Micro Perceptual Human Computation for Visual Tasks

Yotam Gingold
*George Mason University**

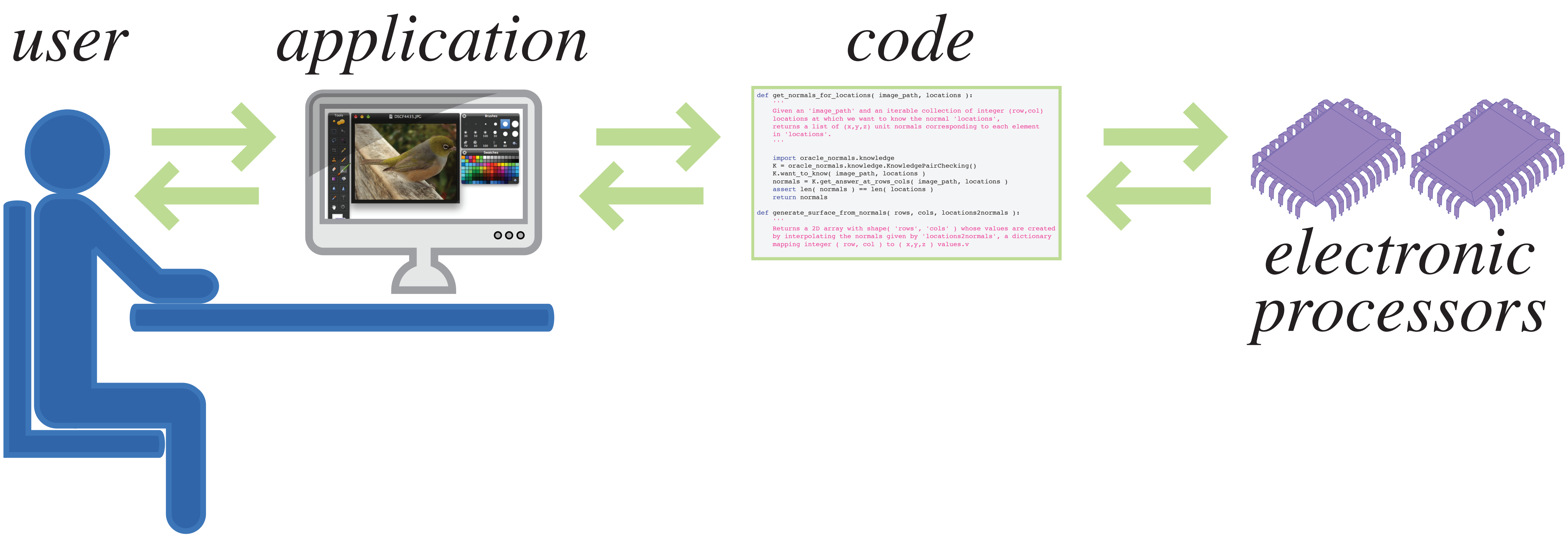
Ariel Shamir
Herzliya IDC

Daniel Cohen-Or
Tel-Aviv University

**Research performed while affiliated with Tel-Aviv University/Herzliya IDC/Rutgers/Columbia.*

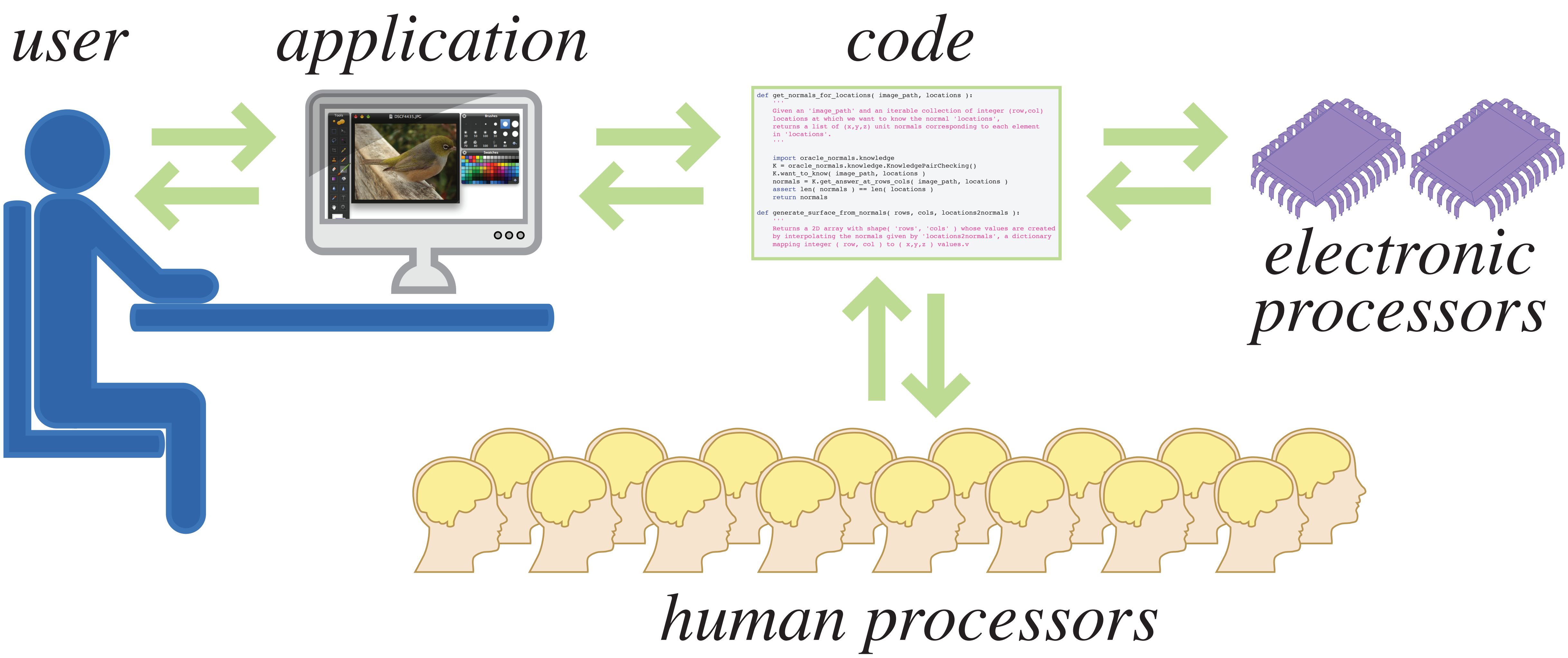
I'm going to be talking about "Human Computation"; the term was coined by Luis von Ahn in his 2005 PhD thesis, where he wrote, "We treat human brains as processors in a distributed system, each performing a small part of a massive computation."

