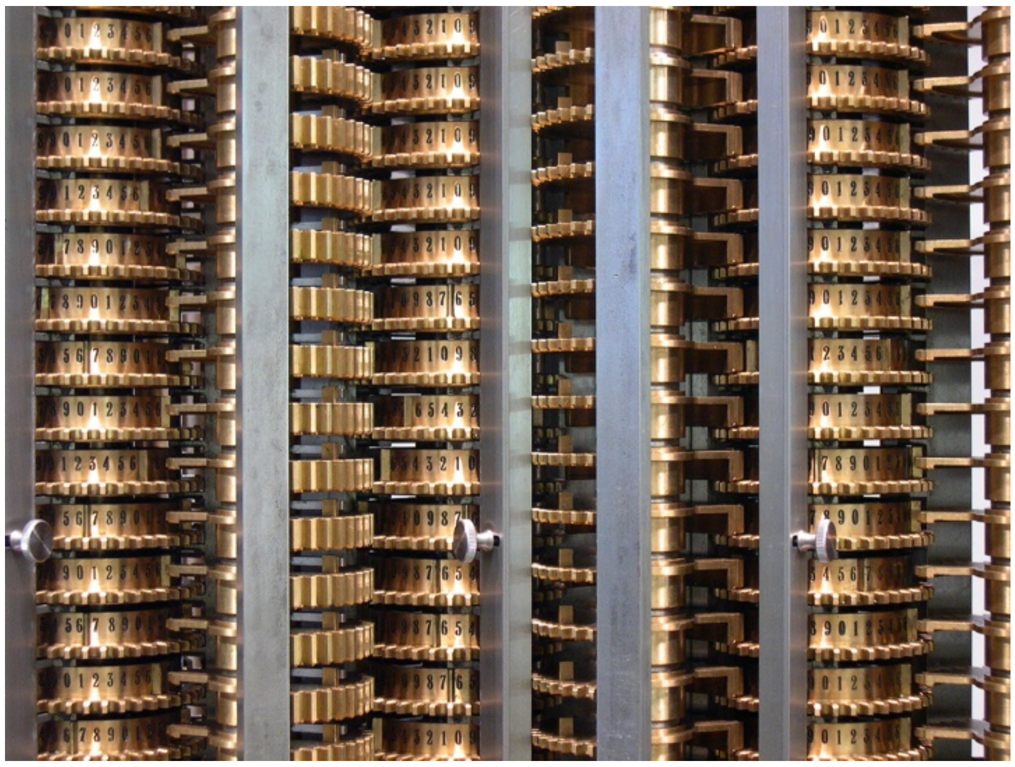
Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1800's





[Carsten Ullrich]

Babbage

Difference Engine

Historical note:

The term “computer” used to refer to humans who did computation [Grier 2005 “When Computers Were Human”].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1900's

Historical note:

The term "computer" used to refer to humans who did computation [Grier 2005 "When Computers Were Human"].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1900’s

WPA/war effort/NACA

Historical note:

The term “computer” used to refer to humans who did computation [Grier 2005 “When Computers Were Human”].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

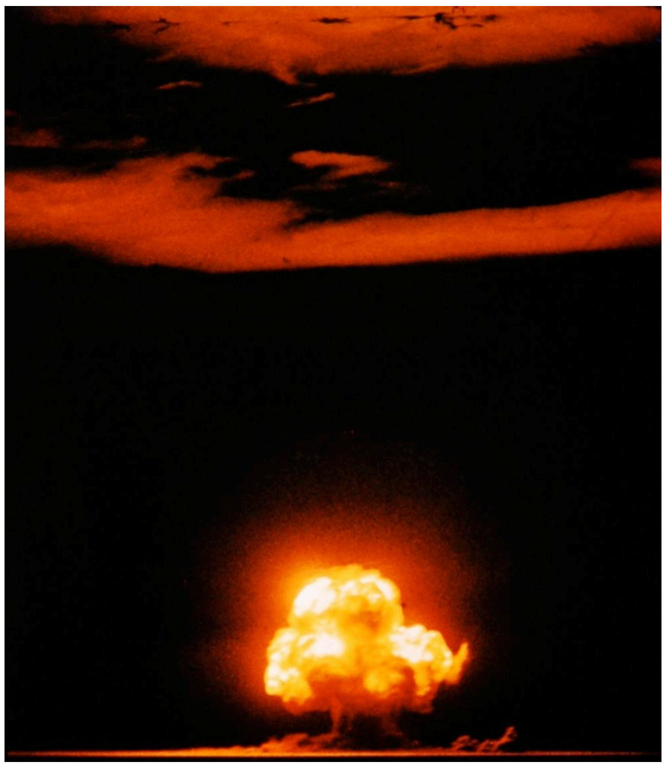
Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1900's

WPA/war effort/NACA



Trinity

Historical note:

The term "computer" used to refer to humans who did computation [Grier 2005 "When Computers Were Human"].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

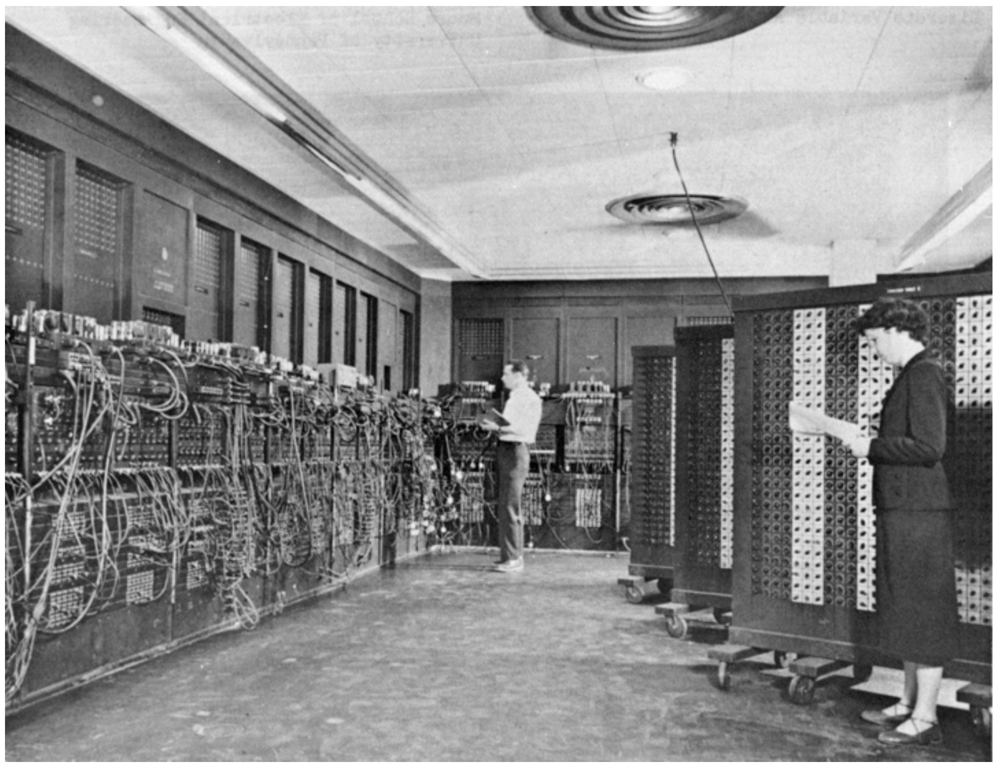
<click> In the 1940's, the ENIAC was developed, the first electronic computer.

Since then, computer has come to mean an electronic computer.

Historical Digression

[David Alan Grier 2005]

1900's



ENIAC

Historical note:

The term "computer" used to refer to humans who did computation [Grier 2005 "When Computers Were Human"].

1700's: <click> Alexis Claude de Clairaut and friends divide the calculations <click> for the return of Halley's Comet.

1800's: <click> Charles Babbage's "Law of Errors" (two humans performing the same computation are likely to make the same error); <click> the Difference Engine was invented because Babbage was frustrated by limitations of (human) computers.

1900's: <click> WPA (Works Progress Administration) Mathematical Tables Project provided work during the depression, and other groups aided the <click> war effort (navigation, radar, bombs, Manhattan Project).

<click> In the 1940's, the ENIAC was developed, the first electronic computer.

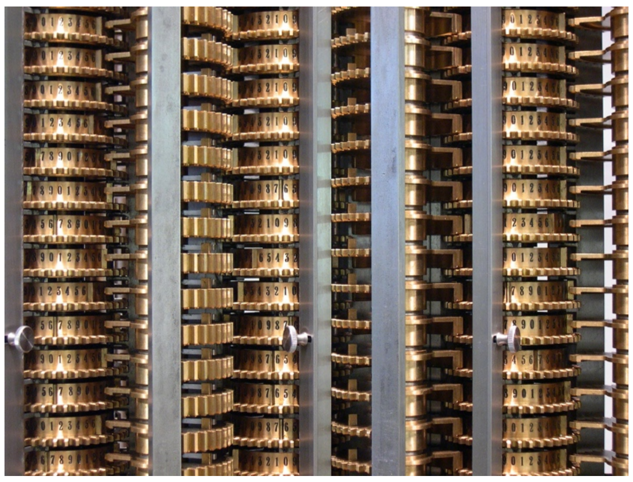
Since then, computer has come to mean an electronic computer.

Electronic

- Fast
- Deterministic
- Arithmetic

Human

- Slow
- Inconsistent & noisy
- ???


[Carsten Ullrich]

Electronic computers versus human computers. (This is a historic photograph from NACA/NASA.)

electronic: fast, deterministic (at arithmetic)
human: slow, inconsistent, yet still better at some things (which?)

The Human Advantage

- Perception
- Preference
- Creativity

I’m going to suggest a taxonomy... (every time I revisit this slide I change the taxonomy)

Humans better at:

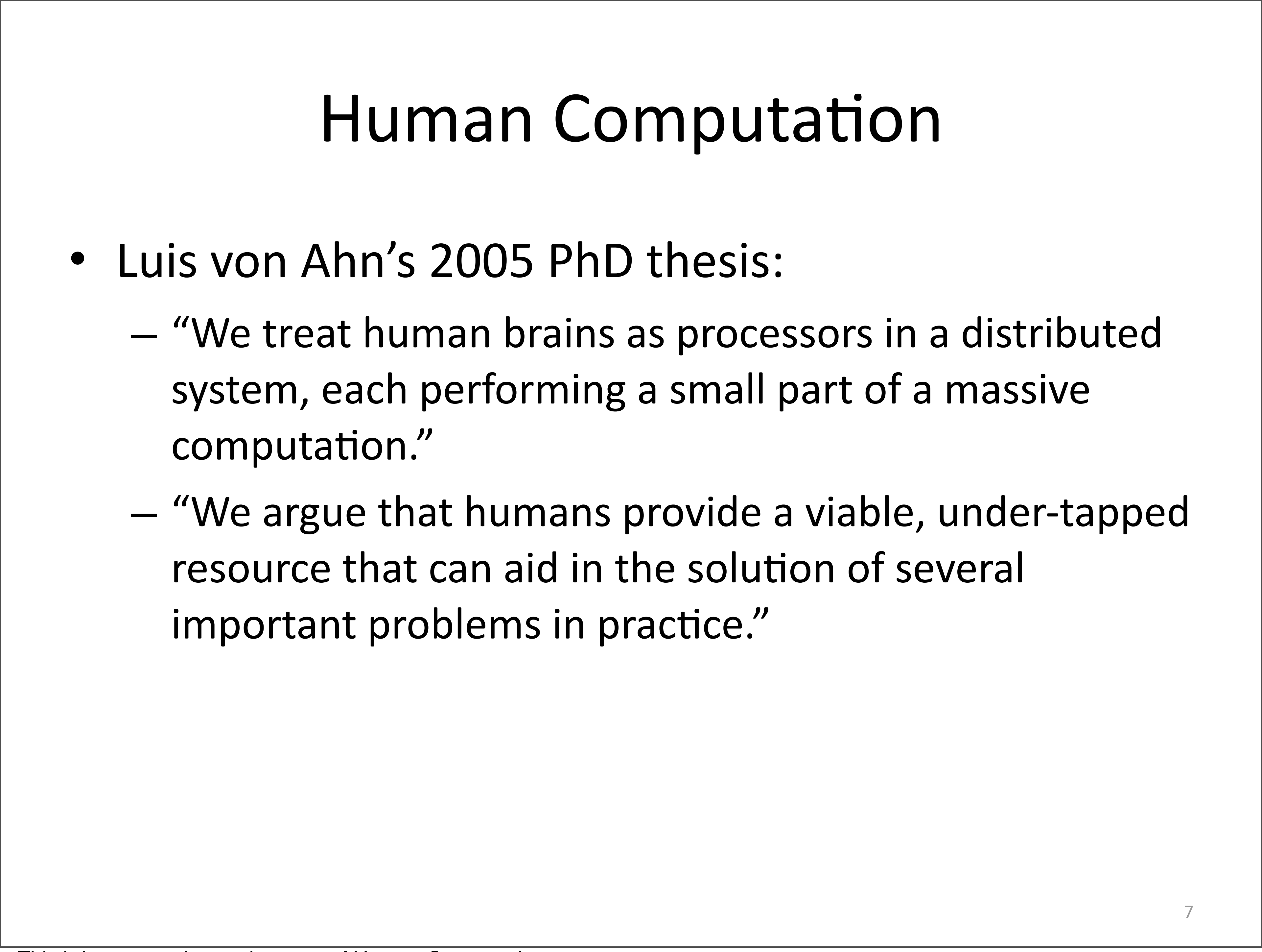
perception/comprehension: reconstructing information that wasn't captured at capture-time (as in a photo or surface scan) or constructing/inferring information that was never recorded (as in a sketch/recognizing emotions/labeling images) using knowledge humans naturally possess?

preference/aesthetic judgements: evaluate goodness ("beauty") for sorting or optimization (see Sims, Electric Sheep, Interactive Genetic Algorithm/Human-Based Genetic Algorithms, [Little 2009]/[Bernstein 2011])?

creativity

- search: finding images that go well together
- art projects like The Sheep Market [Koblin 2006]
- [Little 2009/10] for expanding text/jokes/shirt design
- [Yu and Nickerson 2011] for sketching chair designs (“Cooks or Cobblers”)
- [Bernstein 2011] for posing humans
- [Kittur 2011] for wikipedia... or wikipedia

Preference vs Creativity has a parallel to P vs NP (recognize versus generate)



Human Computation

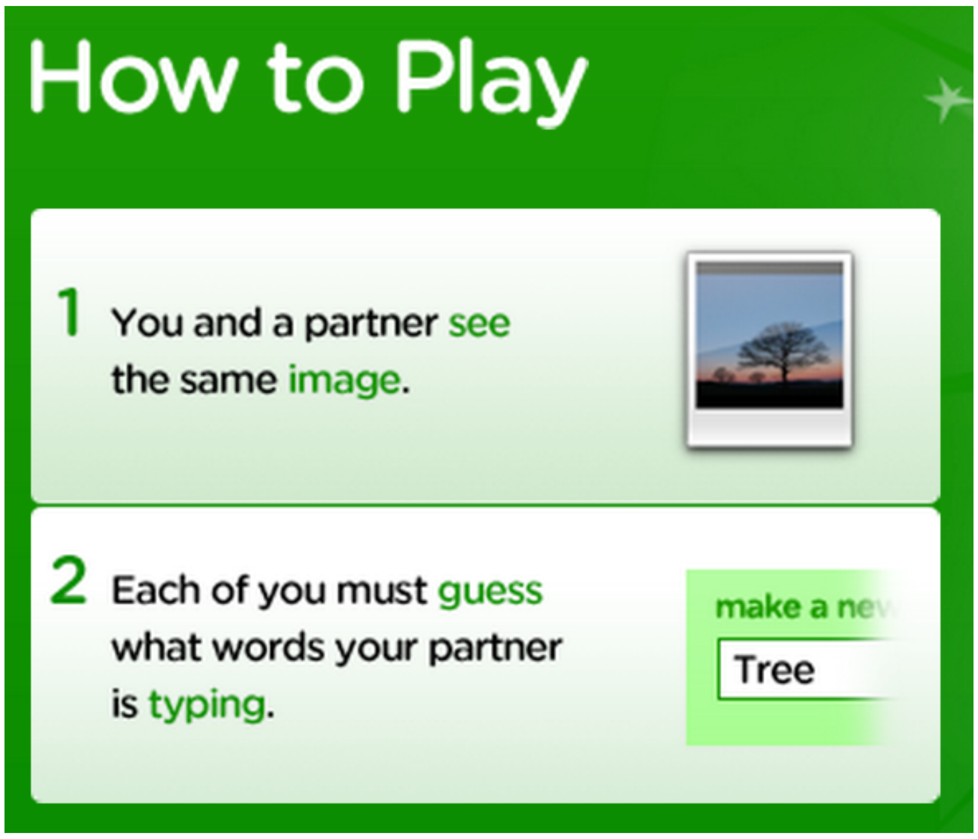
- Luis von Ahn's 2005 PhD thesis:
  - "We treat human brains as processors in a distributed system, each performing a small part of a massive computation."
  - "We argue that humans provide a viable, under-tapped resource that can aid in the solution of several important problems in practice."

This brings us to the modern use of Human Computation...

(This is independent of the earlier historical digression; that forgotten history was published simultaneously.)

Example 1: ESP Game

Concentrate...


How to Play

1 You and a partner see the same image.

2 Each of you must guess what words your partner is typing.

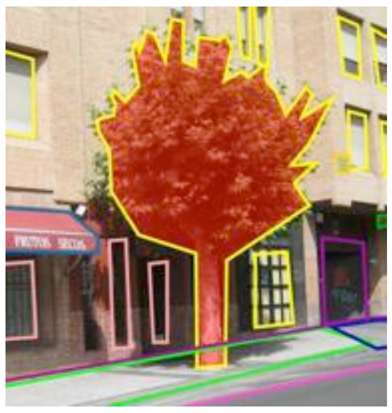
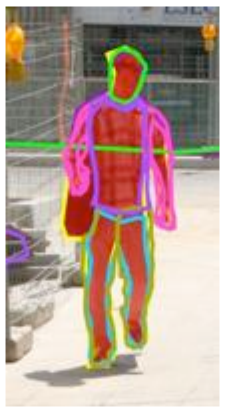
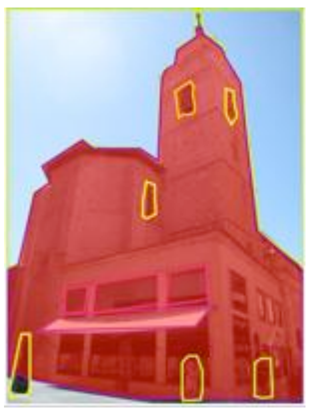
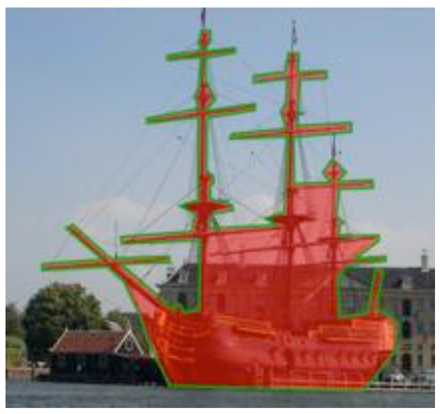
[von Ahn and Dabbish 2004]

[von Ahn and Dabbish 2004] http://www.gwap.com/gwap/gamesPreview/espgame/

Collecting data. Inspired by the Open Mind Initiative (1999+), a project for generating supervised machine learning datasets by crowdsourcing (a term which hadn't been invented yet (it was invented by Jeff Howe in a 2006 Wired article)).

Motivation: Fun!
Quality control for labels: matching words with another human
Main use: tagging images (good for google!)
Very successful; became Google Image Labeler (though recently shut down, in 2011).

Example 2: LabelMe

[Russel et al. 2005/2008]

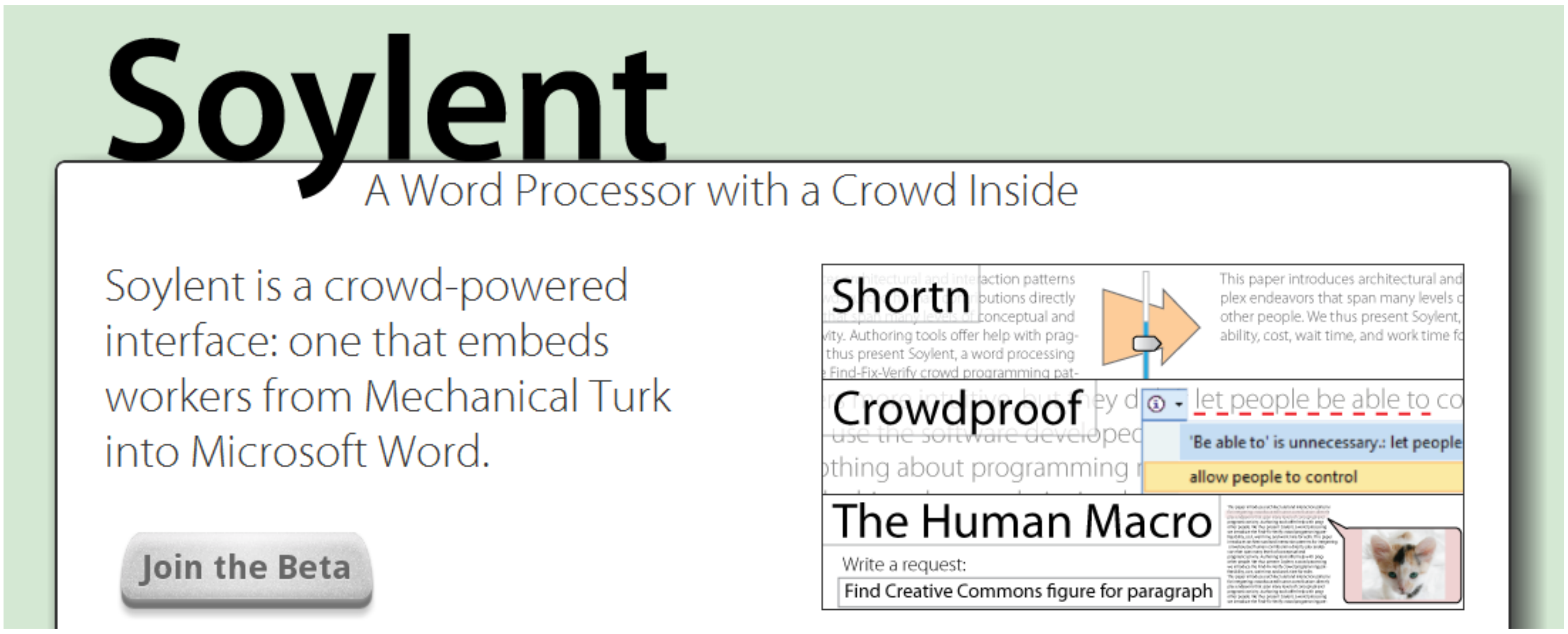
[Russel et al. 2005/2008] http://labelme.csail.mit.edu/

Collecting data.