Computation



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



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

With Human Computation, the code can run on a pool of human processors as well as electronic processors.

In this model, human processors are unskilled and isolated and there is high communication latency.

The Human Advantage

Visual Perception

Why? One human advantage is in visual or graphics perception and comprehension tasks.
Given a photograph or drawing, humans have visual perceptual abilities far superior to electronic computers. For example, answering: <click>x5

Humans do this with abilities they naturally possess.

The Human Advantage

Visual Perception

- What is in this photo?

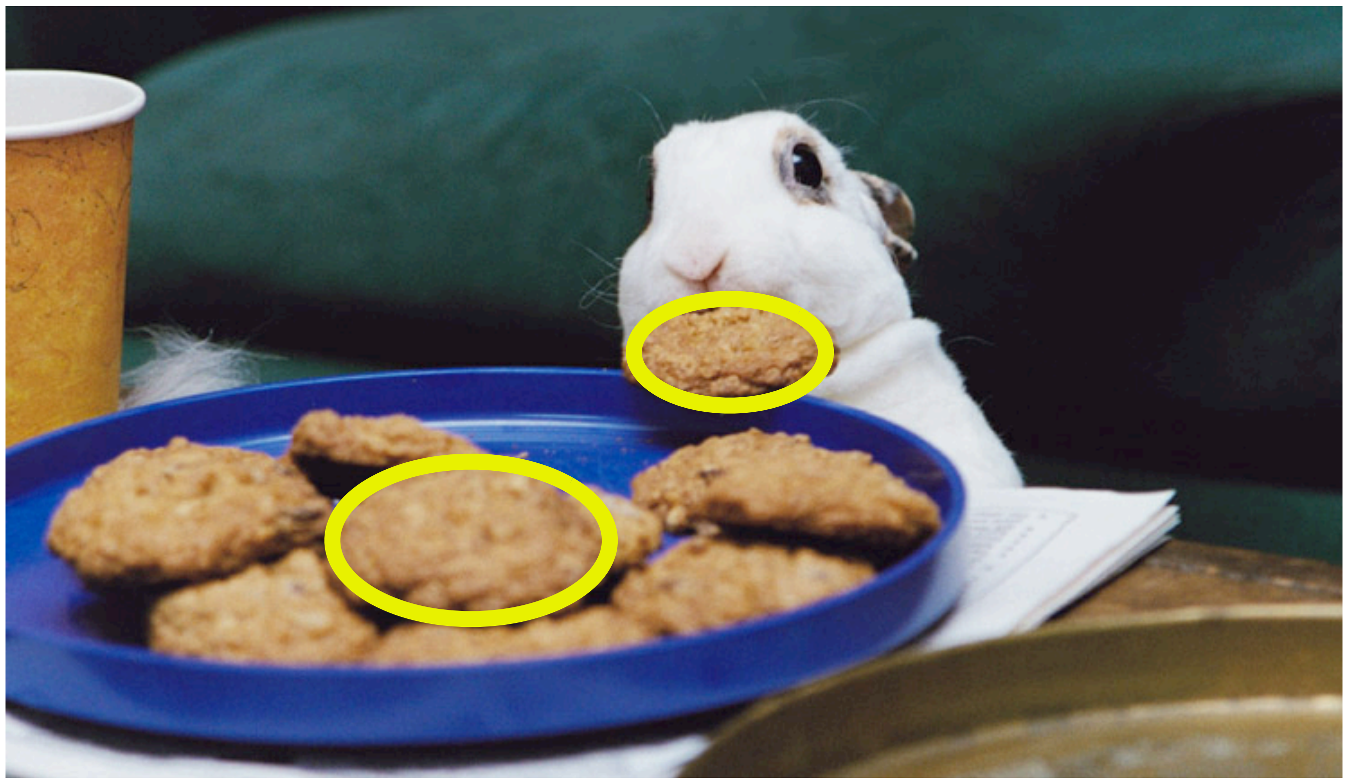
Why? One human advantage is in visual or graphics perception and comprehension tasks.
Given a photograph or drawing, humans have visual perceptual abilities far superior to electronic computers. For example, answering: <click>x5

Humans do this with abilities they naturally possess.