Click out polygons around the boundary of individual objects and type a name.

Motivation: ?? researchers using the data please click out some objects. paid some people to do it (using Amazon's Mechanical Turk)
Quality control for labels: users can revise other users
Main use: dataset for supervised learning

Example 3:


Soylent
A Word Processor with a Crowd Inside
Soylent is a crowd-powered interface: one that embeds workers from Mechanical Turk into Microsoft Word.
Join the Beta
Shortn
Crowdproof
let people be able to co
'Be able to' is unnecessary.: let people
allow people to control
The Human Macro
Write a request:
Find Creative Commons figure for paragraph

[Bernstein et al. 2010]

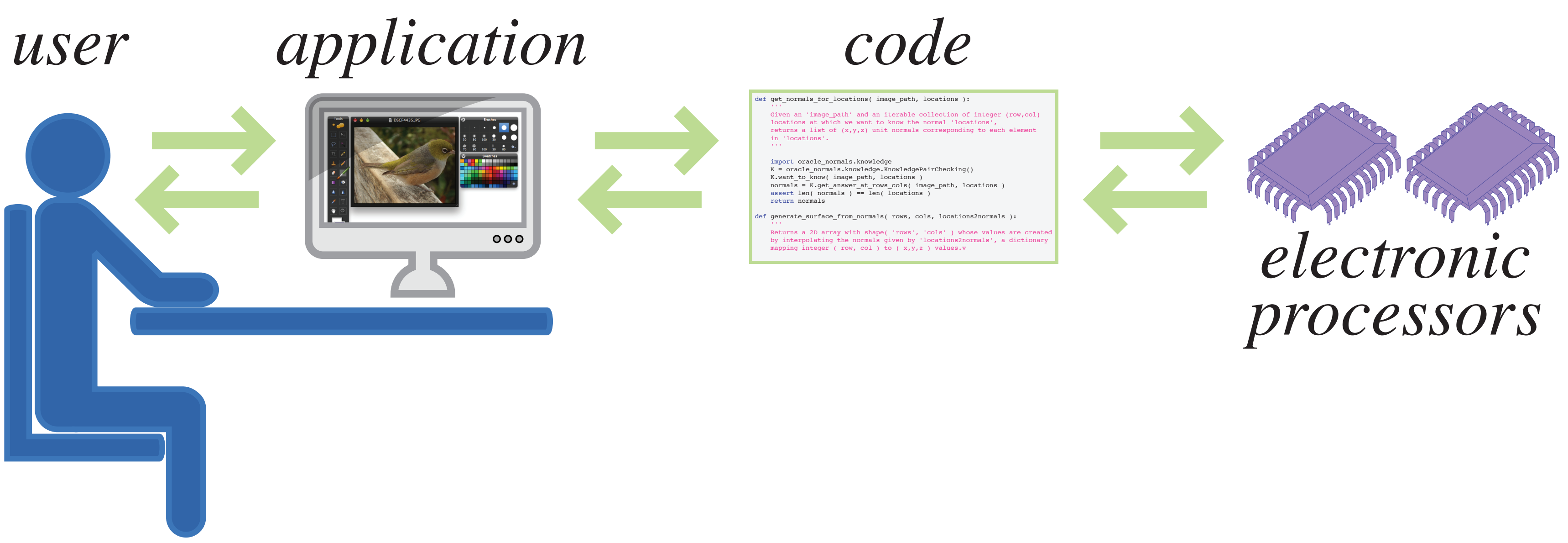
[Bernstein et al. 2010] http://projects.csail.mit.edu/soylent/

The HC is now online. (This isn't for use offline to generate supervised learning datasets.)

A Word plug-in that uses crowd contributions to perform document shortening, proofreading, and human-language macros.

Motivation: Payment
Quality control: Soylent introduced a human computation design pattern called Find-Fix-Verify that splits tasks into three HC stages: identifying the region of interest; performing the action; verifying the action.

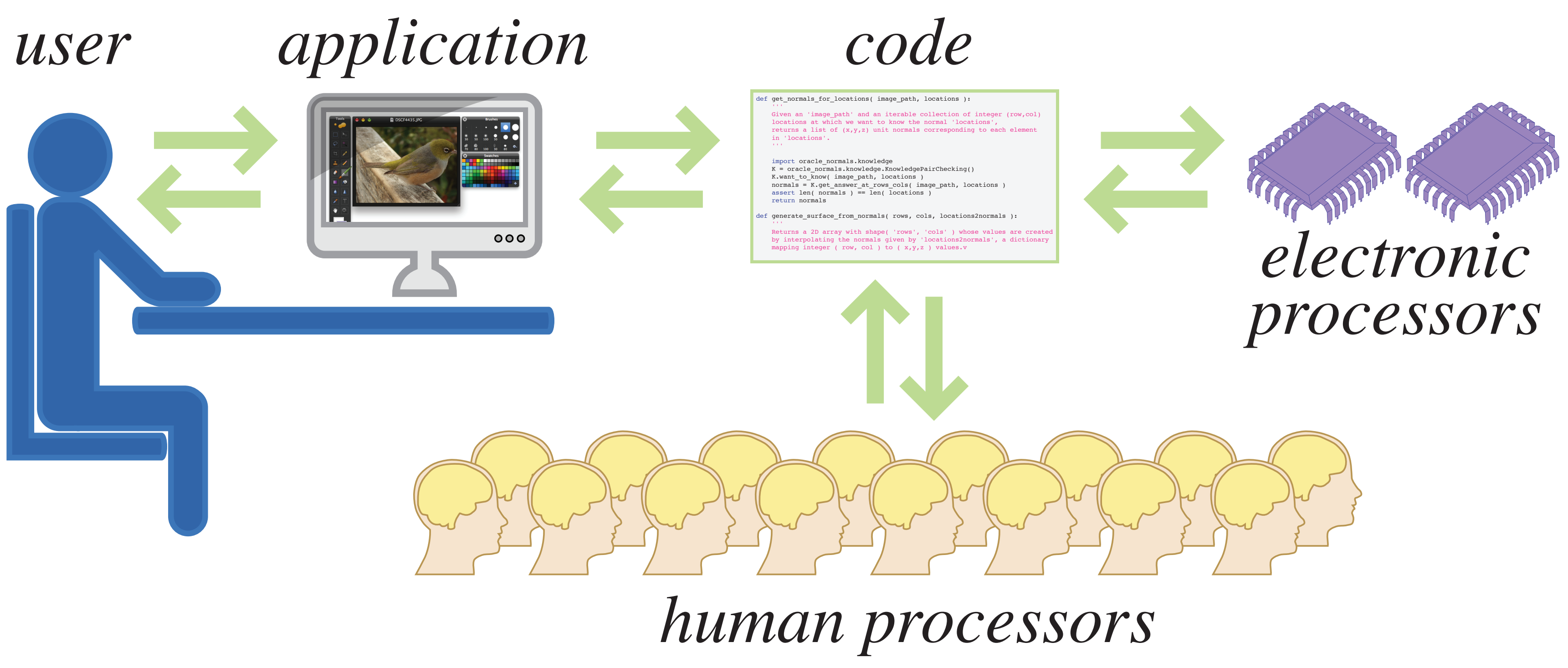
Computation



```
def get_normals_for_locations( image_path, locations ):
    '''
    Given an 'image_path' and an iterable collection of integer (row,col)
    locations at which we want to know the normal 'locations',
    returns a list of (x,y,z) unit normals corresponding to each element
    in 'locations'.
    '''

    import oracle_normals.knowledge
    K = oracle_normals.knowledge.KnowledgePairChecking()
    K.want_to_know( image_path, locations )
    normals = K.get_answer_at_rows_cols( image_path, locations )
    assert len( normals ) == len( locations )
    return normals

def generate_surface_from_normals( rows, cols, locations2normals ):
    '''
    Returns a 2D array with shape( 'rows', 'cols' ) whose values are created
    by interpolating the normals given by 'locations2normals', a dictionary
    mapping integer ( row, col ) to ( x,y,z ) values.v
```

Let’s compare the traditional and this new model of computation.
Here we have a model of (interactive) computation we are familiar with.
The user sits at a computer and uses an application.
The application is written in code, which runs on electronic processors.

Human Computation



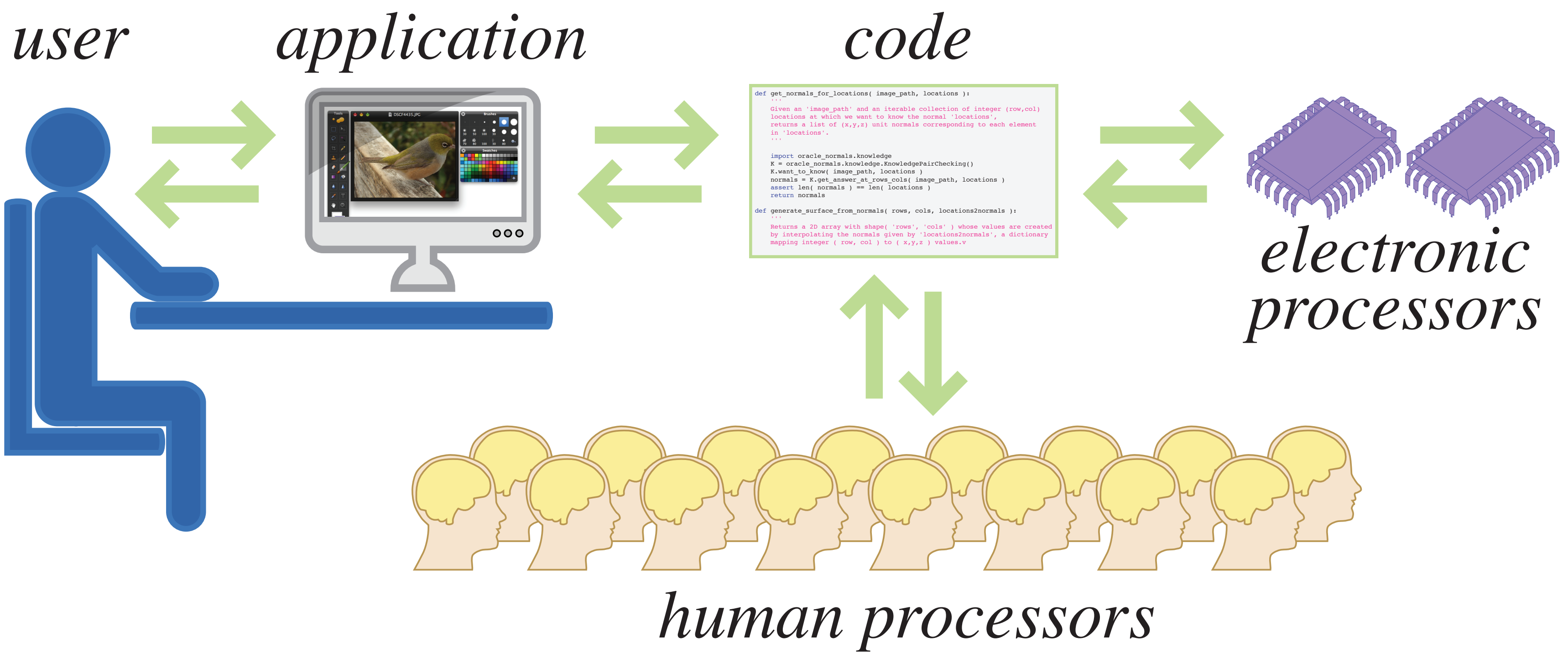
```
def get_normals_for_locations( image_path, locations ):
    '''
    Given an 'image_path' and an iterable collection of integer (row,col)
    locations at which we want to know the normal 'locations',
    returns a list of (x,y,z) unit normals corresponding to each element
    in 'locations'.
    '''

    import oracle_normals.knowledge
    K = oracle_normals.knowledge.KnowledgePairChecking()
    K.want_to_know( image_path, locations )
    normals = K.get_answer_at_rows_cols( image_path, locations )
    assert len( normals ) == len( locations )
    return normals

def generate_surface_from_normals( rows, cols, locations2normals ):
    '''
    Returns a 2D array with shape( 'rows', 'cols' ) whose values are created
    by interpolating the normals given by 'locations2normals', a dictionary
    mapping integer ( row, col ) to ( x,y,z ) values.v
```

And here is a model of human computation for interactive algorithms.

In this model, human processors are unskilled and isolated and there is high communication latency.
% Such a pool has been available (with an API) since 2005 via Amazon's Mechanical Turk platform, though there are others (SamaSource, txteagle).

We believe this model is especially relevant to computer graphics, because of humans' capabilities at visual perception.
In particular, I/we want to see human processors used for online, interactive applications.

Why?

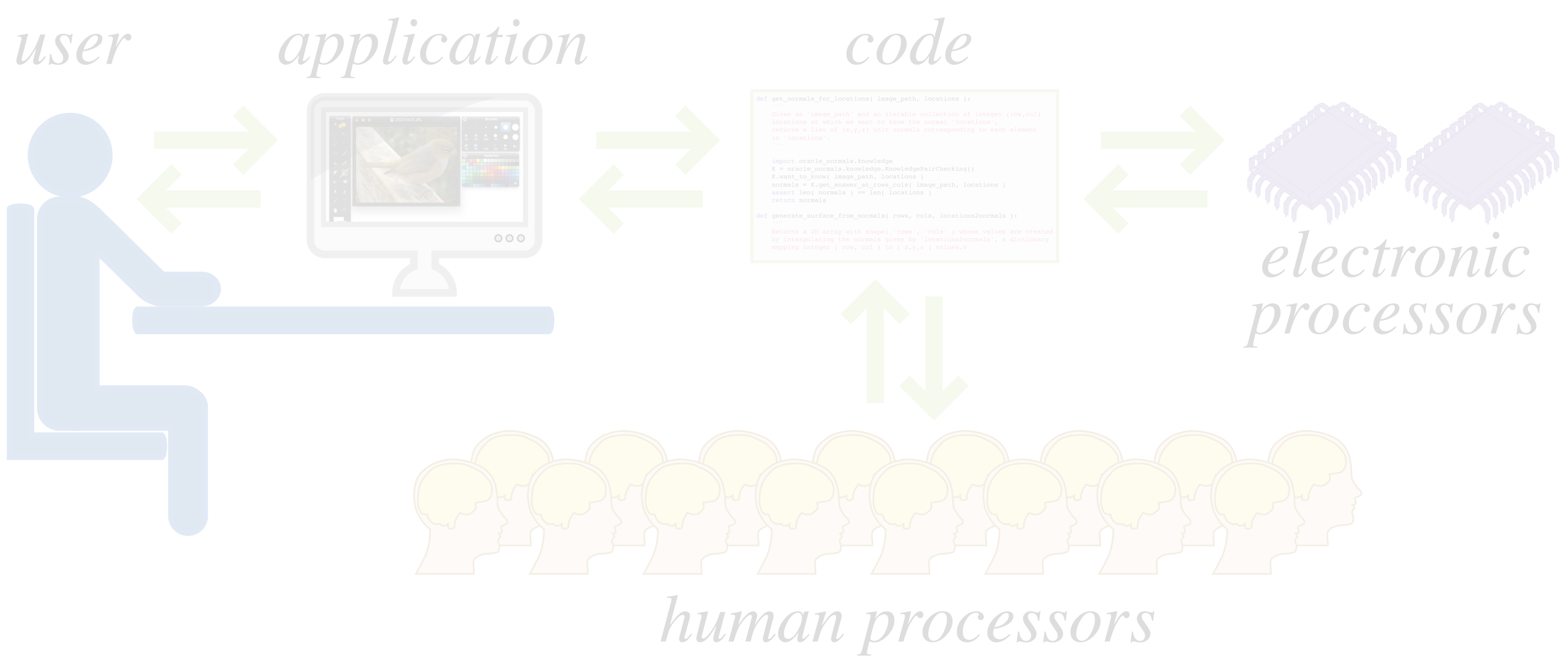


```
def get_normals_for_locations( image_path, locations ):
    '''
    Given an 'image_path' and an iterable collection of integer (row,col)
    locations at which we want to know the normal 'locations',
    returns a list of (x,y,z) unit normals corresponding to each element
    in 'locations'.
    '''

    import oracle_normals.knowledge
    K = oracle_normals.knowledge.KnowledgePairChecking()
    K.want_to_know( image_path, locations )
    normals = K.get_answer_at_rows_cols( image_path, locations )
    assert len( normals ) == len( locations )
    return normals

def generate_surface_from_normals( rows, cols, locations2normals ):
    '''
    Returns a 2D array with shape( 'rows', 'cols' ) whose values are created
    by interpolating the normals given by 'locations2normals', a dictionary
    mapping integer ( row, col ) to ( x,y,z ) values.v
```

1. Magic: make the impossible possible. For many problems we have no/there are no (low-dimensional) models, so machine learning cannot really help. We are many, many years from being able to solve such problems with computers.
2. Speed or Scalability. Via parallelization, jobs complete faster than “giving them to a human to solve.” You also gain access to a giant pool of workers.
3. Cheaper (or free). Human Processors are unskilled, so their time costs less. We advocate tasks that require no training. If you can make a game out of it, it could even be free, though this is an issue of incentives, and I won’t be talking about it much today.

Why?

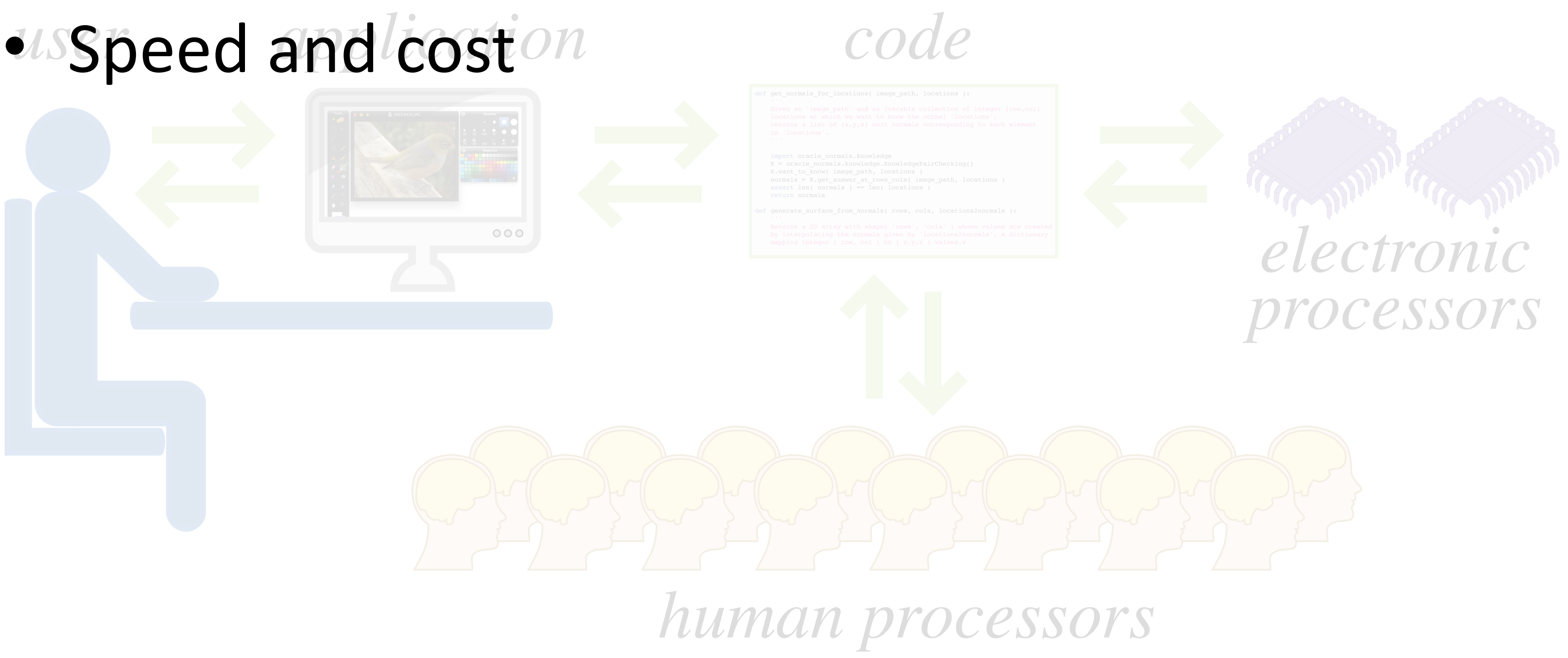
- Make the impossible possible



1. Magic: make the impossible possible. For many problems we have no/there are no (low-dimensional) models, so machine learning cannot really help. We are many, many years from being able to solve such problems with computers.
2. Speed or Scalability. Via parallelization, jobs complete faster than "giving them to a human to solve." You also gain access to a giant pool of workers.
3. Cheaper (or free). Human Processors are unskilled, so their time costs less. We advocate tasks that require no training. If you can make a game out of it, it could even be free, though this is an issue of incentives, and I won't be talking about it much today.

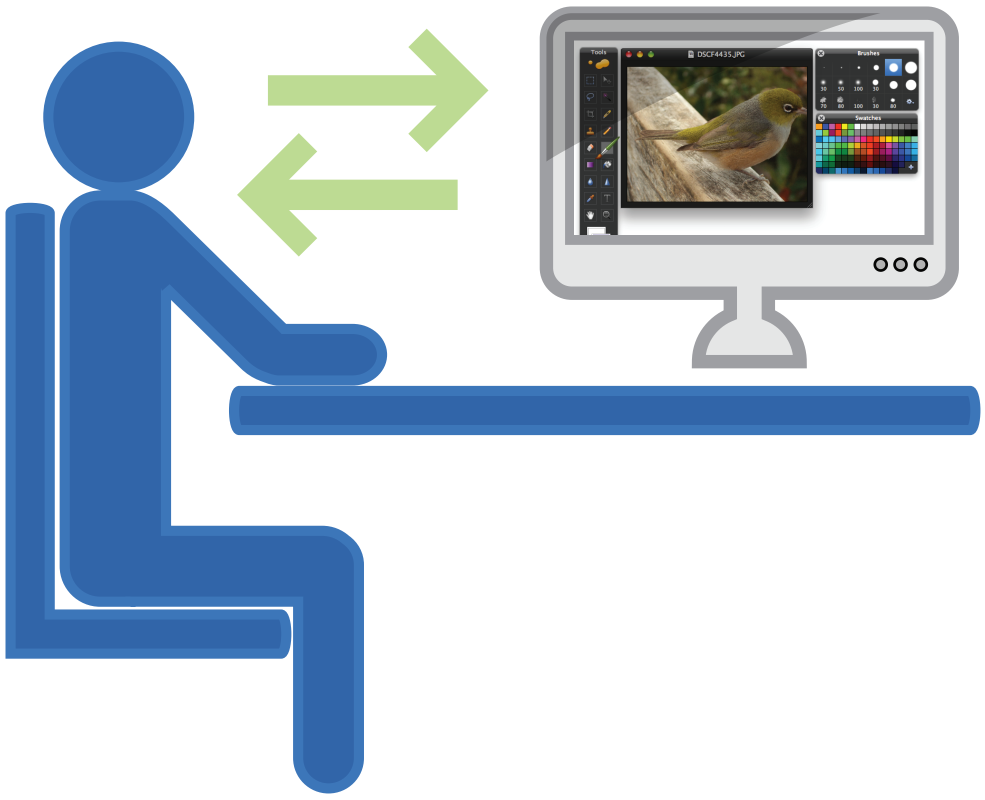
Why?