The Human Advantage

Visual Perception

- What is in this photo?
- Which object is farther away?



Why? One human advantage is in visual or graphics perception and comprehension tasks.
Given a photograph or drawing, humans have visual perceptual abilities far superior to electronic computers. For example, answering: <click>x5

Humans do this with abilities they naturally possess.

The Human Advantage

Visual Perception

- What is in this photo?
- Which object is farther away?
- Is this shape symmetric?

Why? One human advantage is in visual or graphics perception and comprehension tasks.
Given a photograph or drawing, humans have visual perceptual abilities far superior to electronic computers. For example, answering: <click>x5

Humans do this with abilities they naturally possess.

The Human Advantage

Visual Perception

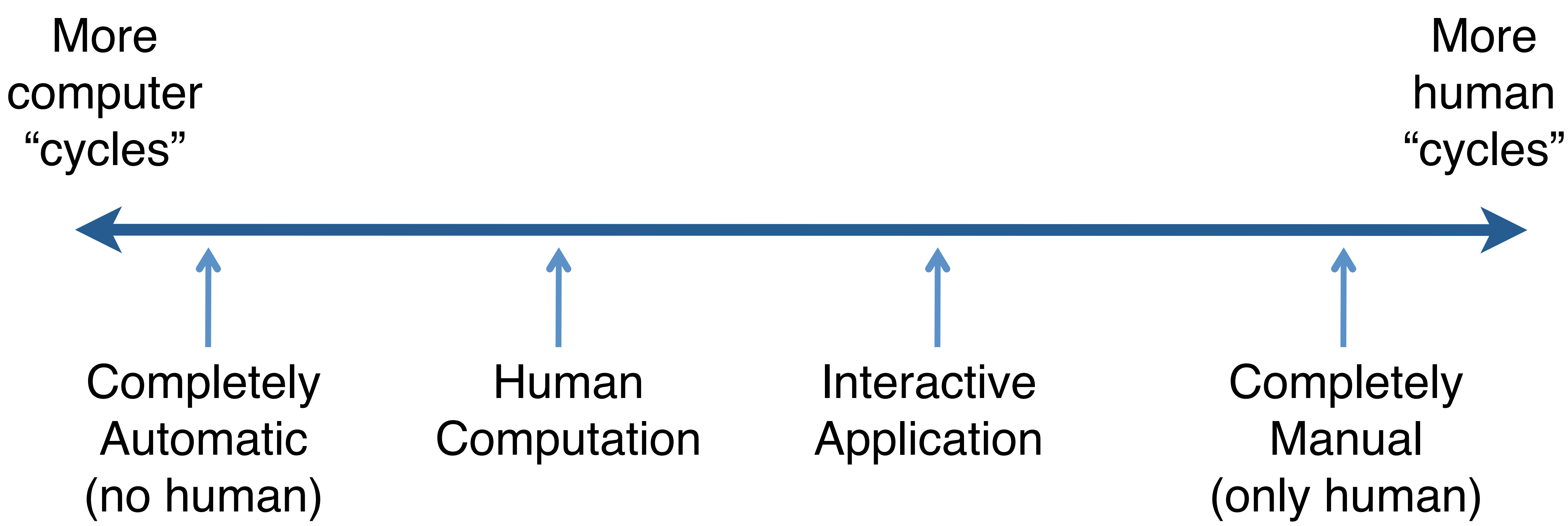
- What is in this photo?
- Which object is farther away?
- Is this shape symmetric?
- What is the surface orientation (normal)?

Why? One human advantage is in visual or graphics perception and comprehension tasks.
Given a photograph or drawing, humans have visual perceptual abilities far superior to electronic computers. For example, answering: <click>x5

Humans do this with abilities they naturally possess.

Range of Solutions

How much human and how much computer is involved?



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

<click> All the way on the **left**, we have **automatic algorithms** with no human computation.