- Make the impossible possible
- Speed and cost



1. Magic: make the impossible possible. For many problems we have no/there are no (low-dimensional) models, so machine learning cannot really help. We are many, many years from being able to solve such problems with computers.
2. Speed or Scalability. Via parallelization, jobs complete faster than "giving them to a human to solve." You also gain access to a giant pool of workers.
3. Cheaper (or free). Human Processors are unskilled, so their time costs less. We advocate tasks that require no training. If you can make a game out of it, it could even be free, though this is an issue of incentives, and I won't be talking about it much today.

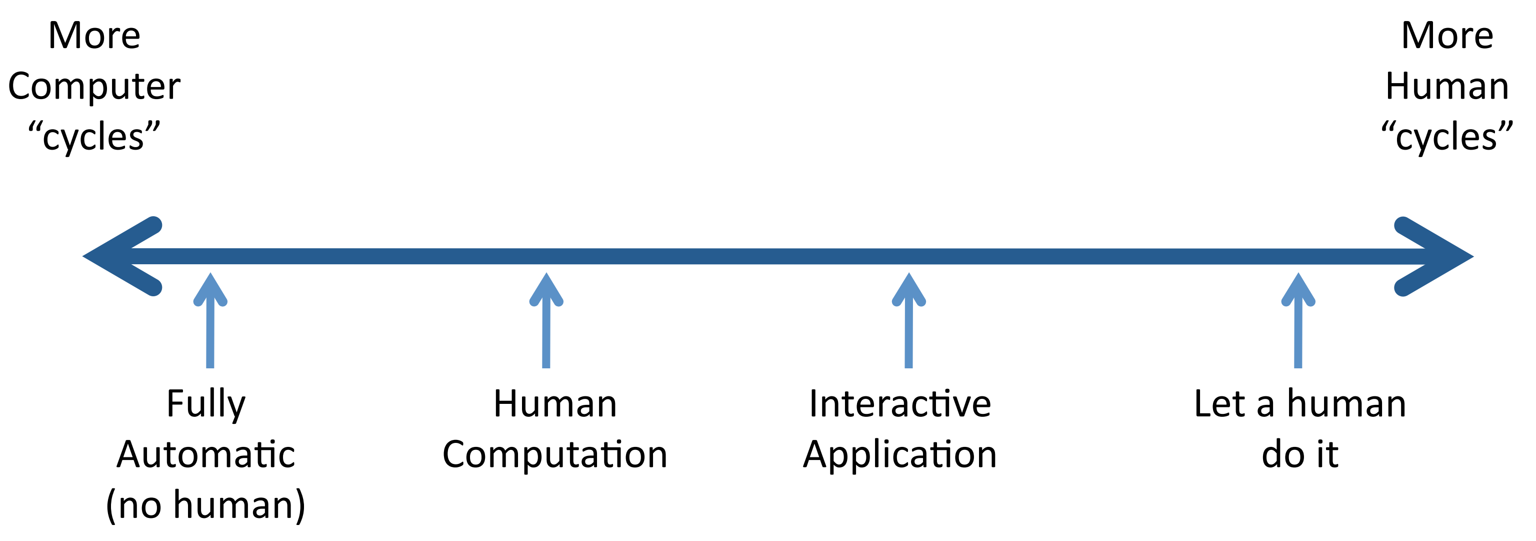
Humans using Computers



There is always a tradeoff between how much work the human does and how much work the computer does.

Range of Solutions

- How much human and how much computer is involved?

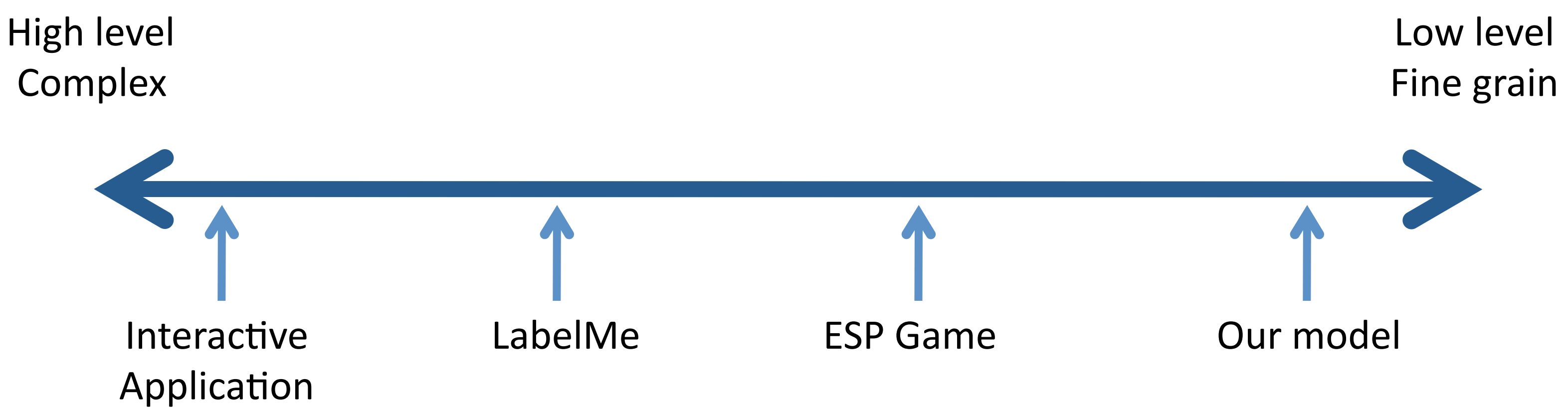


There are a range of solutions, which you can think of as a slider in terms of How much human and how much computer is involved.

<describe axis>

Type of Human Cycles

- You can also think of the type of activity the human does.

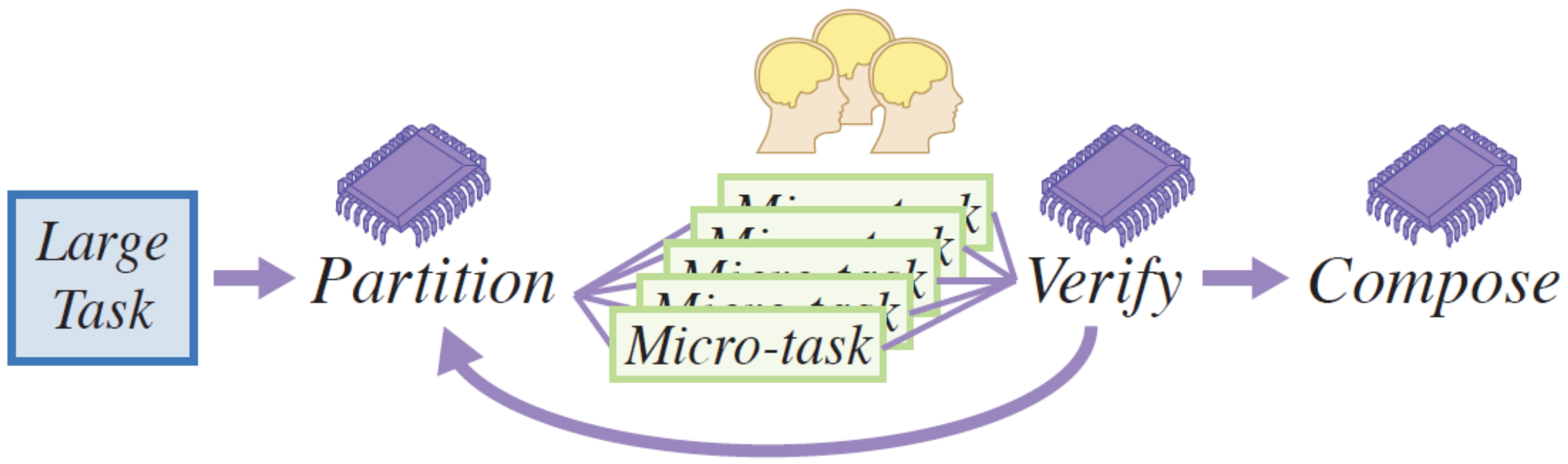


You can also think of the type of activity the human does.
<describe axis>

When dealing with graphics problems the key characteristic that still provides an advantage to humans is visual perception.
In our model, HC tasks are based on visual perceptual queries
- No training or skill needed—any human has good visual perception. Simple tasks help keep cost low, since we don't have to pay for training (up front or amortized).
- No dependency (between tasks)—compared to typical distributed processing, HPs execute few operations per second and have high latency.
- Highly parallel—in theory, with perfect parallelism, our algorithms would take 3 minutes to complete.

So, our model can be summarized as massive parallelism with extremely simple (training-free/instantaneous) visual queries.

Algorithm Design Pattern



This is the design pattern we use for our algorithms.

<describe diagram>

%We use Amazon's Mechanical Turk, online since 2005, which lets you advertise a job (brief description, amount of payment, time estimate) and has a large pool of workers (tens or hundreds of thousands).
%Has an API, so you can program it; there are others (SamaSource, txteagle, Farmville?).

The Question We Ask

- What is the minimum amount of information a human could provide in order to solve the original problem?
- Rephrase the algorithm in terms of the smallest piece of information that without it the problem could not be solved.

When designing an algorithm with HC inside, the question we ask is...

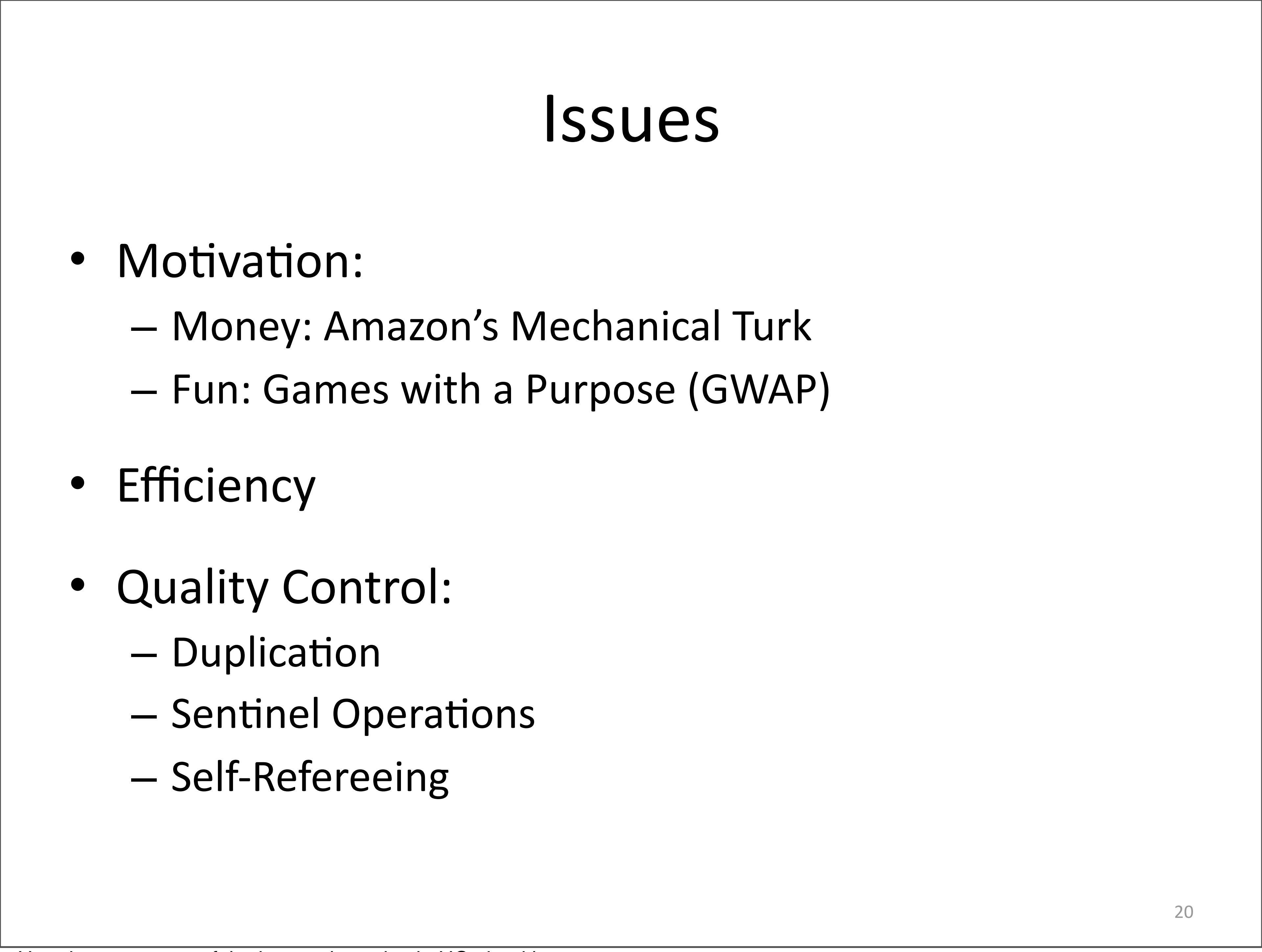
Three Example Algorithms

- Given an image, create
  - depth layers
  - a normal map
  - a bilateral symmetry map

I will show three micro perceptual human computation algorithms:

- recovering depth layers from a photograph (useful for object insertion/removal, dehazing, depth of field, retargeting)
- normals (useful for relighting or surface reconstruction)
- bilateral symmetry (useful for edit propagation, retargeting)

These algorithms are intended for use in, say, Photoshop. The HC must be online—inside the algorithm—because input images are too high-dimensional.



Issues

- Motivation:
  - Money: Amazon’s Mechanical Turk
  - Fun: Games with a Purpose (GWAP)
- Efficiency
- Quality Control:
  - Duplication
  - Sentinel Operations
  - Self-Refereeing

Here is a summary of the issues that arise in HC algorithms.

Incentives (money or fun)

“money, love, or glory” taxonomy due to Thomas W. Malone, Robert Laubacher, and Chrysanthos Dellarocas (MIT Center for Collective Intelligence); motivations for network collaboration

- You can pay human processors with an online labor market such as Amazon’s Mechanical Turk. Mechanical Turk has been online since 2005, and lets you advertise a job (brief description, amount of payment, time estimate) and has a large pool of workers (tens or hundreds of thousands). Has an API, so you can program it; there are others (SamaSource, txteagle, Farmville?).

- If you can make a game out of your human computation, it could become free, though this is an issue of incentives, and I won’t be talking about it much today. You can think of it as the “Inverse Karate Kid” problem. If you tackle a worthwhile cause, such as protein folding in FoldIt, you can also get people to participate for free. (You could also find a way to force people to do your work, like (re)Captchas.)

Efficiency means using as little HC as possible. HC is slow (compared to electronic, in terms of carrying out operation and in terms of latency), so this is typically the bottleneck. For example, at what granularity do we partition the problem?

Quality control is important!

humans are: noisy/inconsistent/non-deterministic. depending on their motivations, they may cheat.

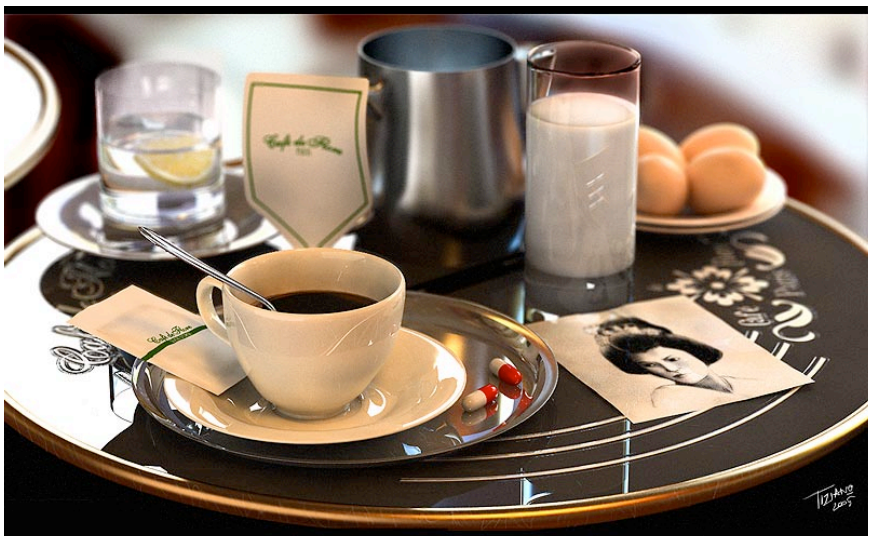
humans have internal biases (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

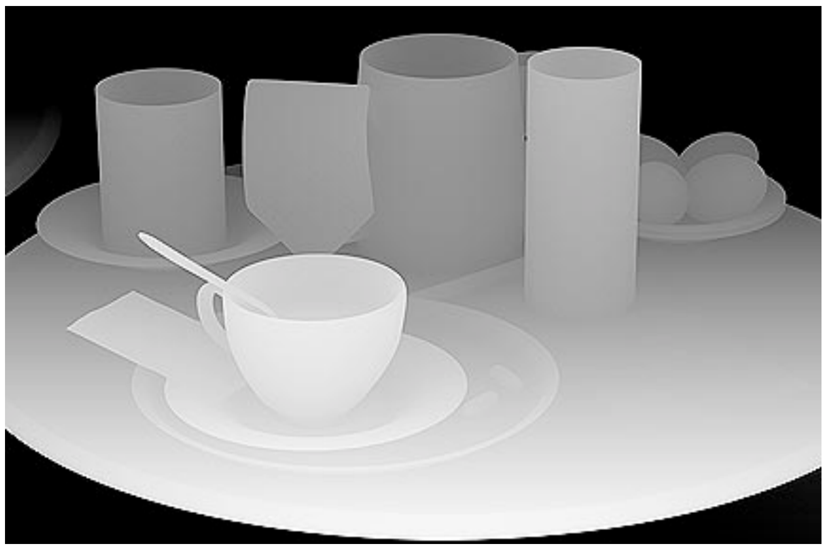
It’s not an algorithm if there’s a researcher in the loop! We are only interested in human computation that runs on a pool of unskilled, isolated, and oblivious human processors. This means no expert user inside the algorithm accepting and rejecting human computation.

- duplication (multiple HPs or same HP multiple times—used in perceptual experiments and, for example, [Cole et al. 2009])
- sentinel operations (using known answers or “gold data”)
- self-refereeing [Little et al. 2010b; Bernstein et al. 2010] --- increases amount of HC and adds data dependency.
- [Quinn and Bederson 2011] describe more variations

- We use Amazon's Mechanical Turk.
- We opt for massive parallelism with extremely simple visual queries in our examples.
- For our quality control, we use both kinds of duplication and sentinel operations.

Algorithm 1: Depth Layers





Depth (distance from the camera) is an important cue that can assist image manipulations (insertion and removal of objects, retargeting, adding depth-of-field effects, de-hazing, etc.)

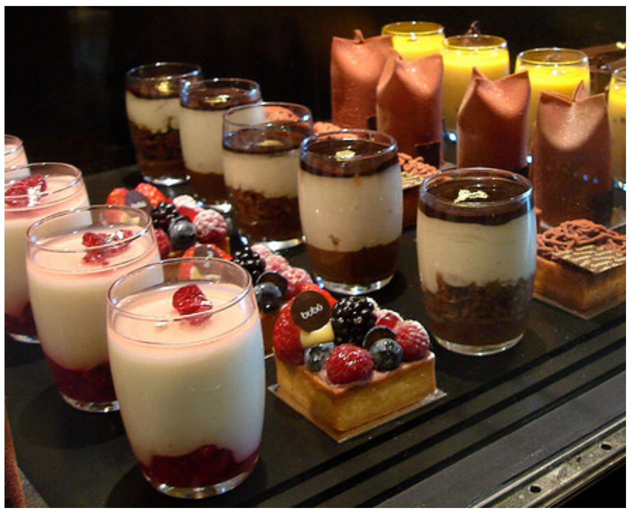
Today, you could use a depth camera, but you may not have one, you may already have your image, or your scene may not be applicable due to depth camera limitations.

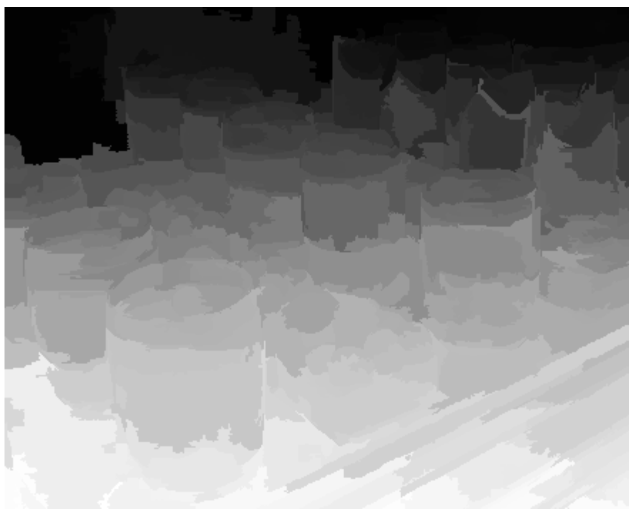
Calculate Depth of a Given Image?

We aim to be more robust than automatic techniques [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009].
For example, Make3D [Saxena et al. 2009] seems to assume that...
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Calculate Depth of a Given Image?

- Automatic methods:
  - Depth increases in the up direction
  - Color similarity implies depth similarity

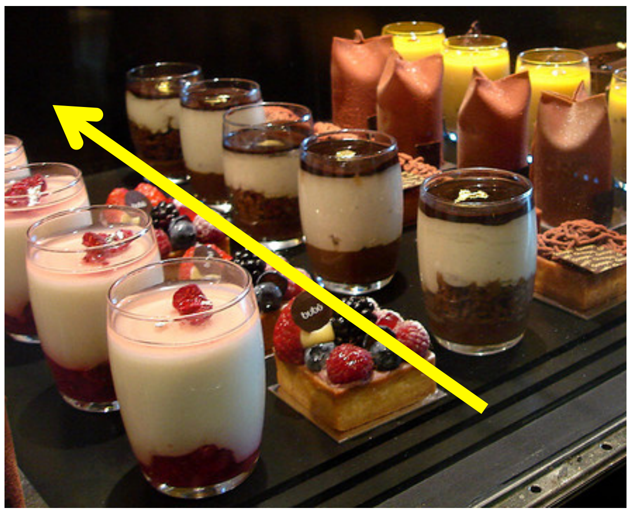
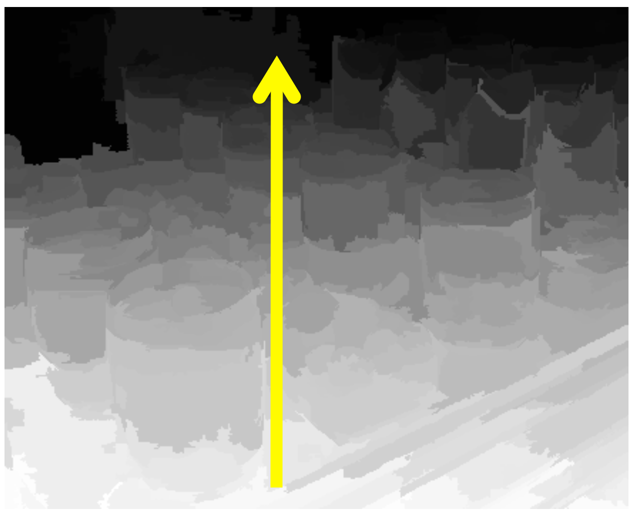




We aim to be more robust than automatic techniques [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009].
For example, Make3D [Saxena et al. 2009] seems to assume that...
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Calculate Depth of a Given Image?