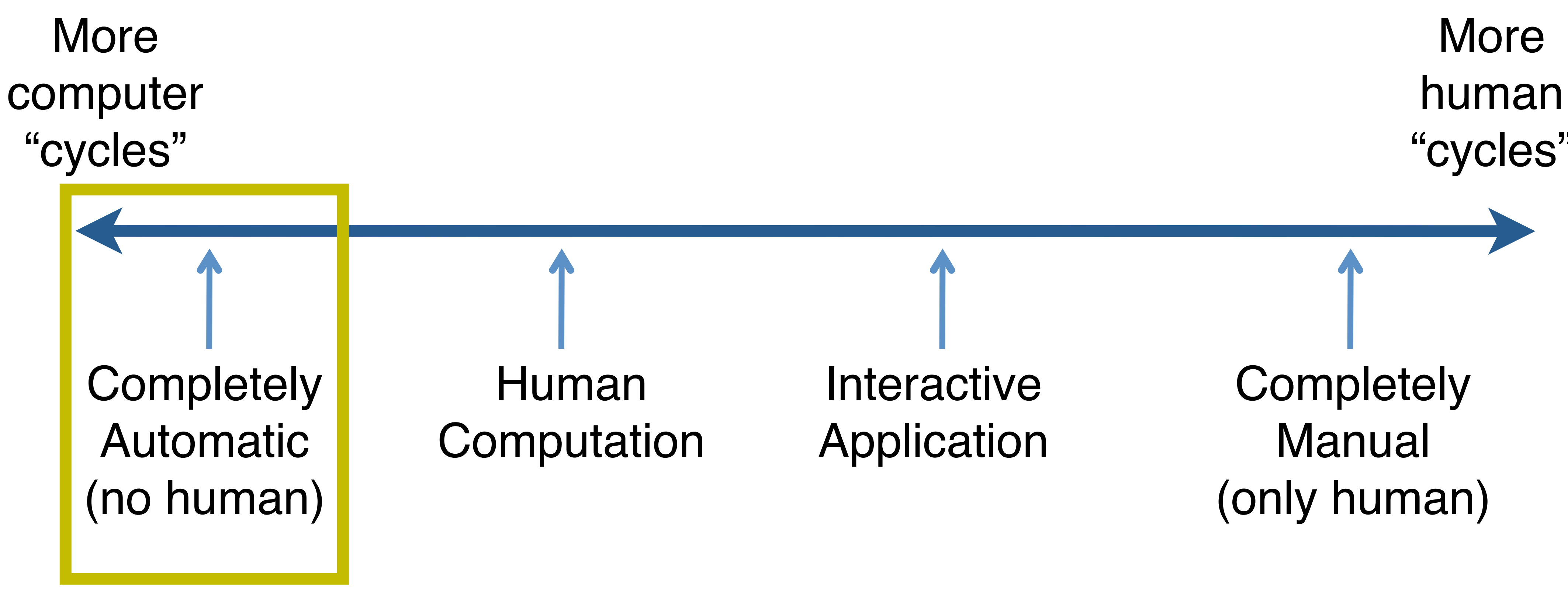
<click> All the way on the **right**, we have **fully manual solutions** with no electronic computation. The human is aware of the **high-level goal** and sets about to achieve it.

<click> To the left of fully manual solutions is interactive applications, where a **human is assisted by a computer**.

<click> And between that and automatic (machine computation) solutions, we have **computers being assisted by humans**. The humans are no longer aware of the high-level goal; instead, they are **given tasks to complete** by an electronic computer.

Range of Solutions

How much human and how much computer is involved?



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

<click> All the way on the **left**, we have **automatic algorithms** with no human computation.

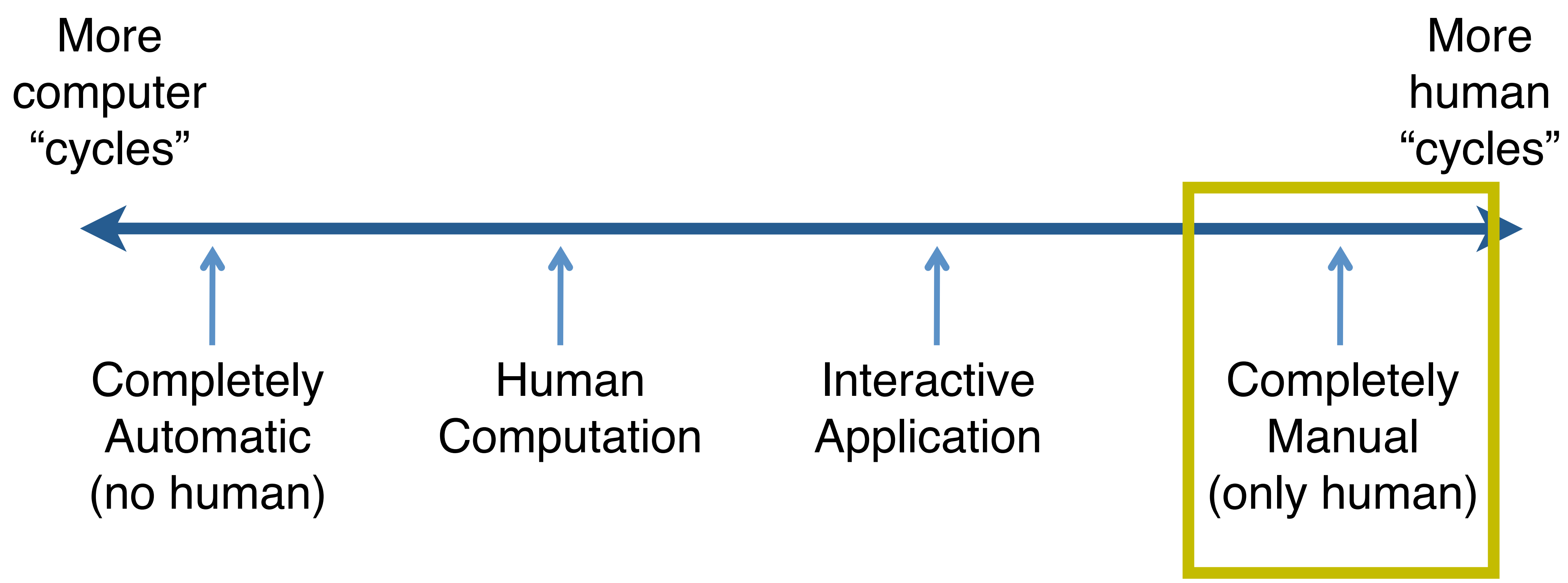
<click> All the way on the **right**, we have **fully manual solutions** with no electronic computation. The human is aware of the **high-level goal** and sets about to achieve it.

<click> To the left of fully manual solutions is interactive applications, where a **human is assisted by a computer**.