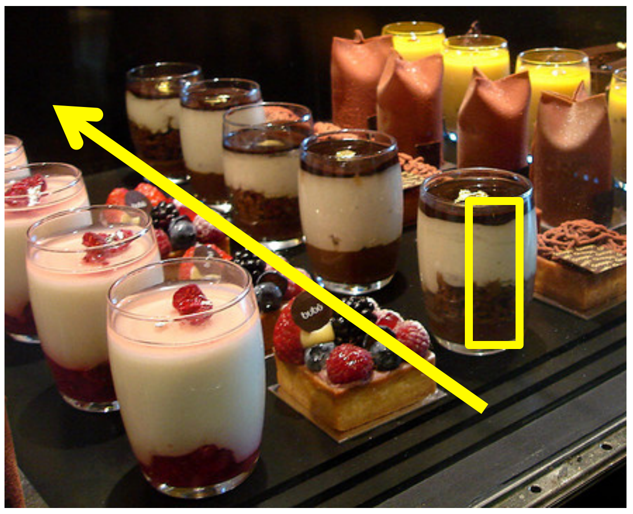
- Automatic methods:
  - Depth increases in the up direction
  - Color similarity implies depth similarity
- Not always correct

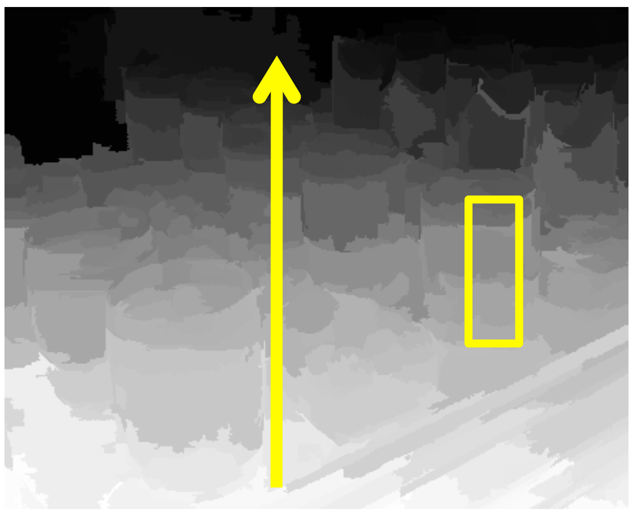



We aim to be more robust than automatic techniques [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009].
For example, Make3D [Saxena et al. 2009] seems to assume that...
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Calculate Depth of a Given Image?

- Automatic methods:
  - Depth increases in the up direction
  - Color similarity implies depth similarity
- Not always correct

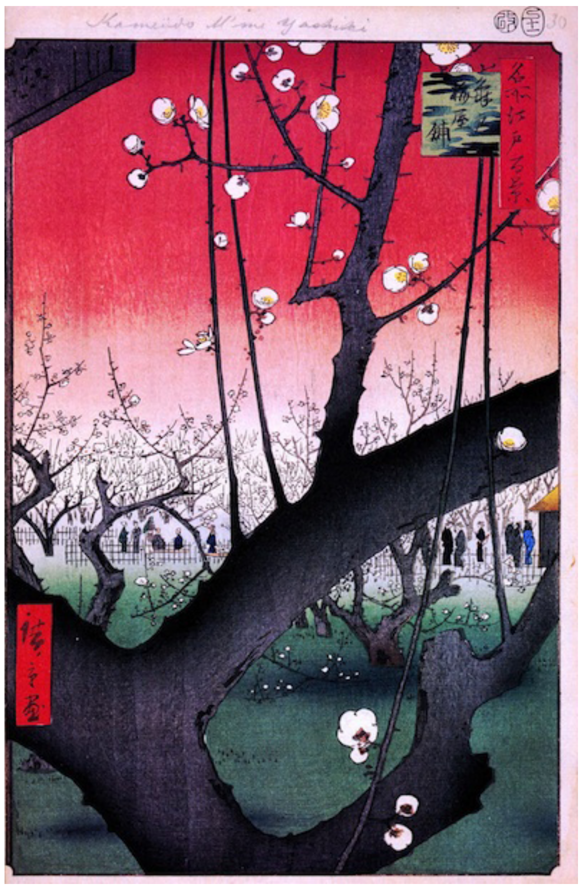




We aim to be more robust than automatic techniques [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009].
For example, Make3D [Saxena et al. 2009] seems to assume that...
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Calculate Depth of a Given Image?

- Automatic methods:
  - Depth increases in the up direction
  - Color similarity implies depth similarity
- Not always correct
- Some images are very challenging (art)



[Hiroshige]

We aim to be more robust than automatic techniques [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009].
For example, Make3D [Saxena et al. 2009] seems to assume that...
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

There are some manual techniques one could use, but they require a trained user: [Oh et al. (including Durand) 2001; Ventura et al. 2009; Sykora et al. 2010]

Micro Task?

So what should our micro-task be?

<click>
<click>
<click>
We can compute image patches using a superpixel-type algorithm which divides the image into small pieces.

I will show the result of using this third one later.

Micro Task?

- Ask "what is the depth of the pixel?"
  - Too fine, can be ambiguous

So what should our micro-task be?

<click>
<click>
<click>
We can compute image patches using a superpixel-type algorithm which divides the image into small pieces.

I will show the result of using this third one later.

Micro Task?

- Ask "what is the depth of the pixel?"
  - Too fine, can be ambiguous
- Ask "what is the depth of an object?"
  - Segmentation is too complex

So what should our micro-task be?

<click>
<click>
<click>
We can compute image patches using a superpixel-type algorithm which divides the image into small pieces.

I will show the result of using this third one later.

Micro Task?

- Ask "what is the depth of the pixel?"
  - Too fine, can be ambiguous
- Ask "what is the depth of an object?"
  - Segmentation is too complex
- Ask "what is the depth of a patch in the image?"
  - Getting better... but humans are not good at assessing absolute depth

So what should our micro-task be?

<click>
<click>
<click>
We can compute image patches using a superpixel-type algorithm which divides the image into small pieces.

I will show the result of using this third one later.

Relative Ordering

- Ask "which is closer" on neighboring patches?

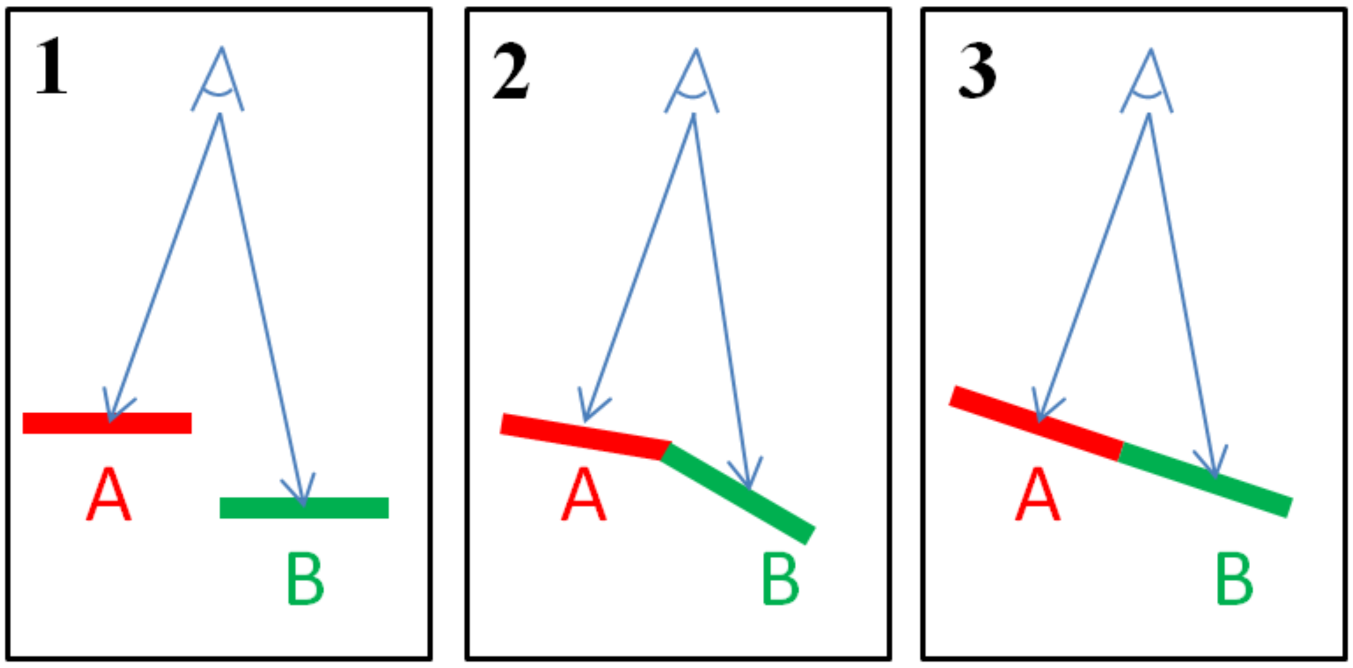
This is reliable ([Koenderink 2001]).

However, it's still ambiguous:
1 depth jump between A and B
2 non-smooth depth change between A and B
3 smooth depth change between A and B

Our depth layer task matches 1.
I will also show a comparison to a continuous version of this question.

Relative Ordering

- Ask “which is closer” on neighboring patches?
  - Reliable, but not well-defined. A is closer than B:

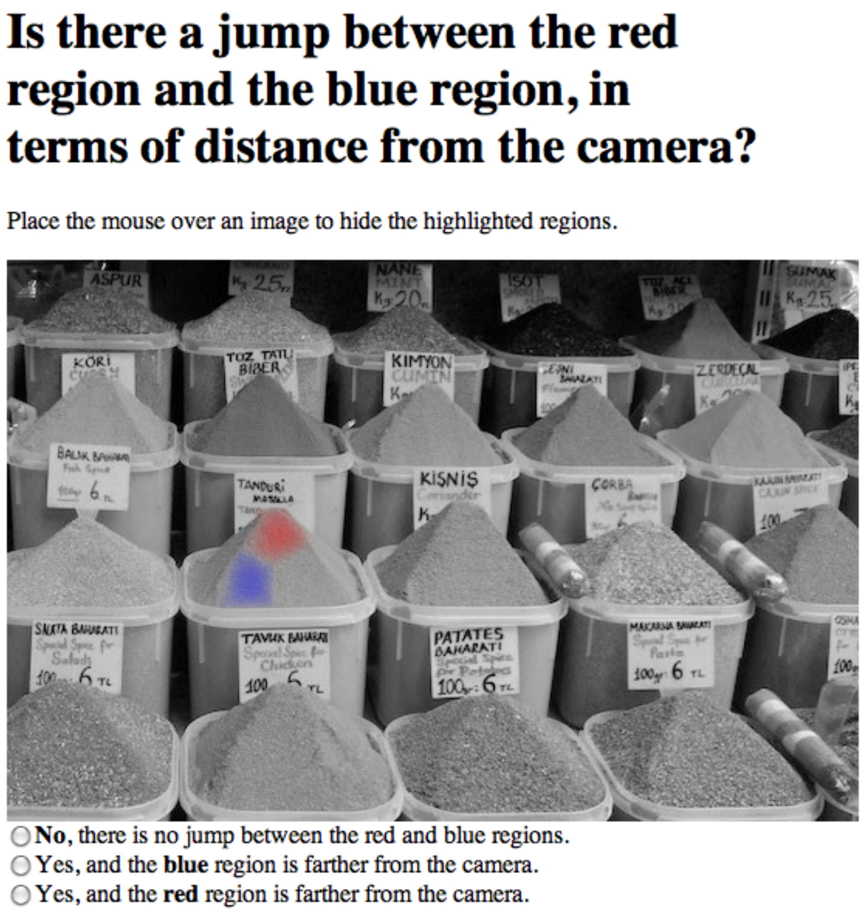


This is reliable ([Koenderink 2001]).

However, it’s still ambiguous:
1 depth jump between A and B
2 non-smooth depth change between A and B
3 smooth depth change between A and B

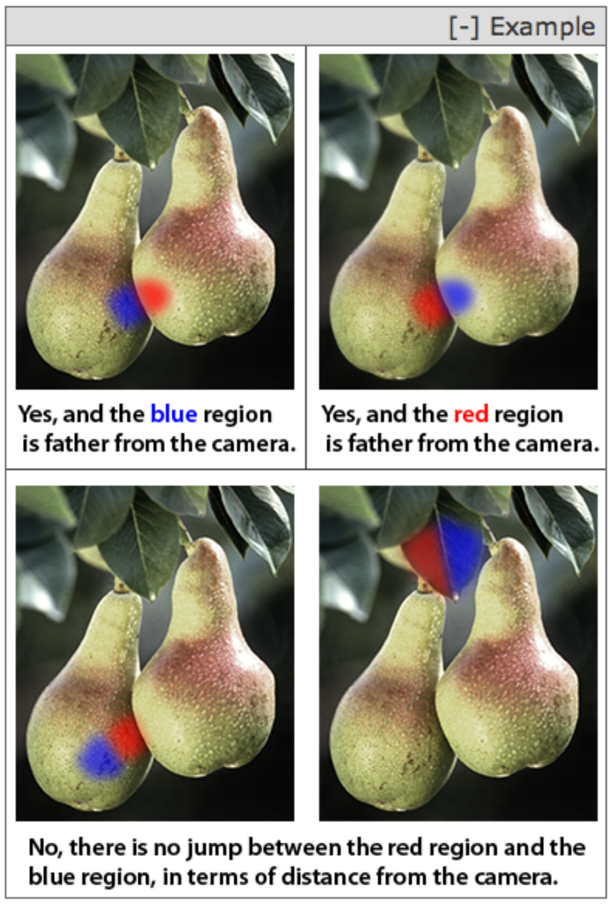
Our depth layer task matches 1.
I will also show a comparison to a continuous version of this question.

Our Micro Task


Is there a jump between the red region and the blue region, in terms of distance from the camera?

Place the mouse over an image to hide the highlighted regions.

- ○ **No**, there is no jump between the red and blue regions.
- ○ Yes, and the **blue** region is farther from the camera.
- ○ Yes, and the **red** region is farther from the camera.


[-] Example

Yes, and the blue region is father from the camera.

Yes, and the red region is father from the camera.

No, there is no jump between the red region and the blue region, in terms of distance from the camera.

Note the static example in the corner. That’s it, there is no other training.

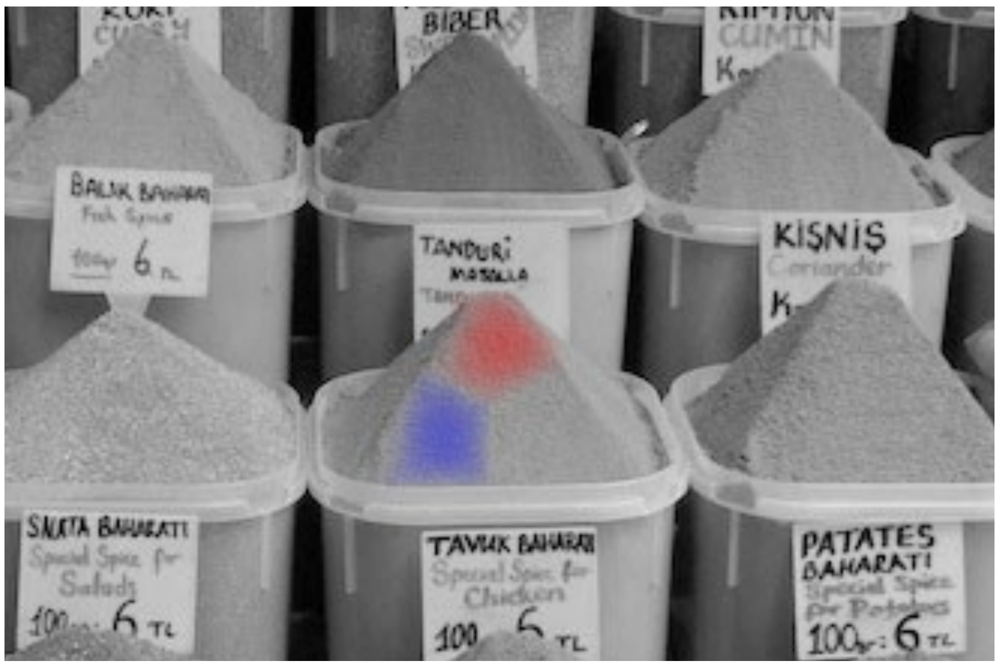
Guidelines for Choosing Tasks

- Task must be simple (instantaneous)
- Task must be specific (well-defined)
- Task must be reliable (humans can do it)

- Near-instantaneous.
- Well-defined so we can program with it. A metaphor is sampling the real-world with a temperature sensor; we get a number back, which we can program with.
- We also want this task to be something humans can actually do, not just something humans think they can do (like absolute depth).

Of course, it must also be something computers can't do.

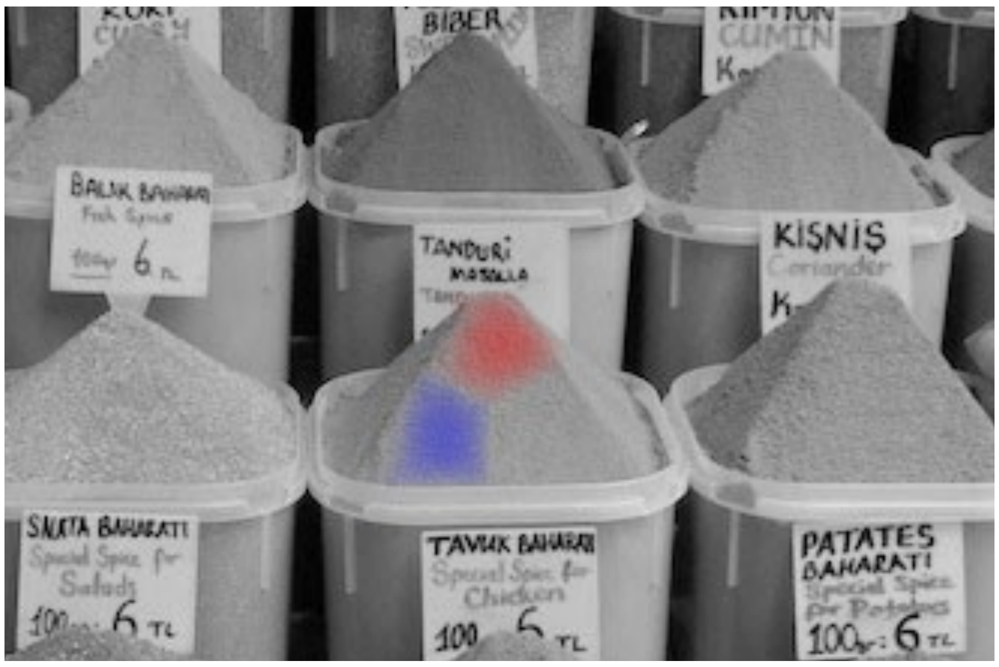
Combining



The relative depth ordering provides offsets of -1, 0, or 1 between adjacent regions in the image.

To reconstruct a continuous depth map we solve a <click> Laplace equation with derivative constraints of -1, 0, or 1 across region boundaries.

Combining



- Laplace equation $\Delta f = 0$ with constraints

The relative depth ordering provides offsets of -1, 0, or 1 between adjacent regions in the image.

To reconstruct a continuous depth map we solve a <click> Laplace equation with derivative constraints of -1, 0, or 1 across region boundaries.

Algorithm

```
DEPTH-LAYERS(image $I$, sentinel queries $S$)
Segment $I$ into regions (using mean-shift and SLIC)
Insert all pairs of neighboring regions into $Q$
**loop in parallel until** each pair has been visited $N$ times
    Gather $K$ random pairs from $Q$
    Gather $M$ random pairs from $S$
    **for** each pair: Build the visual query & Duplicate it
    Mix the $2K + 2M$ queries
    $results$ = send all queries to an HP
    **if** $average(consistent(results)) \geq 0.75$
        **for** each pair
            Add consistent results to the list of votes
            Increment #visited
**for** each pair of neighboring regions
    $final\_result = majority$(list of votes)
Solve the Laplace equation to construct a depth map
```

This is what an HC algorithm looks like in pseudocode.

<click>
We partition the task into superpixels
<click>
In our quality control setup, we dispatch batches of 20 micro-tasks at a time to human processors.
Each batch is composed of 6 queries whose results we need plus 4 sentinel queries whose results we know. All queries appear twice in each batch in random order, so we can check an HP's output for internal consistency as well as for agreement on the sentinel tasks.

Note that we use sentinel queries from the same image, so the user must answer ~10–20 queries himself to get the results of the crowd. This is to prevent detection of the sentinel tasks. If many instances of the algorithm were run at once, the queries could be intermingled and then sentinel answers wouldn't have to come from the same image.
<click>
We send each batch to 3 different HPs for a median vote.
<click>
We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>
Finally, we combine the input by solving a laplace equation.

Algorithm

Partition

```
DEPTH-LAYERS(image $I$, sentinel queries $S$)
1   Segment $I$ into regions (using mean-shift and SLIC)
2   Insert all pairs of neighboring regions into $Q$
3   loop in parallel until each pair has been visited $N$ times
4       Gather $K$ random pairs from $Q$
5       Gather $M$ random pairs from $S$
6       for each pair: Build the visual query & Duplicate it
7       Mix the $2K + 2M$ queries
8       $results$ = send all queries to an HP
9       if $average(consistent(results)) \geq 0.75$
10          for each pair
11              Add consistent results to the list of votes
12              Increment #visited
13  for each pair of neighboring regions
14      $final\_result = majority$(list of votes)
15  Solve the Laplace equation to construct a depth map
```

This is what an HC algorithm looks like in pseudocode.

<click>
We partition the task into superpixels
<click>
In our quality control setup, we dispatch batches of 20 micro-tasks at a time to human processors.
Each batch is composed of 6 queries whose results we need plus 4 sentinel queries whose results we know. All queries appear twice in each batch in random order, so we can check an HP's output for internal consistency as well as for agreement on the sentinel tasks.

Note that we use sentinel queries from the same image, so the user must answer ~10–20 queries himself to get the results of the crowd. This is to prevent detection of the sentinel tasks. If many instances of the algorithm were run at once, the queries could be intermingled and then sentinel answers wouldn't have to come from the same image.
<click>
We send each batch to 3 different HPs for a median vote.
<click>
We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>
Finally, we combine the input by solving a laplace equation.

Algorithm

Partition

Quality Control Setup

```
DEPTH-LAYERS(image $I$, sentinel queries $S$)
1   Segment $I$ into regions (using mean-shift and SLIC)
2   Insert all pairs of neighboring regions into $Q$
3   loop in parallel until each pair has been visited $N$ times
4       Gather $K$ random pairs from $Q$
5       Gather $M$ random pairs from $S$
6       for each pair: Build the visual query & Duplicate it
7       Mix the $2K + 2M$ queries
8       $results$ = send all queries to an HP
9       if $average(consistent(results)) \geq 0.75$
10          for each pair
11              Add consistent results to the list of votes
12              Increment #visited
13  for each pair of neighboring regions
14      $final\_result = majority$(list of votes)
15  Solve the Laplace equation to construct a depth map
```

This is what an HC algorithm looks like in pseudocode.

<click>
We partition the task into superpixels
<click>
In our quality control setup, we dispatch batches of 20 micro-tasks at a time to human processors.
Each batch is composed of 6 queries whose results we need plus 4 sentinel queries whose results we know. All queries appear twice in each batch in random order, so we can check an HP's output for internal consistency as well as for agreement on the sentinel tasks.

Note that we use sentinel queries from the same image, so the user must answer ~10–20 queries himself to get the results of the crowd. This is to prevent detection of the sentinel tasks. If many instances of the algorithm were run at once, the queries could be intermingled and then sentinel answers wouldn't have to come from the same image.
<click>
We send each batch to 3 different HPs for a median vote.
<click>
We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>
Finally, we combine the input by solving a laplace equation.

Algorithm

Partition

Quality Control Setup

Dispatch

```
DEPTH-LAYERS(image $I$, sentinel queries $S$)
1   Segment $I$ into regions (using mean-shift and SLIC)
2   Insert all pairs of neighboring regions into $Q$
3   loop in parallel until each pair has been visited $N$ times
4       Gather $K$ random pairs from $Q$
5       Gather $M$ random pairs from $S$
6       for each pair: Build the visual query & Duplicate it
7       Mix the $2K + 2M$ queries
8       $results$ = send all queries to an HP
9       if $average(consistent(results)) \geq 0.75$
10          for each pair
11              Add consistent results to the list of votes
12              Increment #visited
13  for each pair of neighboring regions
14      $final\_result = majority$(list of votes)
15  Solve the Laplace equation to construct a depth map
```

This is what an HC algorithm looks like in pseudocode.

<click>
We partition the task into superpixels
<click>
In our quality control setup, we dispatch batches of 20 micro-tasks at a time to human processors.
Each batch is composed of 6 queries whose results we need plus 4 sentinel queries whose results we know. All queries appear twice in each batch in random order, so we can check an HP's output for internal consistency as well as for agreement on the sentinel tasks.

Note that we use sentinel queries from the same image, so the user must answer ~10–20 queries himself to get the results of the crowd. This is to prevent detection of the sentinel tasks. If many instances of the algorithm were run at once, the queries could be intermingled and then sentinel answers wouldn't have to come from the same image.
<click>
We send each batch to 3 different HPs for a median vote.
<click>
We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>
Finally, we combine the input by solving a laplace equation.

Algorithm

```
DEPTH-LAYERS(image $I$, sentinel queries $S$)
1   Segment $I$ into regions (using mean-shift and SLIC)
2   Insert all pairs of neighboring regions into $Q$
3   loop in parallel until each pair has been visited $N$ times
4       Gather $K$ random pairs from $Q$
5       Gather $M$ random pairs from $S$
6       for each pair: Build the visual query & Duplicate it
7       Mix the $2K + 2M$ queries
8       $results$ = send all queries to an HP
9       if $average(consistent(results)) \geq 0.75$
10          for each pair
11              Add consistent results to the list of votes
12              Increment #visited
13  for each pair of neighboring regions
14      $final\_result = majority$(list of votes)
15  Solve the Laplace equation to construct a depth map
```

Partition

Quality Control Setup

Dispatch

Verify

This is what an HC algorithm looks like in pseudocode.

<click>
We partition the task into superpixels
<click>
In our quality control setup, we dispatch batches of 20 micro-tasks at a time to human processors.
Each batch is composed of 6 queries whose results we need plus 4 sentinel queries whose results we know. All queries appear twice in each batch in random order, so we can check an HP's output for internal consistency as well as for agreement on the sentinel tasks.

Note that we use sentinel queries from the same image, so the user must answer ~10–20 queries himself to get the results of the crowd. This is to prevent detection of the sentinel tasks. If many instances of the algorithm were run at once, the queries could be intermingled and then sentinel answers wouldn't have to come from the same image.
<click>
We send each batch to 3 different HPs for a median vote.
<click>
We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>
Finally, we combine the input by solving a laplace equation.

Algorithm

```
DEPTH-LAYERS(image $I$, sentinel queries $S$)
1   Segment $I$ into regions (using mean-shift and SLIC)
2   Insert all pairs of neighboring regions into $Q$
3   loop in parallel until each pair has been visited $N$ times
4       Gather $K$ random pairs from $Q$
5       Gather $M$ random pairs from $S$
6       for each pair: Build the visual query & Duplicate it
7       Mix the $2K + 2M$ queries
8       $results$ = send all queries to an HP
9       if $average(consistent(results)) \geq 0.75$
10          for each pair
11              Add consistent results to the list of votes
12              Increment #visited
13  for each pair of neighboring regions
14      $final\ result = majority$(list of votes)
15  Solve the Laplace equation to construct a depth map
```

Partition

Quality Control Setup

Dispatch

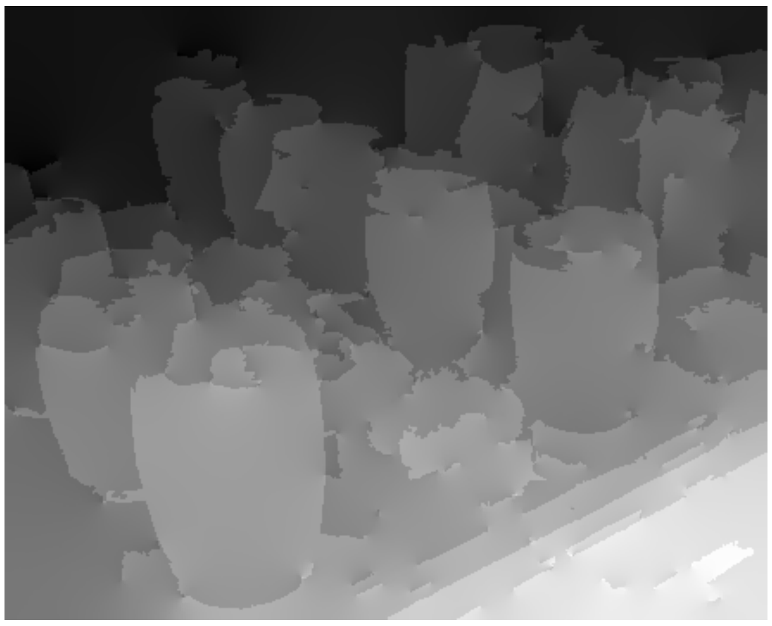
Verify

Combine

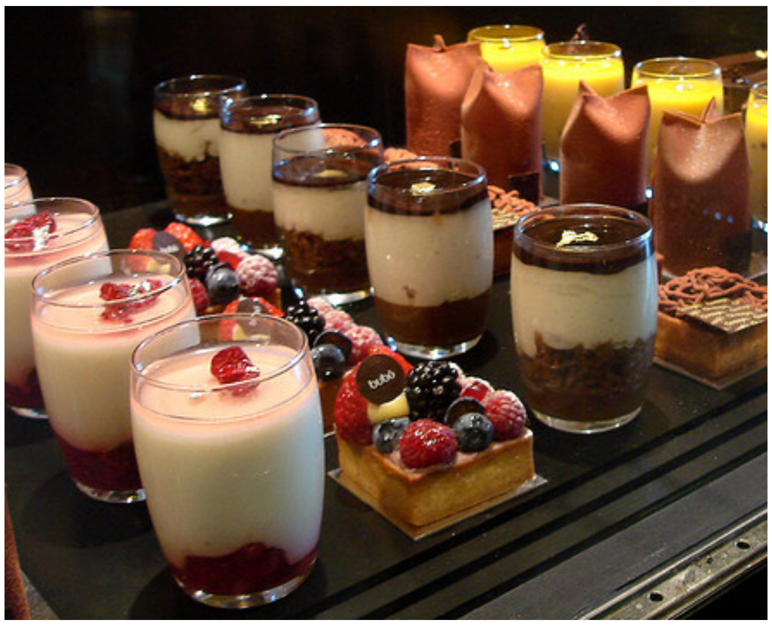
This is what an HC algorithm looks like in pseudocode.

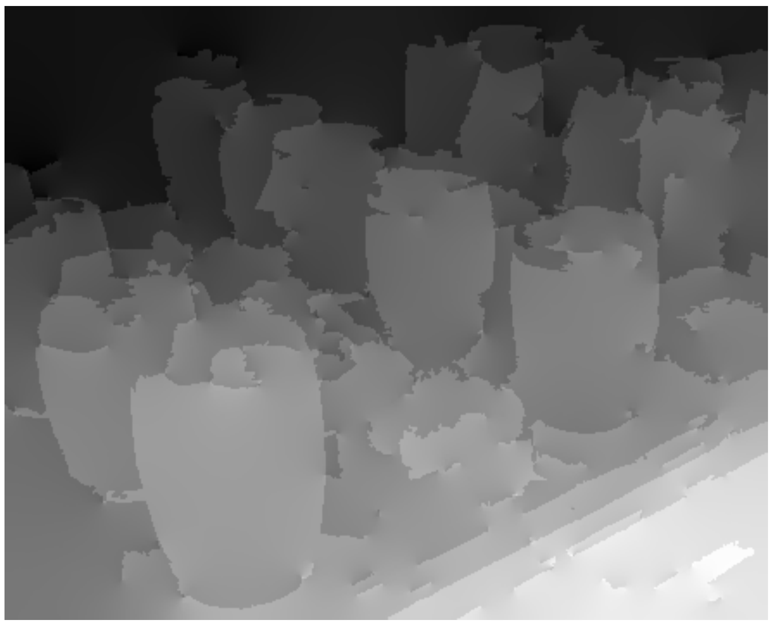
<click>
We partition the task into superpixels
<click>
In our quality control setup, we dispatch batches of 20 micro-tasks at a time to human processors.
Each batch is composed of 6 queries whose results we need plus 4 sentinel queries whose results we know. All queries appear twice in each batch in random order, so we can check an HP's output for internal consistency as well as for agreement on the sentinel tasks.

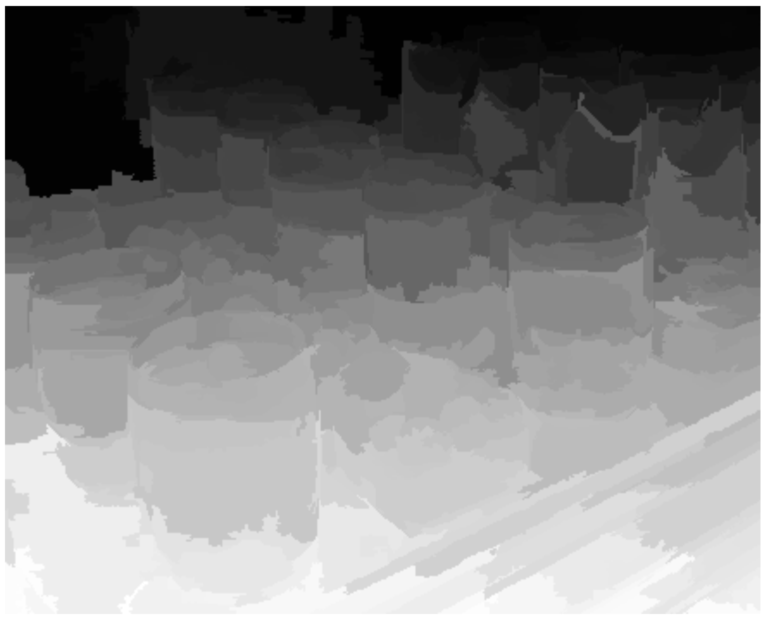
Note that we use sentinel queries from the same image, so the user must answer ~10–20 queries himself to get the results of the crowd. This is to prevent detection of the sentinel tasks. If many instances of the algorithm were run at once, the queries could be intermingled and then sentinel answers wouldn't have to come from the same image.
<click>
We send each batch to 3 different HPs for a median vote.
<click>
We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>
Finally, we combine the input by solving a laplace equation.



Here are our results
<click>
versus Make3D [Saxena et al. 2009].
We get a pretty good depth map, especially compared to a state-of-the-art automatic technique.
<click>
Here is a depth-of-field effect applied.

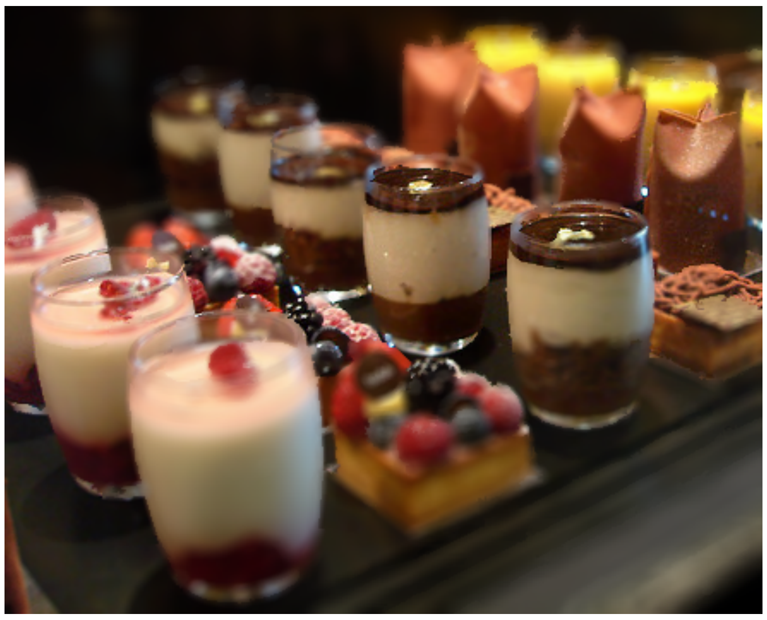




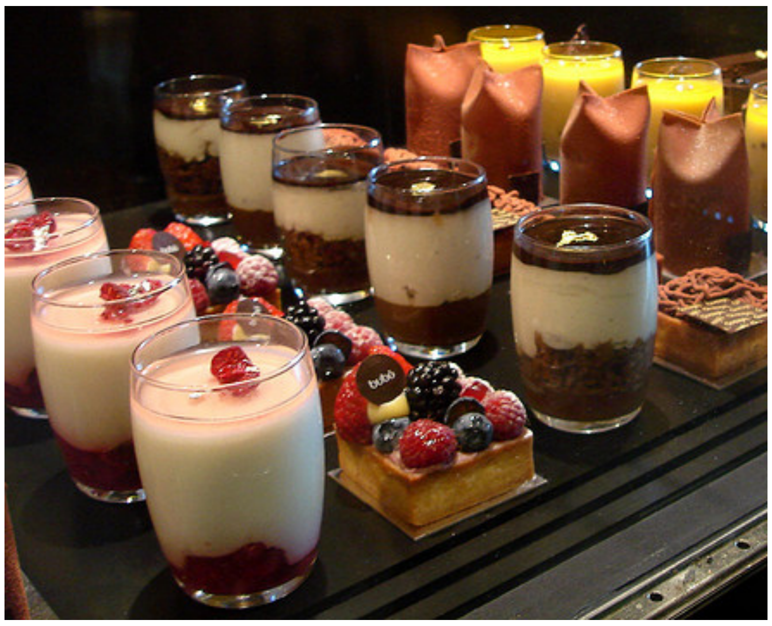


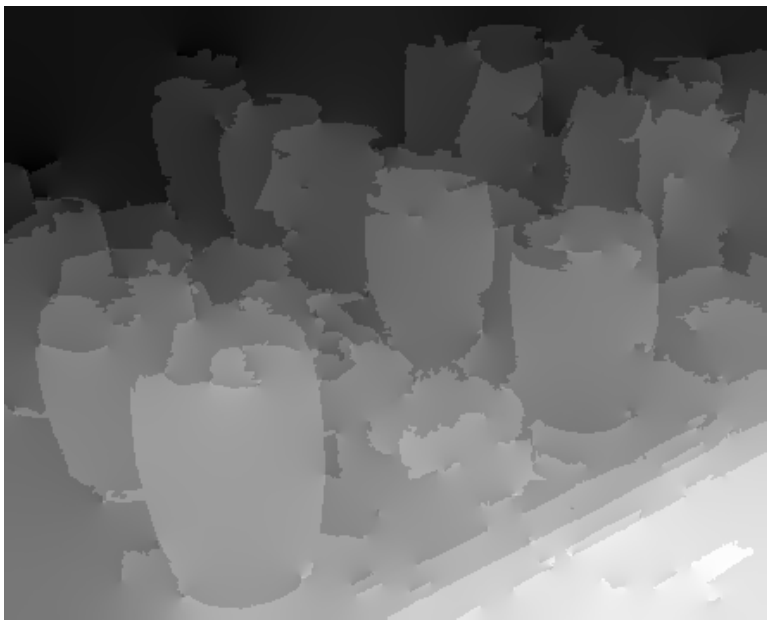
Automatic (Make3D)

Here are our results
<click>
versus Make3D [Saxena et al. 2009].
We get a pretty good depth map, especially compared to a state-of-the-art automatic technique.
<click>
Here is a depth-of-field effect applied.

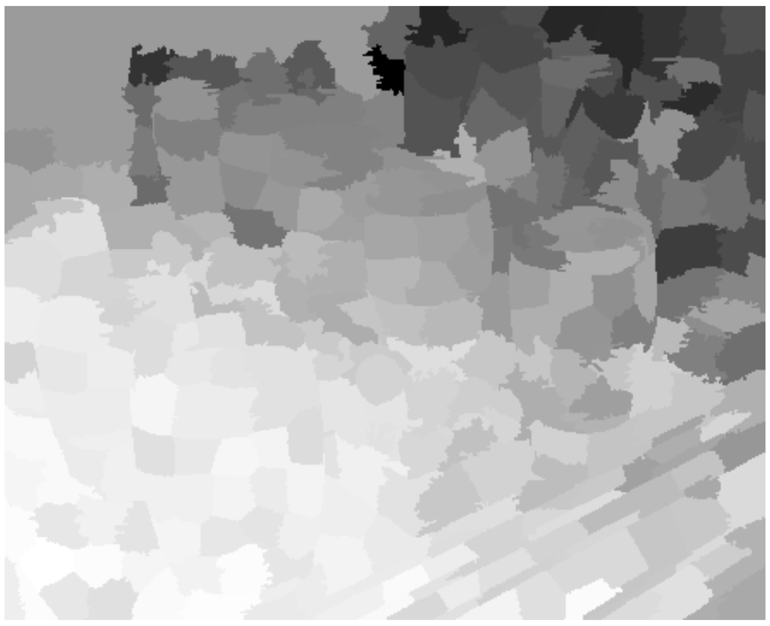


Here are our results
<click>
versus Make3D [Saxena et al. 2009].
We get a pretty good depth map, especially compared to a state-of-the-art automatic technique.
<click>
Here is a depth-of-field effect applied.



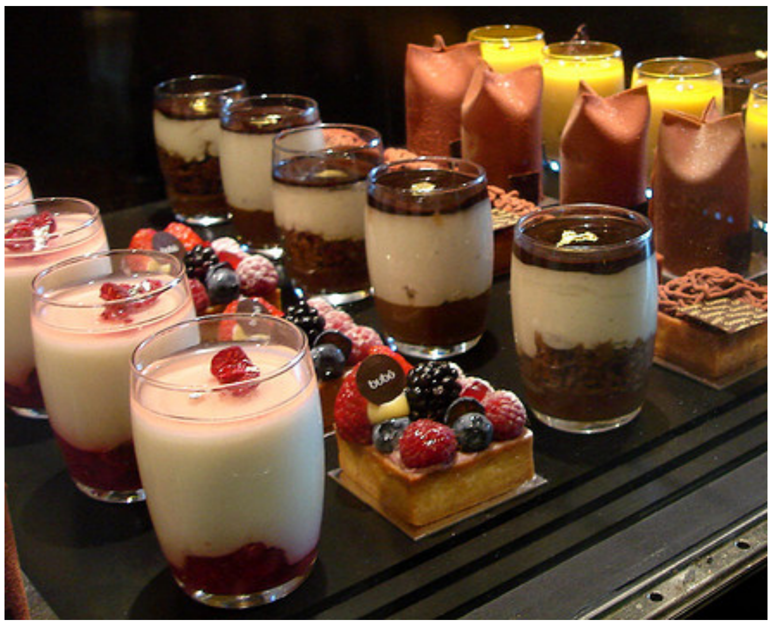


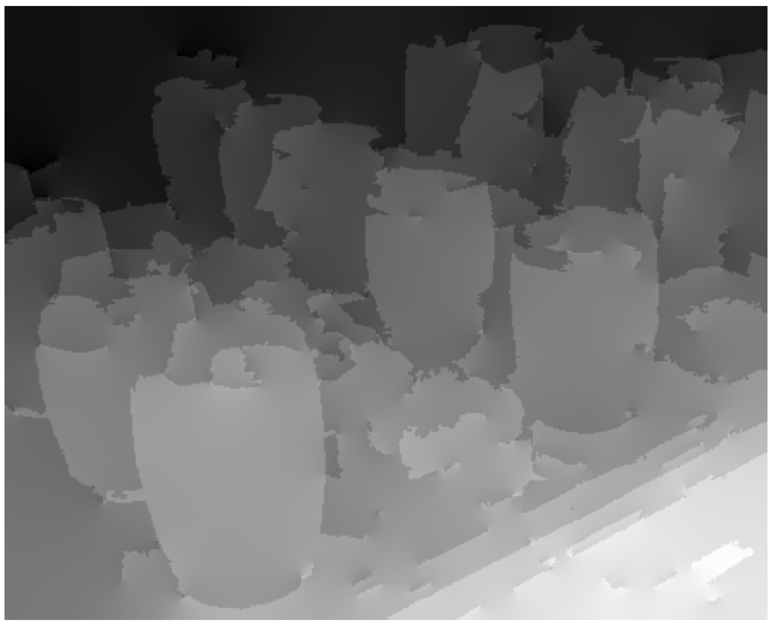
discrete depth



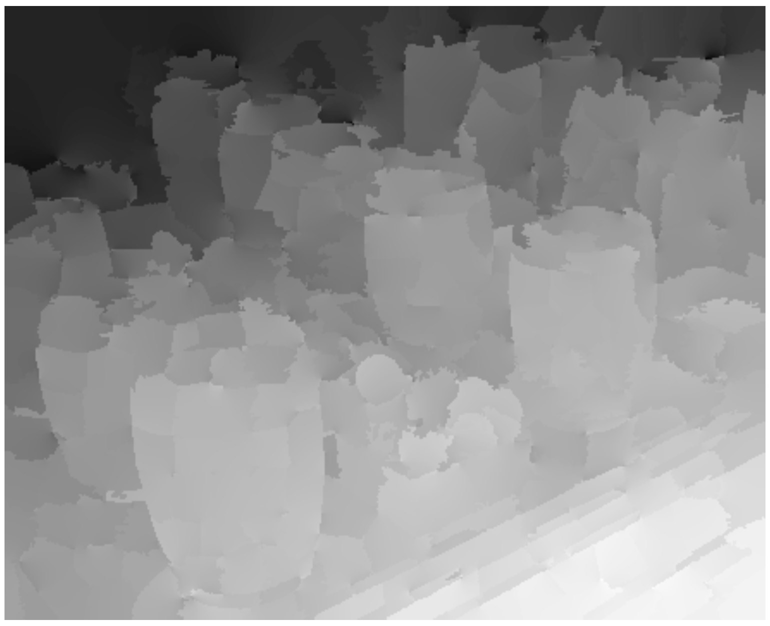
absolute depth

This is the comparison to the “absolute depth” version of the micro-task I promised. While it looks correct overall, it is extremely noisy.
This is what we would expect from the psychology literature; there is no good rectification to correct for humans’ differing internal biases that can be done.
Thresholds for quality control (is it consistent? does it match the sentinel?) are very difficult.