<click> And between that and automatic (machine computation) solutions, we have **computers being assisted by humans**. The humans are no longer aware of the high-level goal; instead, they are **given tasks to complete** by an electronic computer.

Range of Solutions

How much human and how much computer is involved?



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

<click> All the way on the **left**, we have **automatic algorithms** with no human computation.

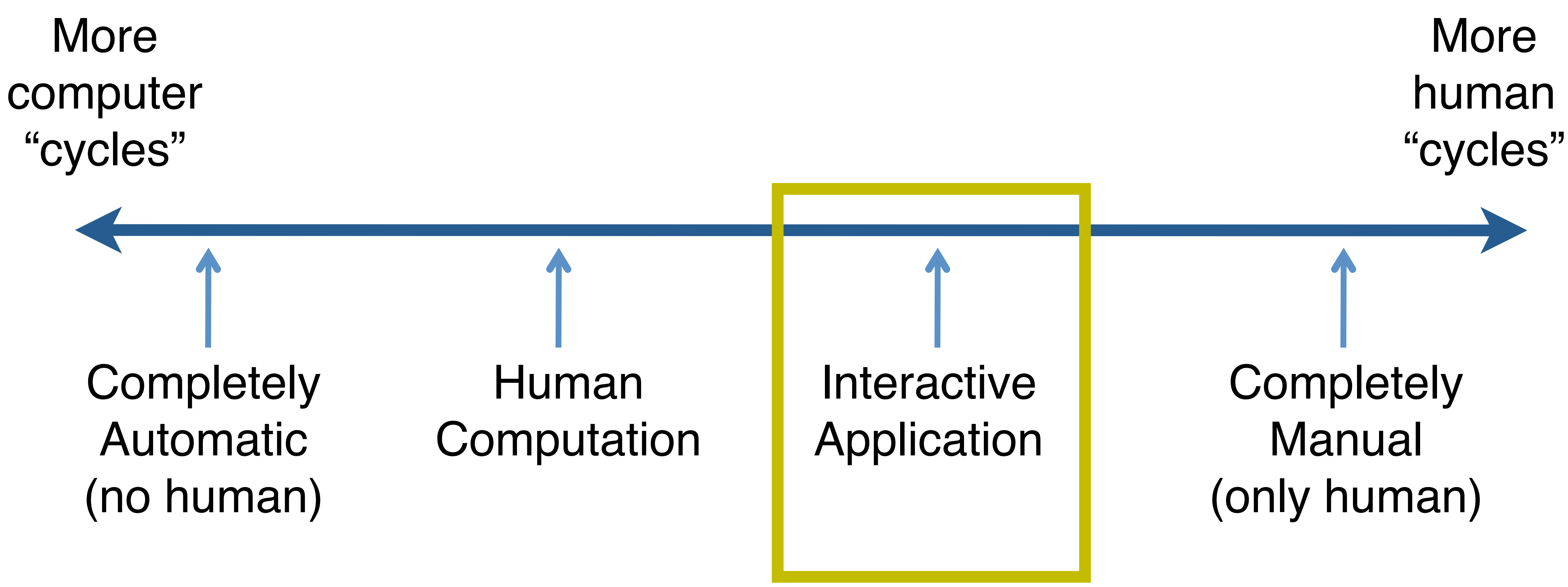
<click> All the way on the **right**, we have **fully manual solutions** with no electronic computation. The human is aware of the **high-level goal** and sets about to achieve it.

<click> To the left of fully manual solutions is interactive applications, where a **human is assisted by a computer**.

<click> And between that and automatic (machine computation) solutions, we have **computers being assisted by humans**. The humans are no longer aware of the high-level goal; instead, they are **given tasks to complete** by an electronic computer.

Range of Solutions

How much human and how much computer is involved?



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

<click> All the way on the **left**, we have **automatic algorithms** with no human computation.

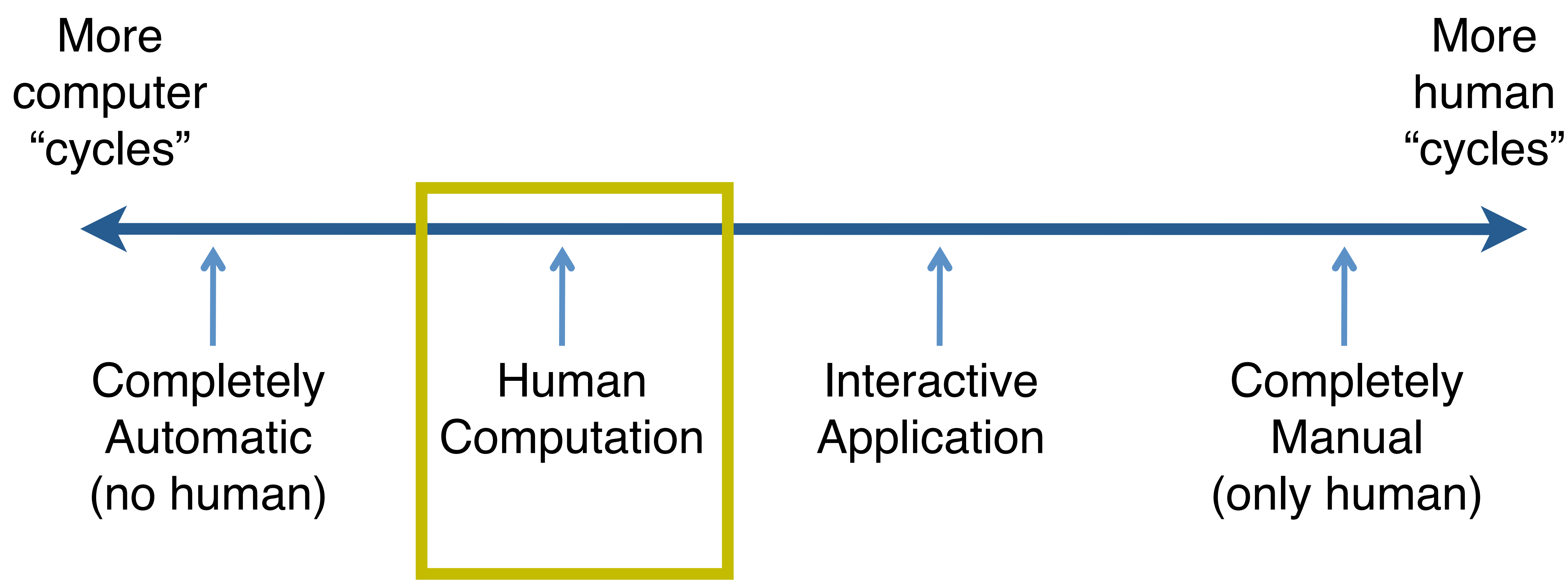
<click> All the way on the **right**, we have **fully manual solutions** with no electronic computation. The human is aware of the **high-level goal** and sets about to achieve it.