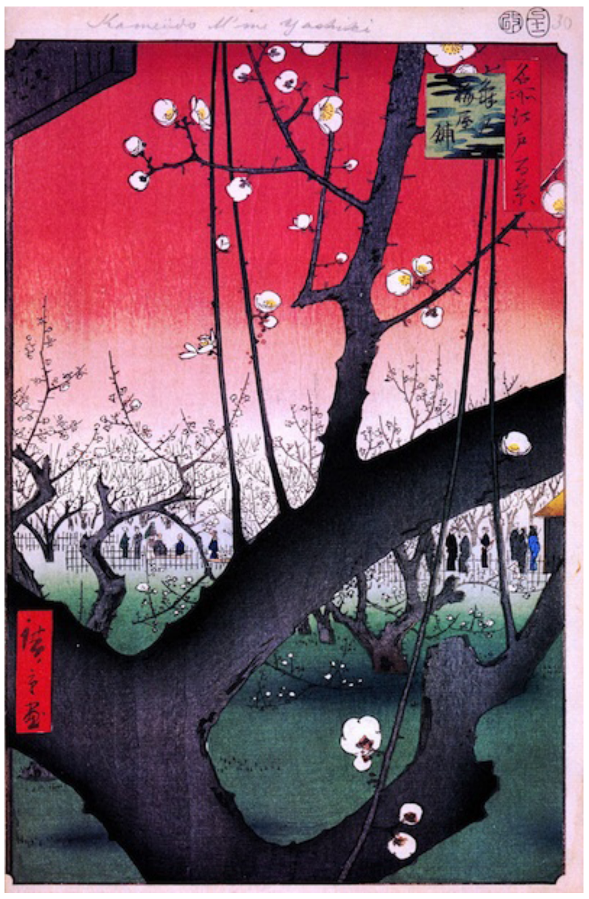


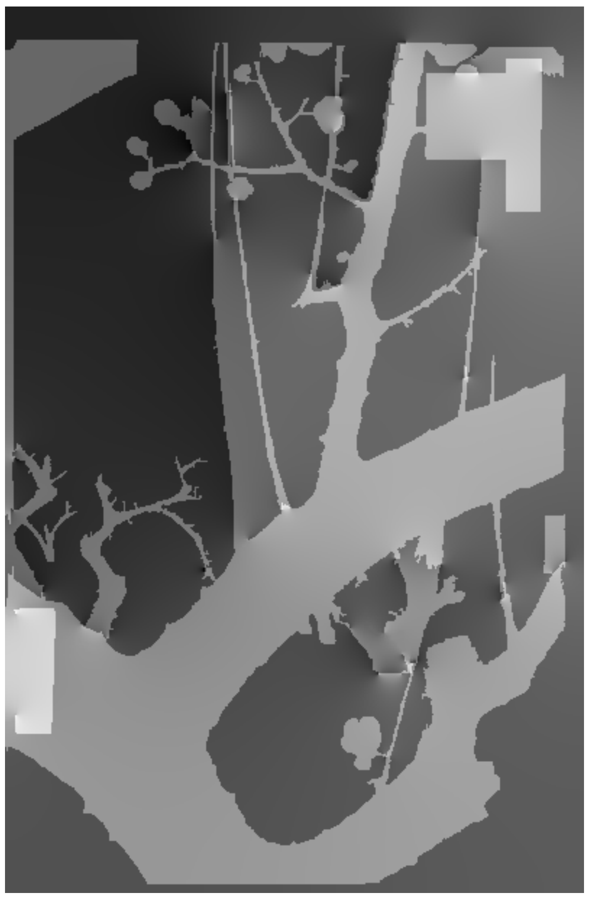
discrete depth

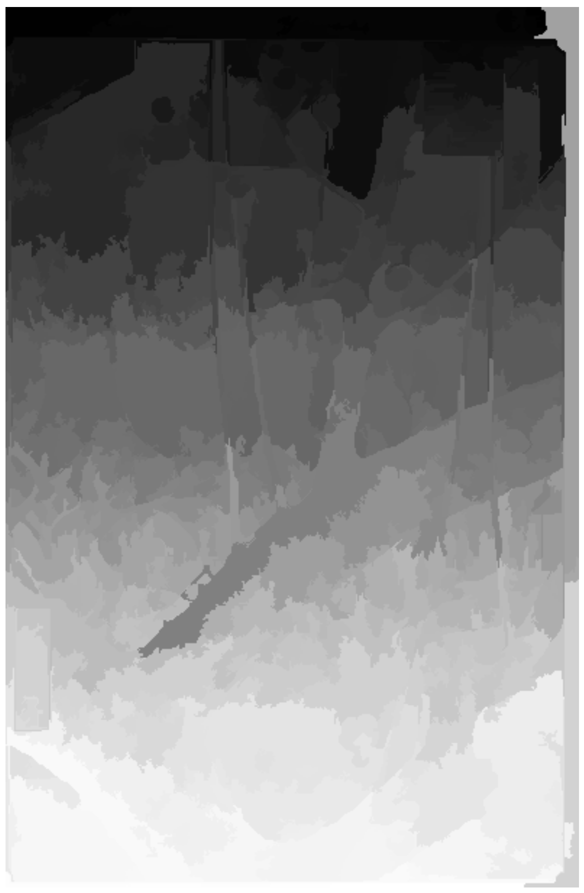


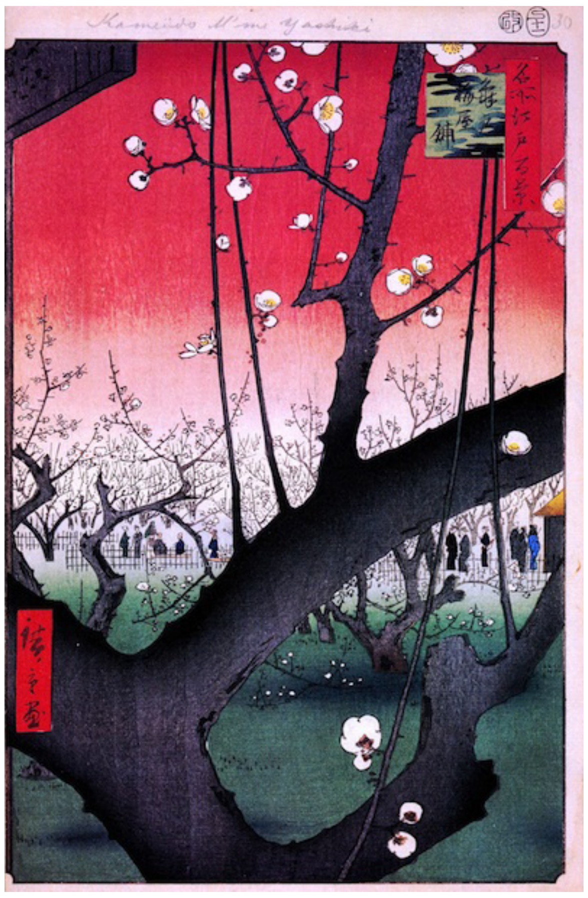
relative depth

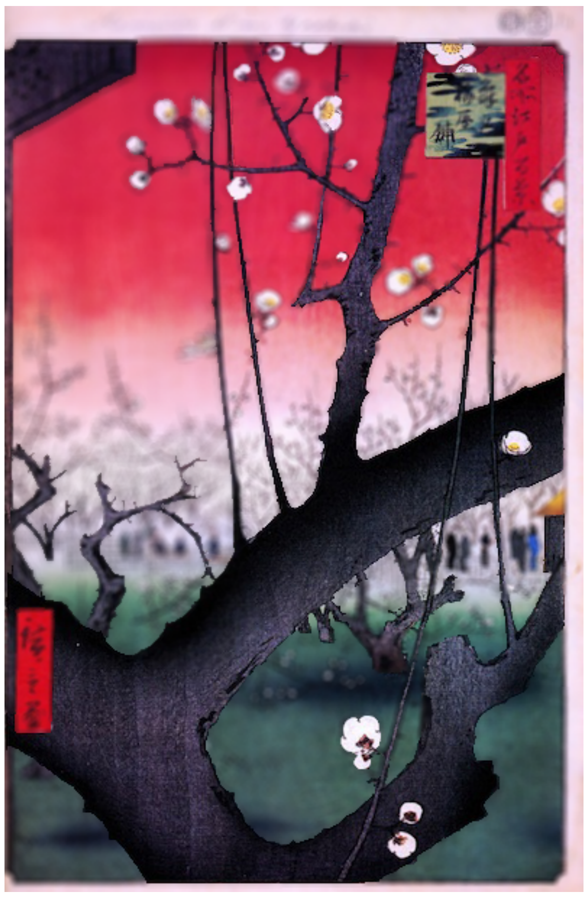
Asking HPs to choose a continuous, relative depth between neighboring patches with a slider leads to pretty good results—essentially a noisier version of the discrete question.


[Hiroshige]

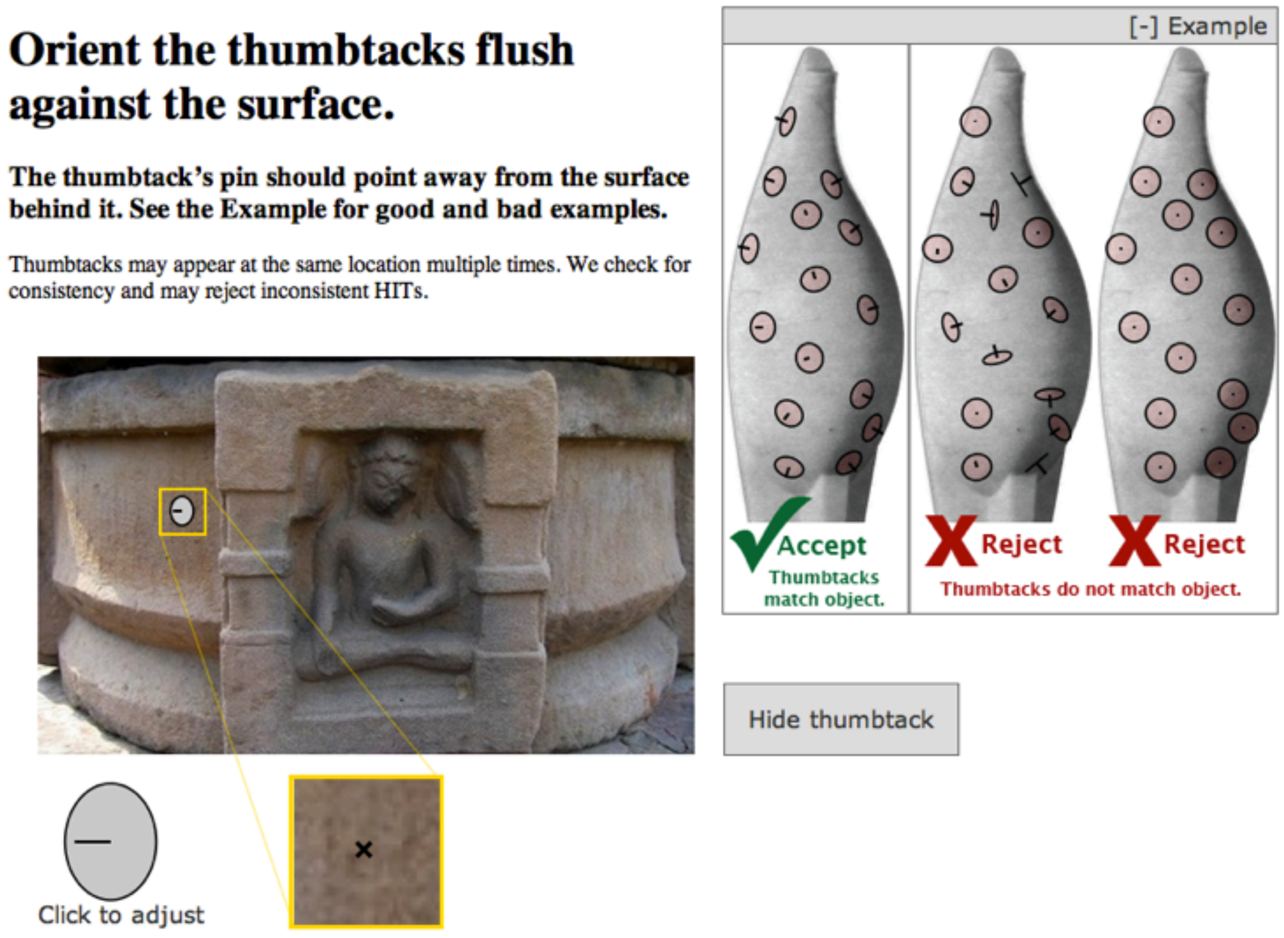



Automatic (Make3D)


[Hiroshige]



Algorithm 2: Normal Map


Orient the thumbtacks flush against the surface.

The thumbtack's pin should point away from the surface behind it. See the Example for good and bad examples.

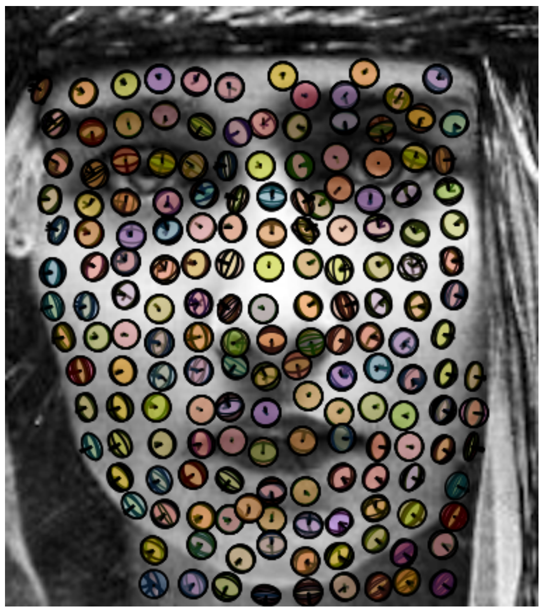
Thumbtacks may appear at the same location multiple times. We check for consistency and may reject inconsistent HITs.

In our second algorithm, we create a normal map for a given image. This can be used for relighting or for surface reconstruction.

We use a "gauge figure" micro-task that comes from the perception literature [Koenderink et al. 1992];
it was also used by [Cole et al. 2009] for gathering normals using the Mechanical Turk.
I should mention that [Cole et al. 2009] was the inspiration for this research.
HPs orient the gauge figure so that it appears to lie flush against the surface in the image.

We implemented this algorithm in an adaptive manner, so large variations in normals led to more queries in that area.

We also experimented with sliders for the xy direction and z-slant of the gauge figures, but did not find a difference in performance.

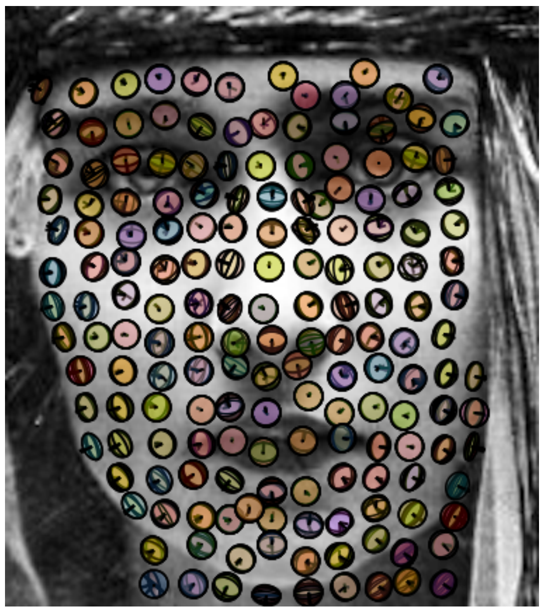
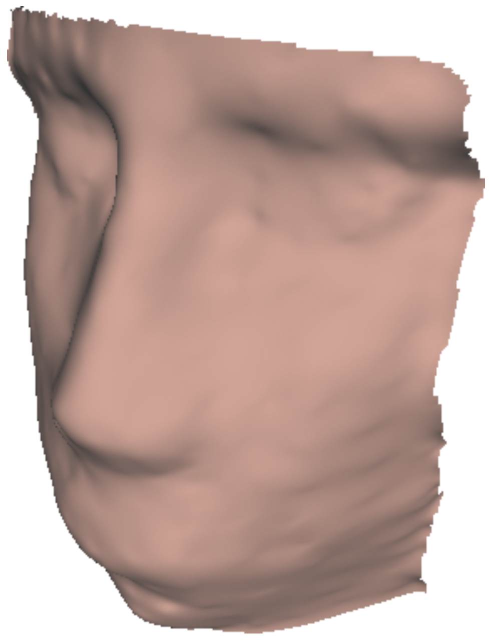
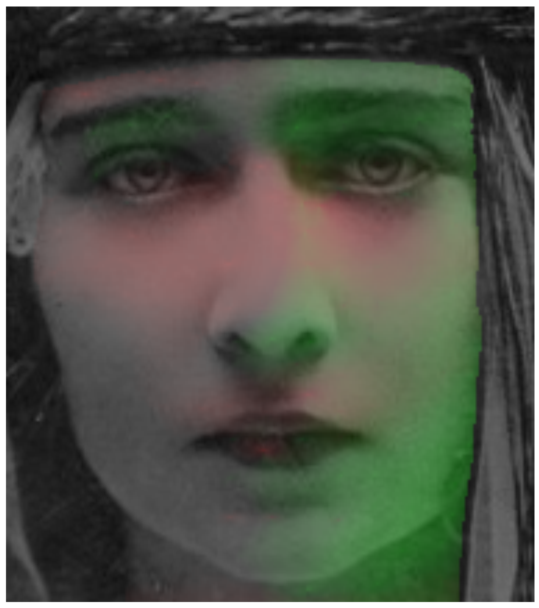


Here we add two new lights to an old photograph of a face and create a 3D reconstruction.
<click>

Our normal map algorithm has a similar quality control setup to our depth layers algorithm, though thresholds are needed when comparing normals as well as a rectification step accounting for humans differing internal biases.
Since we have sentinel queries, we solve for the depth scaling of the normals (not the full bas-relief ambiguity; the difference was found to be minor with gauge figures) to align each HP with the sentinel queries.

For composition, we solve a bilaplace equation (bilaplacian = 0) with the user-given normals as least-squares constraints. This removes noise and ensures the consistency of the normals.

The shape-from-shading approach shown is “Tsai and Shah”, which gave the best output from among the Shape-from-Shading approaches surveyed in [Durou et al. 2008].

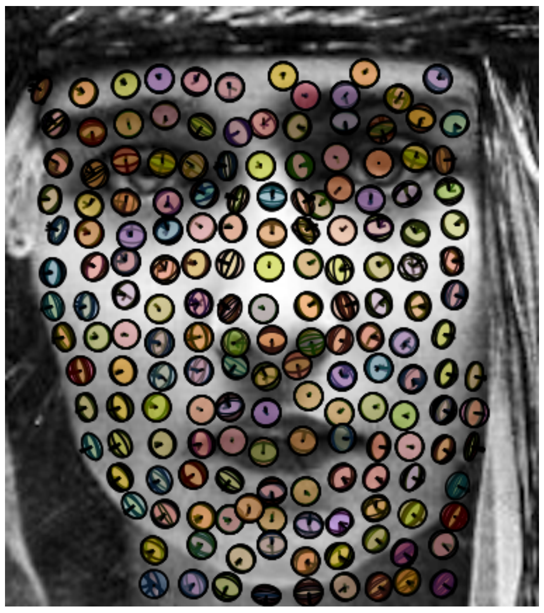
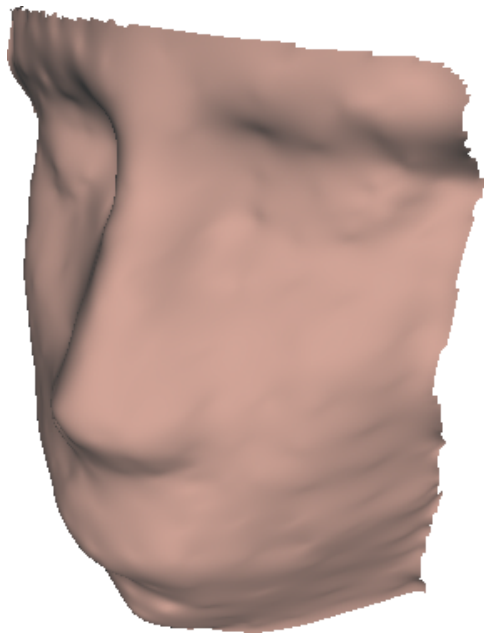
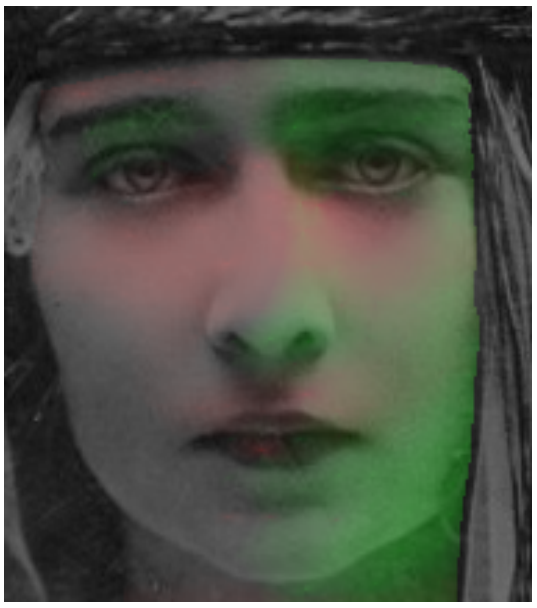
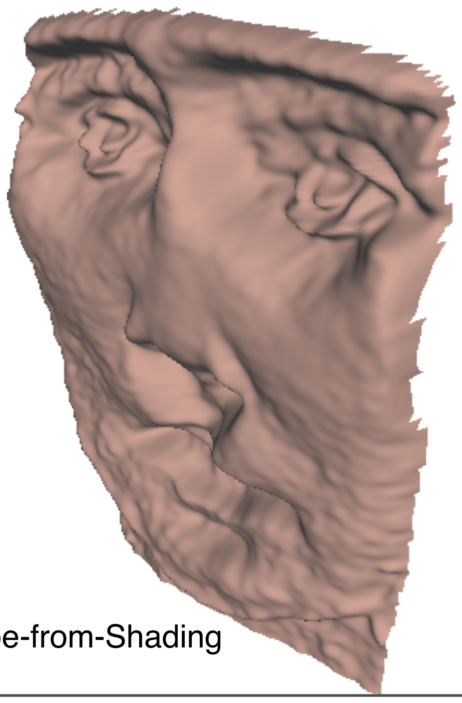




Here we add two new lights to an old photograph of a face and create a 3D reconstruction.
<click>

Our normal map algorithm has a similar quality control setup to our depth layers algorithm, though thresholds are needed when comparing normals as well as a rectification step accounting for humans differing internal biases.
Since we have sentinel queries, we solve for the depth scaling of the normals (not the full bas-relief ambiguity; the difference was found to be minor with gauge figures) to align each HP with the sentinel queries.

For composition, we solve a bilaplace equation (bilaplacian = 0) with the user-given normals as least-squares constraints. This removes noise and ensures the consistency of the normals.

The shape-from-shading approach shown is "Tsai and Shah", which gave the best output from among the Shape-from-Shading approaches surveyed in [Durou et al. 2008].

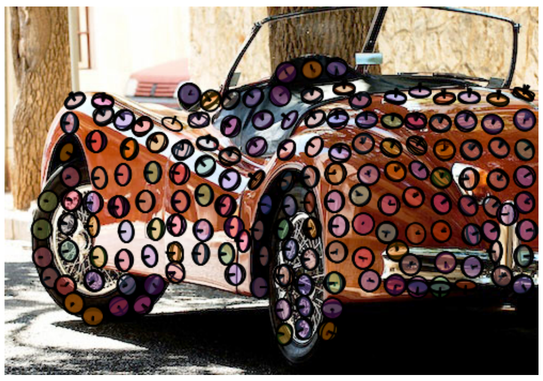
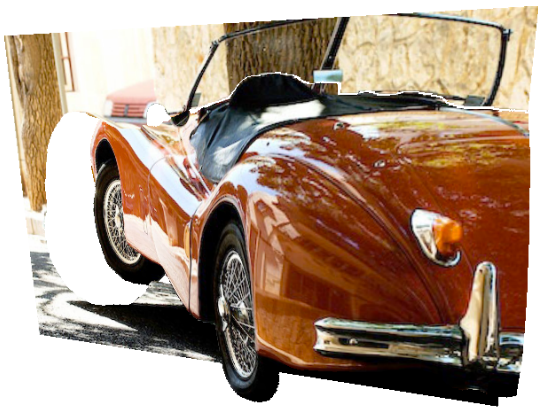
Shape-from-Shading

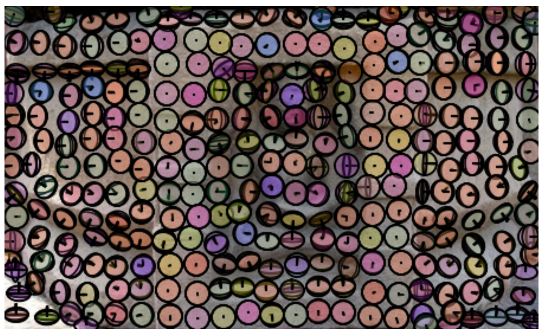
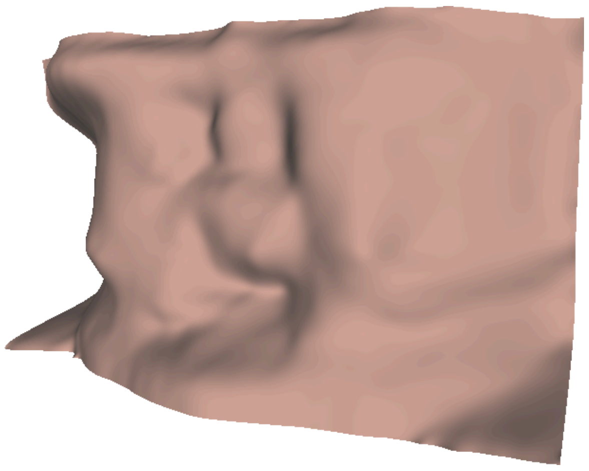
Here we add two new lights to an old photograph of a face and create a 3D reconstruction.
<click>

Our normal map algorithm has a similar quality control setup to our depth layers algorithm, though thresholds are needed when comparing normals as well as a rectification step accounting for humans differing internal biases.
Since we have sentinel queries, we solve for the depth scaling of the normals (not the full bas-relief ambiguity; the difference was found to be minor with gauge figures) to align each HP with the sentinel queries.

For composition, we solve a bilaplace equation (bilaplacian = 0) with the user-given normals as least-squares constraints. This removes noise and ensures the consistency of the normals.

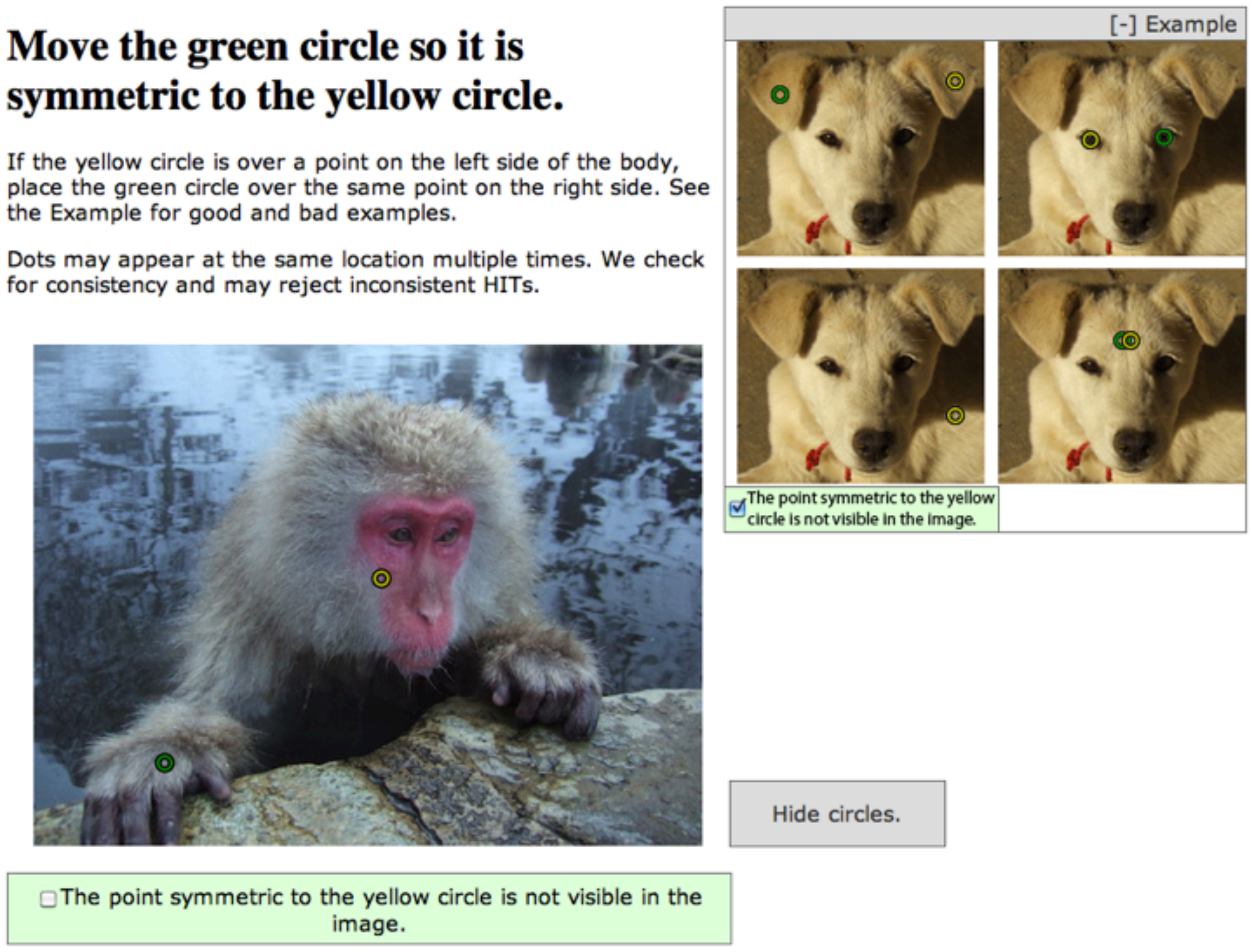
The shape-from-shading approach shown is “Tsai and Shah”, which gave the best output from among the Shape-from-Shading approaches surveyed in [Durou et al. 2008].

[Pedro Ribeiro Simões]



[Warren Apel]



Here are a couple more examples of 3D reconstructions.

Algorithm 3: Bilateral Symmetry Map



Move the green circle so it is symmetric to the yellow circle.

If the yellow circle is over a point on the left side of the body, place the green circle over the same point on the right side. See the Example for good and bad examples.

Dots may appear at the same location multiple times. We check for consistency and may reject inconsistent HITs.

[-] Example

☑ The point symmetric to the yellow circle is not visible in the image.

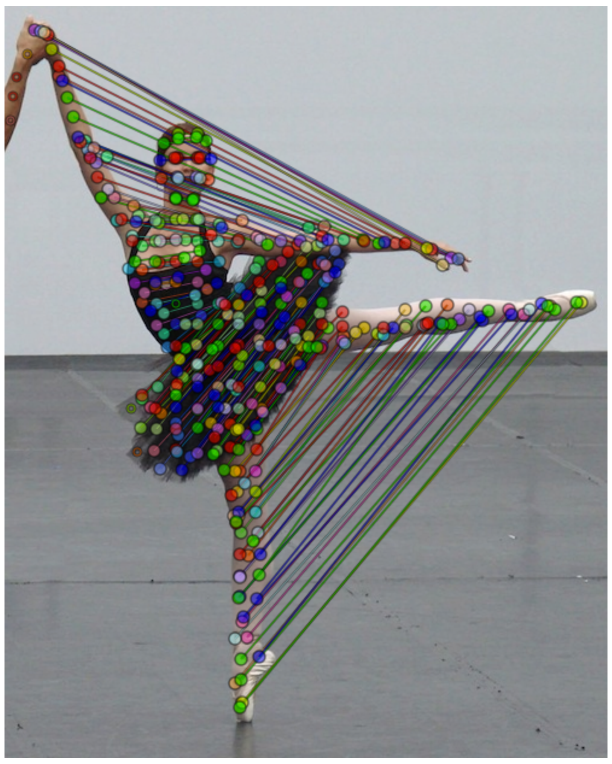
☐ The point symmetric to the yellow circle is not visible in the image.

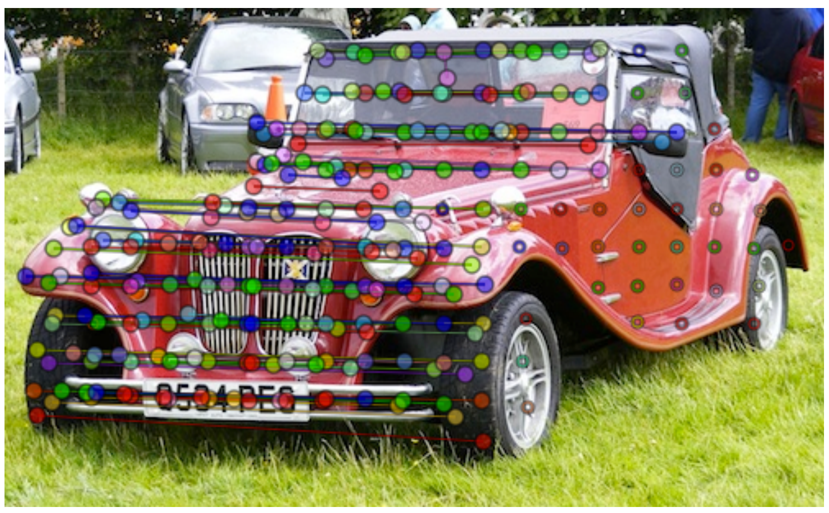
In our third algorithm, we create a bilateral symmetry map for an object in an image. This can be used for edit propagation or retargeting.

We sample points in the region of interest and ask HPs to identify the symmetrically opposed point.

This algorithm also has a similar quality control setup to our other algorithms.

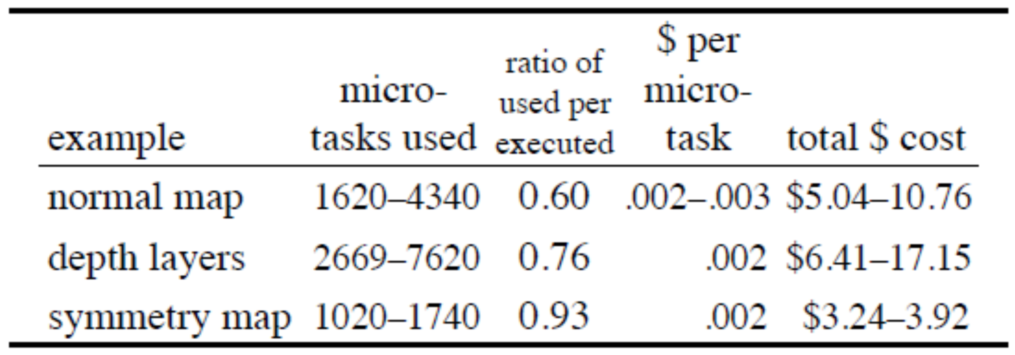
[flickr user dalbera]





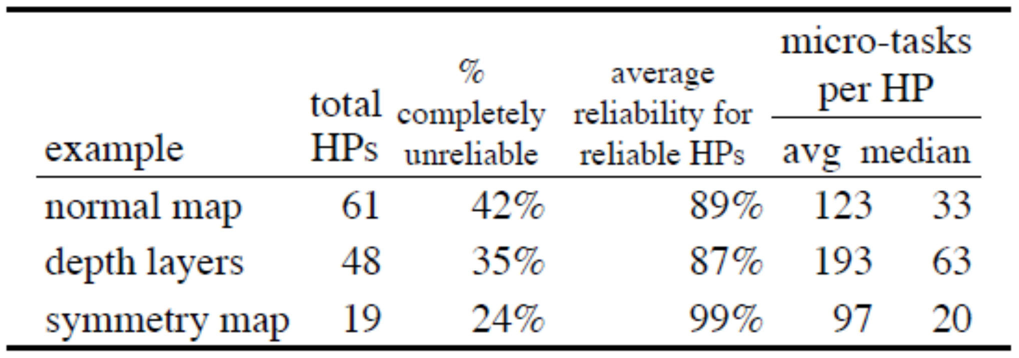
Here are bilateral symmetry maps created with our algorithms.

Some Statistics



example	micro-tasks used	ratio of used per executed	$ per micro-task	total $ cost
normal map	1620–4340	0.60	.002–.003	$5.04–10.76
depth layers	2669–7620	0.76	.002	$6.41–17.15
symmetry map	1020–1740	0.93	.002	$3.24–3.92

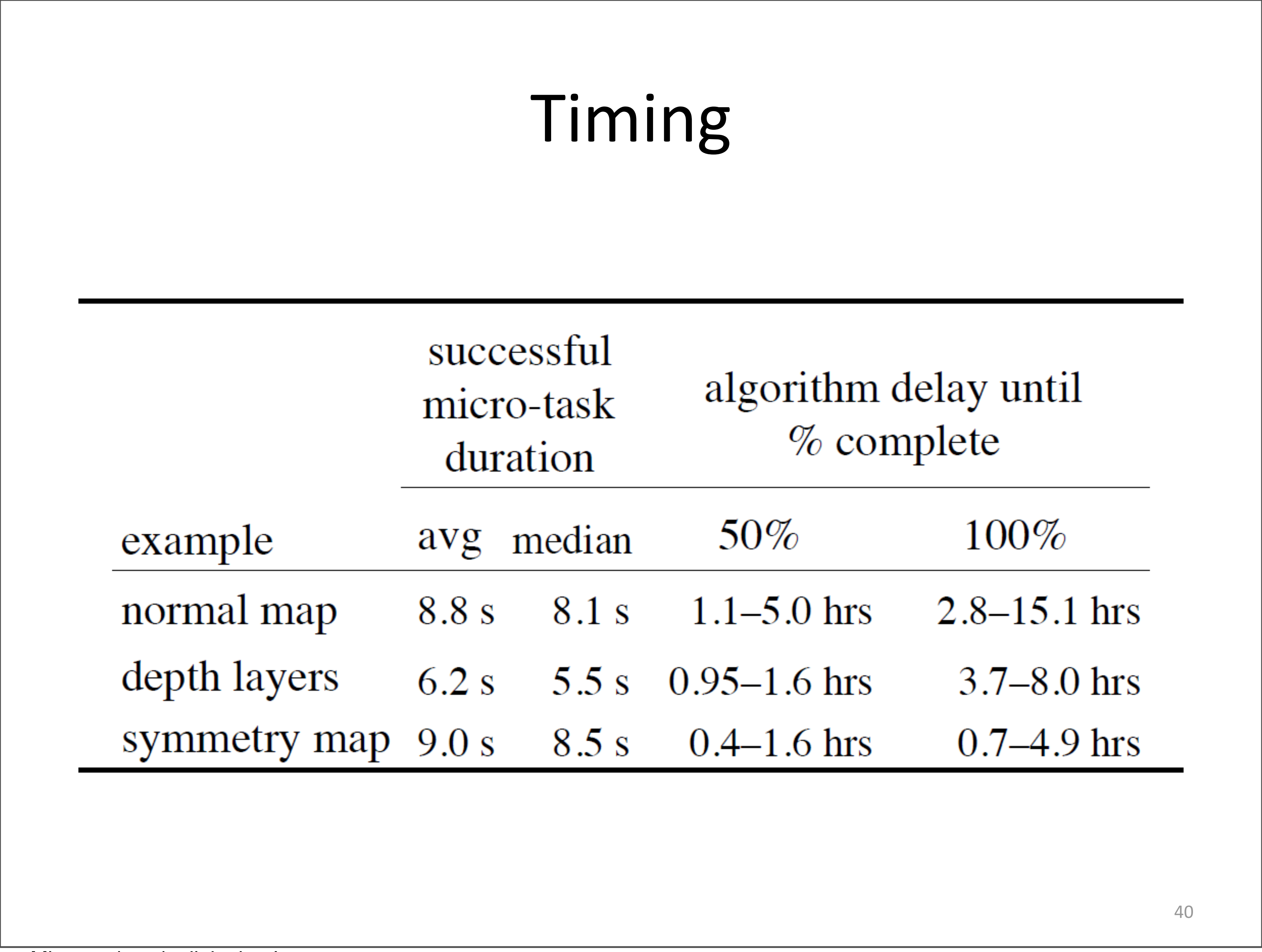
Table 1: *Micro-tasks*



example	total HPs	% completely unreliable	average reliability for reliable HPs	micro-tasks per HP	
				avg	median
normal map	61	42%	89%	123	33
depth layers	48	35%	87%	193	63
symmetry map	19	24%	99%	97	20

Table 2: *Human Processors*

- We use a lot of micro-tasks. Depth layers is most expensive, because it's per neighboring patches, not per patch.
    - We know from the literature ([Mason and Watts 2010]) that payment is not correlated with accuracy, only with how likely an HP is to do the task.
- Normal Map task is most difficult, judging by completely unreliable HPs. Can we make a better task?
- These micro-tasks per HP numbers are low; a median of 1–3 batches per HP. That implies people were able to do it right away and we could scale.



Timing

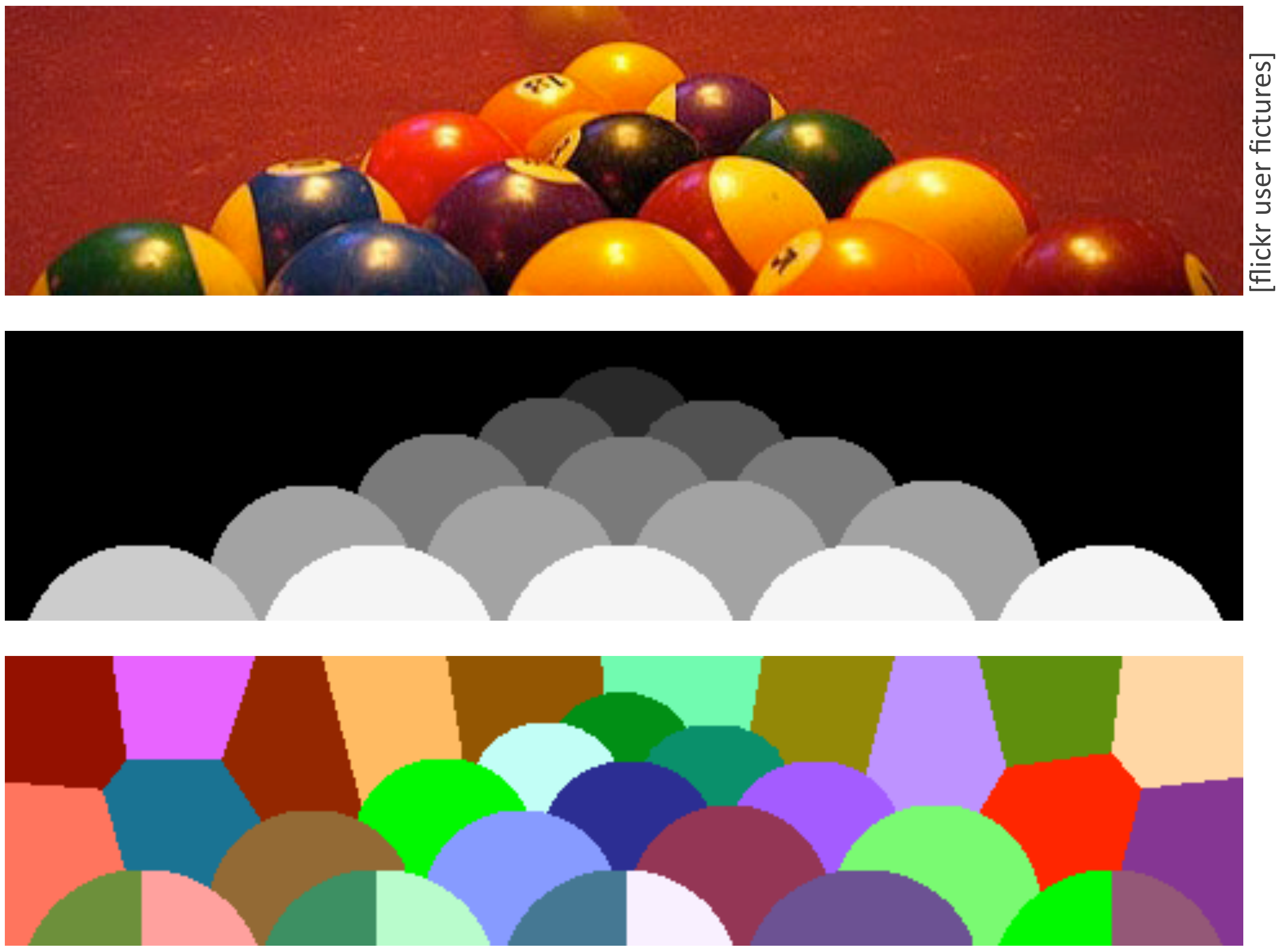
	successful micro-task duration		algorithm delay until % complete	
example	avg	median	50%	100%
normal map	8.8 s	8.1 s	1.1–5.0 hrs	2.8–15.1 hrs
depth layers	6.2 s	5.5 s	0.95–1.6 hrs	3.7–8.0 hrs
symmetry map	9.0 s	8.5 s	0.4–1.6 hrs	0.7–4.9 hrs

- Micro-tasks take little time!
- The total algorithm took a while, though delay is entirely due to waiting for HPs to choose to perform the tasks.
    - It's known from the literature ([Ipeirotis 2010; Chilton et al. 2010; Faridani et al. 2011; Mason and Suri 2011; Mason and Watts 2010]) that this is correlated with the amount one is willing to pay, as well as the amount of same-type jobs there are to do and with newness. Latter two would be helped if this were a popular algorithm, though it's anyone's guess how the market of HC will change in the future.