<click> To the left of fully manual solutions is interactive applications, where a **human is assisted by a computer**.

<click> And between that and automatic (machine computation) solutions, we have **computers being assisted by humans**. The humans are no longer aware of the high-level goal; instead, they are **given tasks to complete** by an electronic computer.

Range of Solutions

How much human and how much computer is involved?



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

<click> All the way on the **left**, we have **automatic algorithms** with no human computation.

<click> All the way on the **right**, we have **fully manual solutions** with no electronic computation. The human is aware of the **high-level goal** and sets about to achieve it.

<click> To the left of fully manual solutions is interactive applications, where a **human is assisted by a computer**.

<click> And between that and automatic (machine computation) solutions, we have **computers being assisted by humans**. The humans are no longer aware of the high-level goal; instead, they are **given tasks to complete** by an electronic computer.

Key Question

What is the minimum amount of information a human could provide in order to solve the original problem?

The key question when designing an algorithm with HC inside is...
<click>
<click>

Key Question

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.

The key question when designing an algorithm with HC inside is...
<click>
<click>

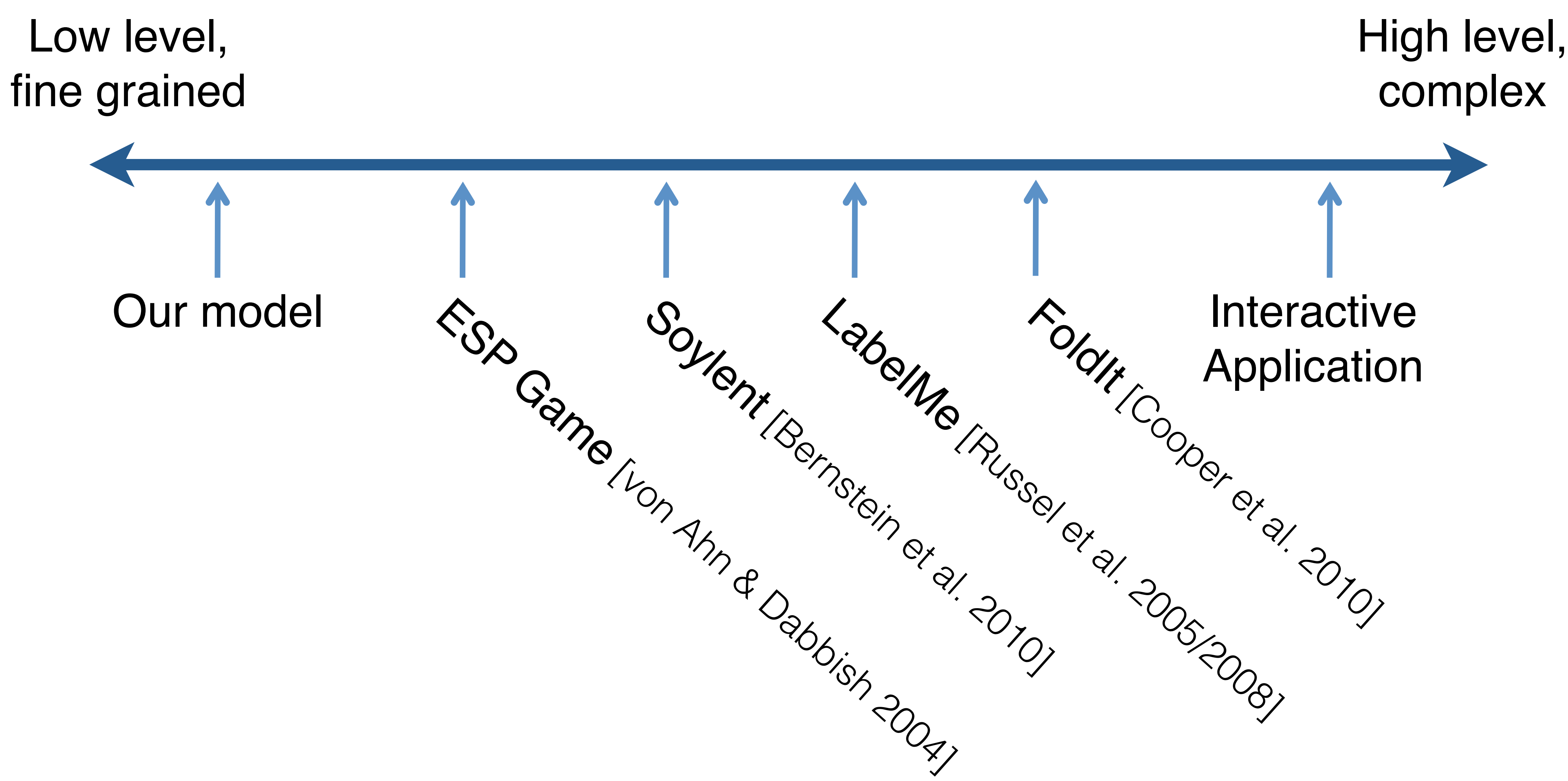
Key Question

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.
- Use only as much human computation as necessary, and no more than is sufficient.

The key question when designing an algorithm with HC inside is...
<click>
<click>

Type of Human Cycles

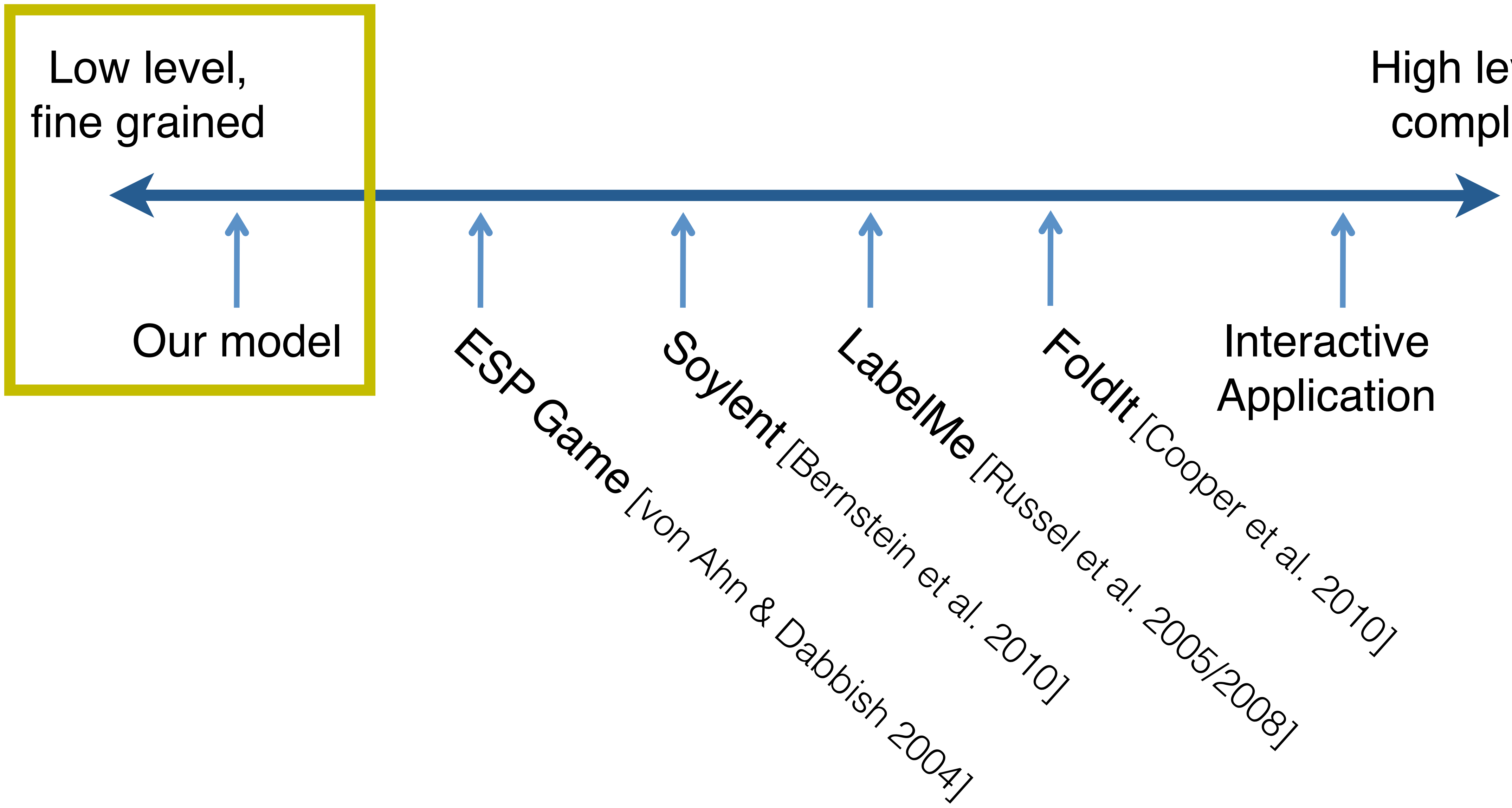


Human Computation **has been looked at before**. Here is an axis comparing the complexity of human cycles.
In contrast with many previous approaches where humans perform complex, high-level tasks, <click> **we go as far to the left as possible**, and advocate for extremely low-level “micro perceptual” queries. In our model, tasks are based on visual perceptual queries.