- Note that we could complete in ~3 minutes if we had enough people.

Paul Sajda (pronounced "shayda") can categorize images at 10 hz by using brain wave scanning ("Is there a ballerina in the image").

Accuracy



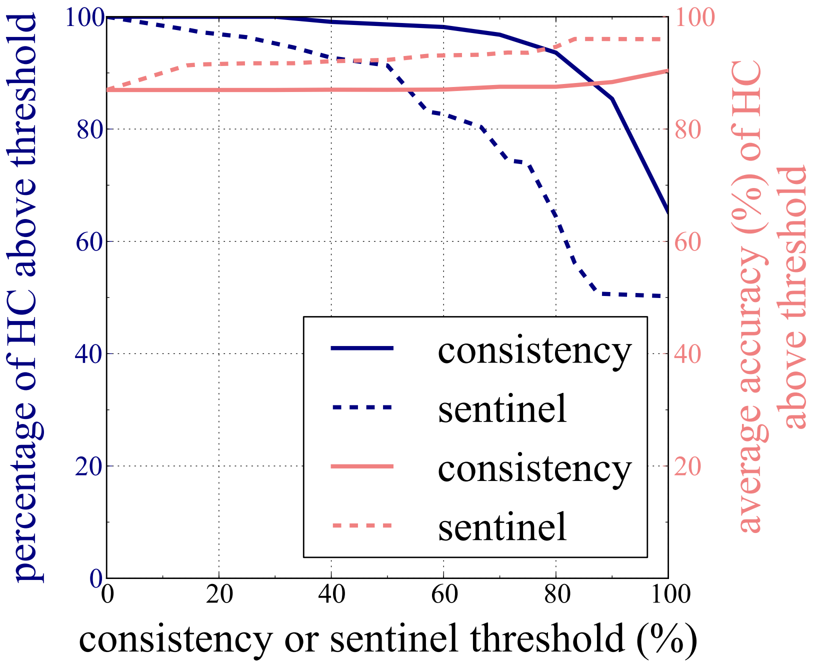
To test accuracy as a function of quality control parameters, we used this billiards image.
We ran many variations; the data you will see in the following graphs was generated from 4 calibrated runs.

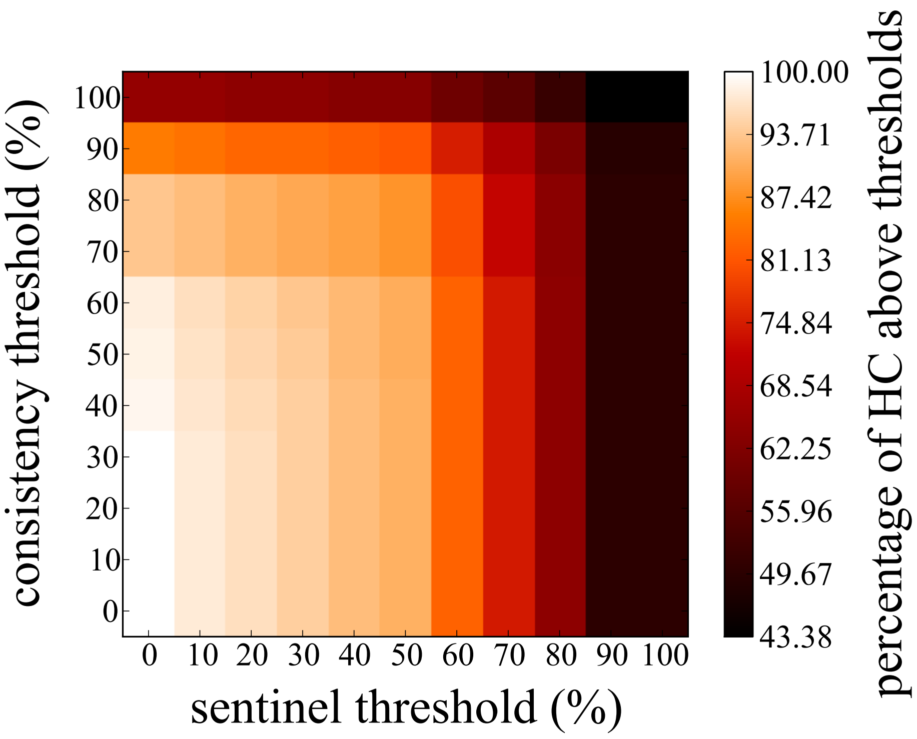
We varied the percent of queries in each batch that had to be internally consistent before throwing the entire batch out.
We varied the percent of sentinel queries in each batch that had to be accurate before throwing the entire batch out.

We also varied the number of reliable queries to use when computing each answer (the voting/median/average in the composition step).
%We also varied the number of different HPs to send each batch to (and receive reliable answers from). When we throw a batch out, we send it to a new HC (until we have N).

NOTE: We varied granularity, using smaller patches, but didn't find an interesting correlation using 30,60,90,120 patches. 120 patches is still far from per-pixel, which would be prohibitively expensive to run.

Accuracy





We ran the billiards example 4 times.

On the line graph, the dark blue lines show how many batches pass either the consistency or the sentinel test (one or the other). The heat map on the right is a 2D version of this.

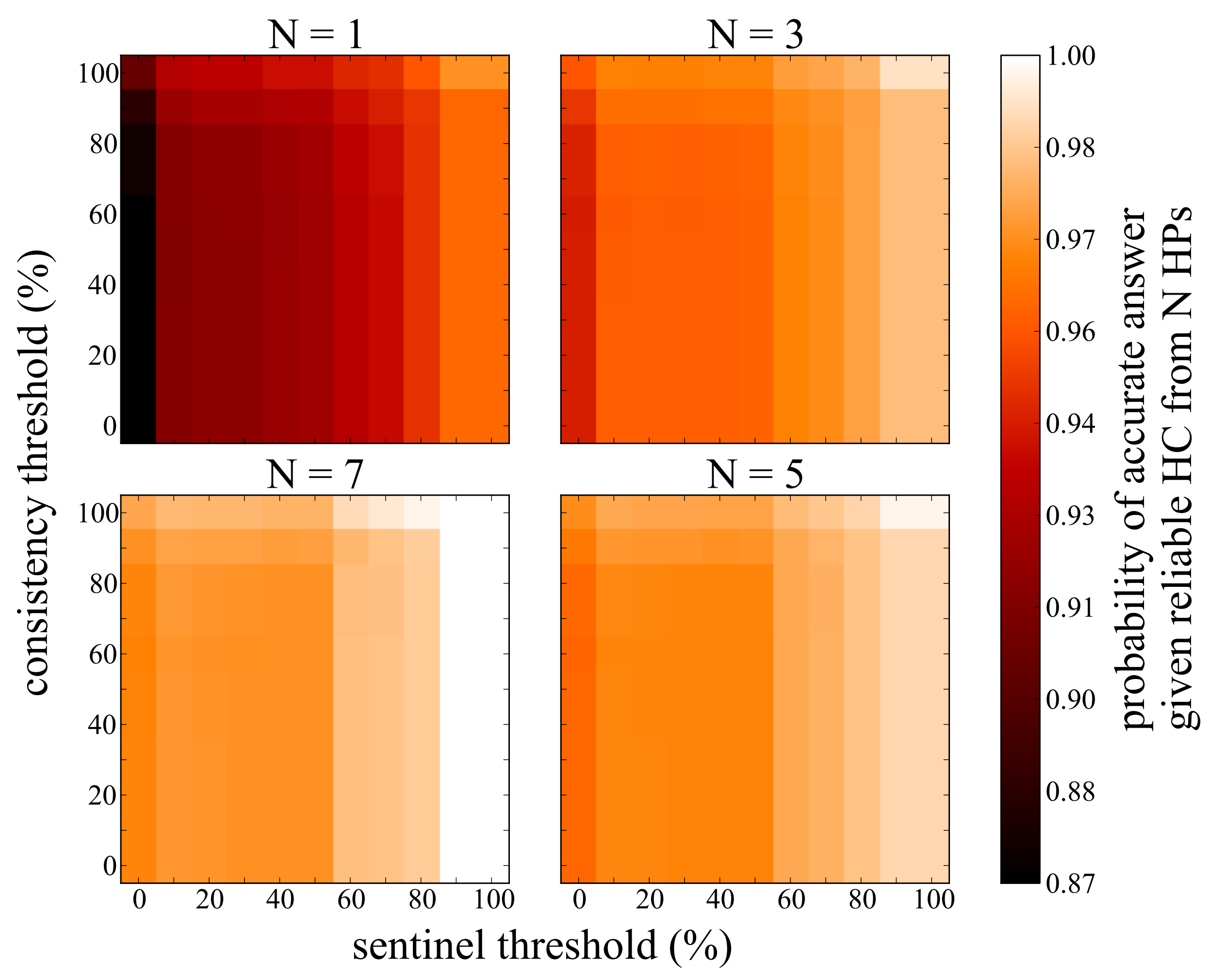
- The vast majority (94%) of HC batches are 80% consistent (or more), though only 65% are 100% consistent.
- Batches are more stratified in agreeing with sentinel operations. Only 74% were correct for 75% or more of the sentinel ops; only 50% were correct for 100%.

The salmon-colored lines depict the average accuracy of all HC batches above either the consistency or the sentinel test (one or the other).

- Average accuracy is only marginally affected by increasing the consistency threshold: from 0% to 80% to 100% only increases the accuracy from 87% to 88% to 90%.
- Increasing the sentinel threshold has a greater effect on average accuracy: from 87% to 94% to 96% as the threshold increases from 0% to 75% to 100% (at the cost of discarding 50% of HC batches!).

On the right, this is a two-dimensional plot of consistency and sentinel thresholds; at each location, both tests are applied.
If we only want 100% consistent and agreeing-with-sentinel, we throw out 57% of all HC batches, but we get 97% accuracy (as we'll see on the next slide).

Accuracy



Here we vary the number of reliable queries to use when computing each answer (the voting/median/average in the composition step). (When we throw a batch out, we send it to a new HC, until we have N.)

The overall trend to see is that the heat map “whitens” as N goes up.

Increasing the number of HPs used in voting reduces the likelihood that the final output is affected by inaccurate HC that nonetheless passes the sentinel and consistency tests.
% Each location in these heat maps depicts the probability that reliable queries from N different HPs produces the correct answer for the depth order between a pair of neighboring patches; the depicted value is the average over all pairs of neighboring patches.

There is no obvious “sweet spot.”
Interesting to note that you can have an expected 97% accuracy with only one reliable HC answer for each micro-task by setting the sentinel and consistency thresholds to 100%.
These thresholds are too strict to use when deciding whether to pay HPs—they get upset—so you must pay for substantially more HC than you’ll use.
Still, it’s cheaper to obtain 97% accuracy by requiring 100% consistency and sentinel accuracy *while paying HPs at 75% thresholds*
than to get reliable HC from N = 3 HPs with 75% consistency and sentinel thresholds and pay for 3x the number of accurate micro-tasks actually needed.

1 HP @ 100% c/s: pay for 100 · 72% = 167% of the number of accurate micro-tasks actually needed instead of 3x.
Need N > 1 HPs for achieve higher than 97% accuracy.

Take-away: you may be able to make-do with a single answer if it comes from a batch of high-quality HC.

Conclusions

- For hard problems, HC algorithms can beat automatic algorithms.
- Rephrase your problem in terms of reliable human perception.

- How can we improve efficiency?
- If this were a Photoshop plug-in, how much would people pay to use it?

efficiency: timing, cost: quality control overhead, different micro-task designs (optimizing perceptual experiment design)

Thank you. Questions?

End

Related Work (1/6)

- Many kinds of collective intelligence
  - open-source software, Wikipedia, PageRank, supervised learning, elections?
- Modern assembly line (Ford Motor Company 1908–1915)
- Interchangeable parts:
  - Adam Smith on division of labor (1776)
  - Terracotta army (3rd century BC)
  - Venetian Arsenal (ship building)

Many kinds of collective intelligence (open-source software, wikipedia, pagerank, supervised learning in general, elections?)
- Collaborative filtering: [Goldberg et al. 1992; Adomavicius and Tuzhilin 2005]
- Open Mind Initiative

Modern assembly line (Ford Motor Company 1908--1915)

Wikipedia:

In his autobiography Henry Ford (1922) mentions several benefits of the assembly line including:

Workers do no heavy lifting
No stooping or bending over
No special training required
There are jobs that almost anyone can do
Provided employment to immigrants

The gains in productivity allowed Ford to increase worker pay from $2.50 per day to $5.00 per day and to reduce the hourly work week while continuously lowering the Model T price. These goals appear altruistic; however, it has been argued that they were implemented by Ford in order to reduce high employee turnover.

Interchangeable parts:

Adam Smith on division of labor (1776)
Terracotta army (3rd century BC)
Venetian Arsenal (ship building)

Related Work (2/6)

- Online:
  - [von Ahn 2008]
  - [Little et al. 2010a,b] and [Bernstein 2010]
  - [Bigham et al. 2010] and [Bernstein 2011]
  - [Davis et al. 2010]
  - [Sorokin et al. 2010]
  - many more recent/contemporary applications
- Recast existing experiments
  - [Koenderink et al. 1992], [Cole et al. 2009]
  - [Chen et al. 2009]

Online algorithms

[von Ahn 2005]: CAPTCHA (not useful computation; in [reCAPTCHA 2008] it became useful), ESP game (labeling)

[Little et al. 10a,b] and [Bernstein 2010] for text processing and sorting. Their Soylent system makes a similar argument as we do for incorporating human computation into a word processor. VizWiz [Bigham et al. 2010] and [Bernstein 2011] focus on decreasing latency (applied to image labeling for the blind (VizWiz) and applied to choosing an image from a short video, a creative task posing a figure, and perceptual sorting (Bernstein)).

[Davis et al. 2010] makes a similar argument about using human computation in online algorithms and evaluated many characteristics of what they call “Human Processing Units”.

[Sorokin et al. 2010] introduced a workflow for 3D object reconstruction to assist robots.

Many more recent works databases (CrowdDB), calorie counting, ...

Can recast existing experiments as human computation operations: [Koenderink et al. 1992]/[Cole et al. 2009] or [Chen et al. 2009]. In those works, primary goal is gathering data on humans, not on the efficiency of the data gathering per se.

Related Work (3/6)

- Training data:
  - ESP Game [von Ahn and Dabbish 2004], …
  - LabelMe [Russel et al. 2008; Yuen et al. 2009]
  - Hands by Hand [Spiro et al. 2010]
- Using HC data gathered offline:
  - [Talton et al. 2009]
  - [Kalogerakis et al. 2010] using [Chen et al. 2009]

Gathering training data, but don't have tight algorithmic coupling. ESP Game [von Ahn and Dabbish 2004], LabelMe [Russel et al. 2008; Yuen et al. 2009], motion tracking [Spiro et al. 2010].

HC for learning: [Talton et al. 2009] for tree modeling by sampling human good models. [Kalogerakis et al. 2010] for segmentation from [Chen et al. 2009] data.

Related Work (4/6)

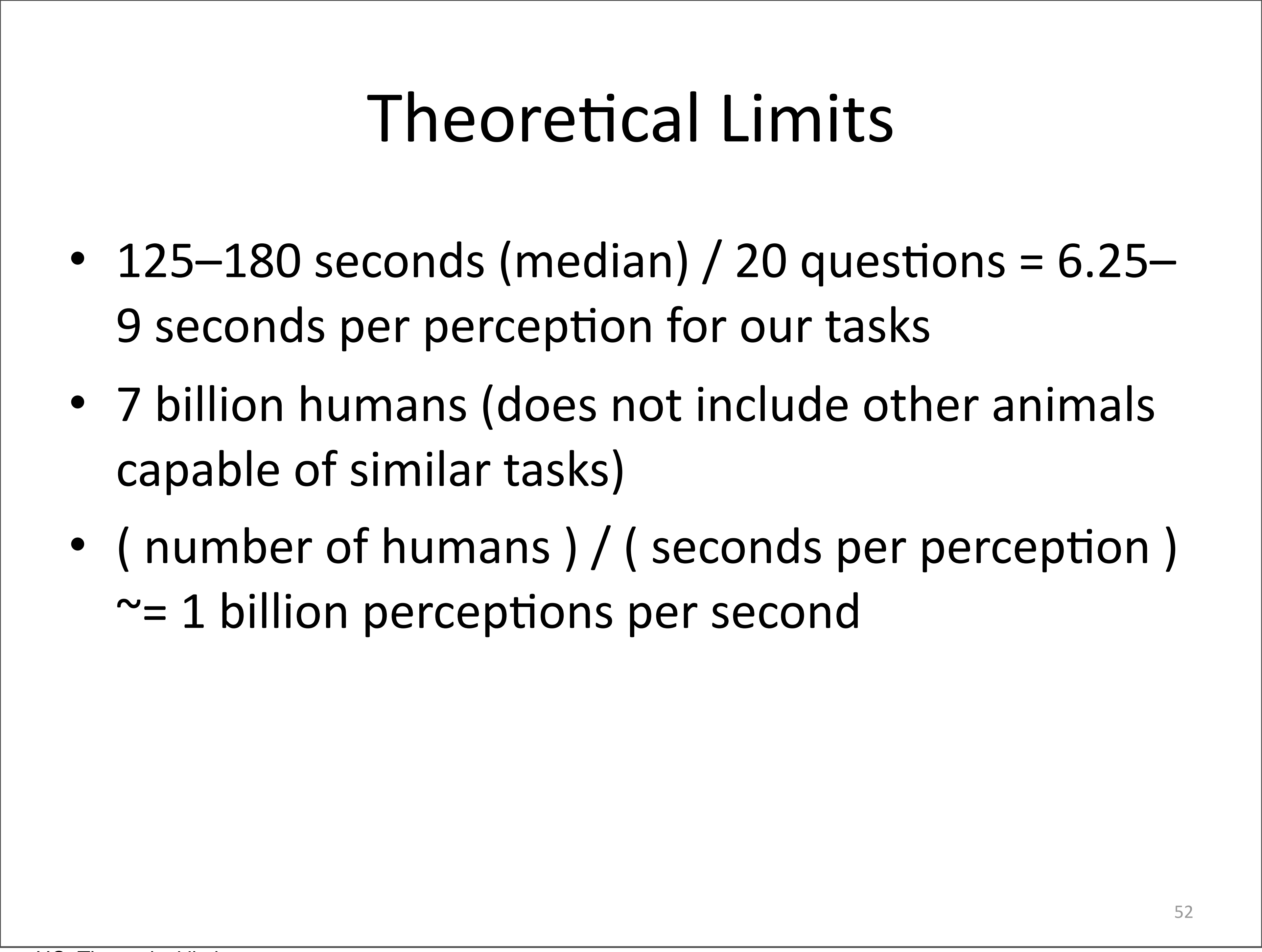
- Depth Layer Algorithm
  - automatic: [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009]
  - manual: [Oh et al. 2001; Ventura et al. 2009; Sykora et al. 2010]
- Normal Map Algorithm
  - manual: [Wu et al. 2008]
- Symmetry Map Algorithm
  - automatic: [Chen et al. 2007]

Related Work (5/6)

- History
  - “When Computers Were Human” [Grier 2005]
  - Genetic Algorithms
    - [Sims 1991]
    - Interactive Genetic Algorithm [Takagi 2001]
    - Human-Based Genetic Algorithms [Kosorukoff 2001]
    - Electric Sheep
  - Open Mind Initiative
  - collaborative filtering: [Goldberg et al. 1992; Adomavicius and Tuzhilin 2005]
- “Human Computation” [von Ahn 2005]

Related Work (6/6)

- Recent survey: [Quinn and Bederson 2011]
- Market properties:
  - [Ipeirotis 2010; Chilton et al. 2010; Faridani et al. 2011; Mason and Suri 2011; Mason and Watts 2010]
- Surface perception:
  - [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]
- Shape-from-Shading:
  - [Durou et al. 2008]



Theoretical Limits

- 125–180 seconds (median) / 20 questions = 6.25–9 seconds per perception for our tasks
- 7 billion humans (does not include other animals capable of similar tasks)
- (number of humans) / (seconds per perception) ~= 1 billion perceptions per second

+ HC: Theoretical limits.
    - 20 ms (.02 s) from brain to hand.
    - ?? ms (. ?? s) for perception
    - between 125 and 180 seconds (median) / 20 questions = 6.25--9 seconds per perception for our tasks.
    - 7 billion humans (does not include other animals capable of similar tasks)
    - (number of humans) / (seconds per perception) ~= 1 billion perceptions per second

There is an upper limit to human computation, which we can get by dividing the number of humans (~7 billion) by the time to record one perception (6.25 to 9 seconds in our examples), for a total of ~1 billion perceptions per second. That's 500 images per second if we want, say, per-pixel depth comparisons in a megapixel image and assume perfect humans: 1 billion perceptions per second / (1M pixels per image * 2 perceptions per pixel (4 neighboring pixels per pixel / 2 because the relationship is symmetric)) = 500 images per second.

Of course, this does not take into account better input (recording a perception may take as little as 1 second instead of 7) or, more importantly, Dog or Cat Computation.