- <click> **No training or skill is needed**—any **sighted** human has good visual perception. Simple tasks help keep cost low, since we don't have to pay for training (up front or amortized).
- <click> There is **no dependency** (between tasks). Compared to typical distributed processing, HPs execute few operations per second and have high latency.
- <click> **Highly parallel**. In theory, with perfect parallelism, the algorithms I will show would take 3 minutes to complete.

So, our model can be summarized as **massive parallelism with extremely simple (training-free/instantaneous) visual queries**.
The human computation in our model is **online, not an offline process to generate training data.**

Type of Human Cycles

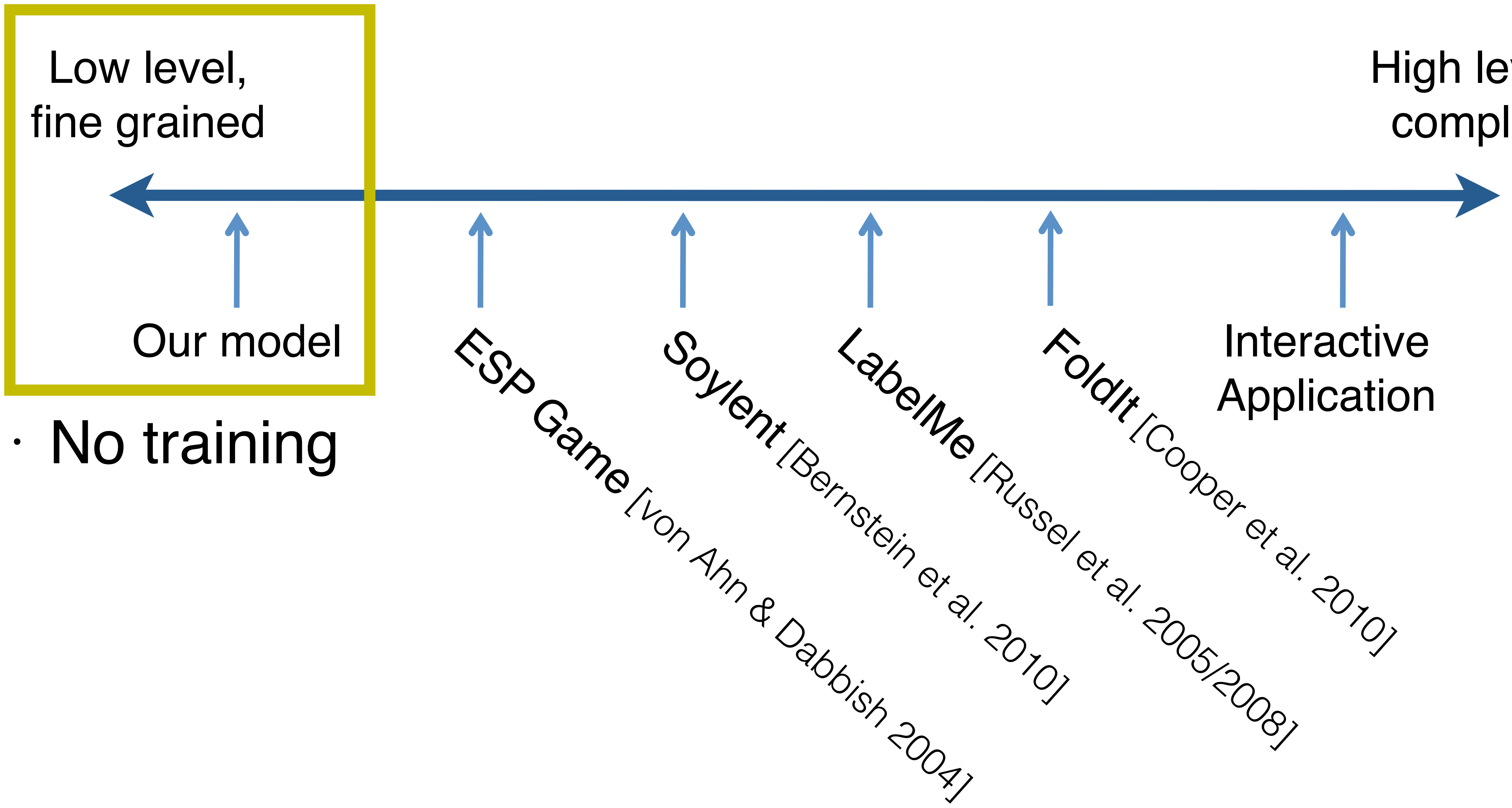


Human Computation **has been looked at before**. Here is an axis comparing the complexity of human cycles.
In contrast with many previous approaches where humans perform complex, high-level tasks, <click> **we go as far to the left as possible**, and advocate for extremely low-level "micro perceptual" queries. In our model, tasks are based on visual perceptual queries.

- <click> **No training or skill is needed**—any **sighted** human has good visual perception. Simple tasks help keep cost low, since we don't have to pay for training (up front or amortized).
- <click> There is **no dependency** (between tasks). Compared to typical distributed processing, HPs execute few operations per second and have high latency.
- <click> **Highly parallel**. In theory, with perfect parallelism, the algorithms I will show would take 3 minutes to complete.

So, our model can be summarized as **massive parallelism with extremely simple (training-free/instantaneous) visual queries**.
The human computation in our model is **online, not an offline process to generate training data.**

Type of Human Cycles

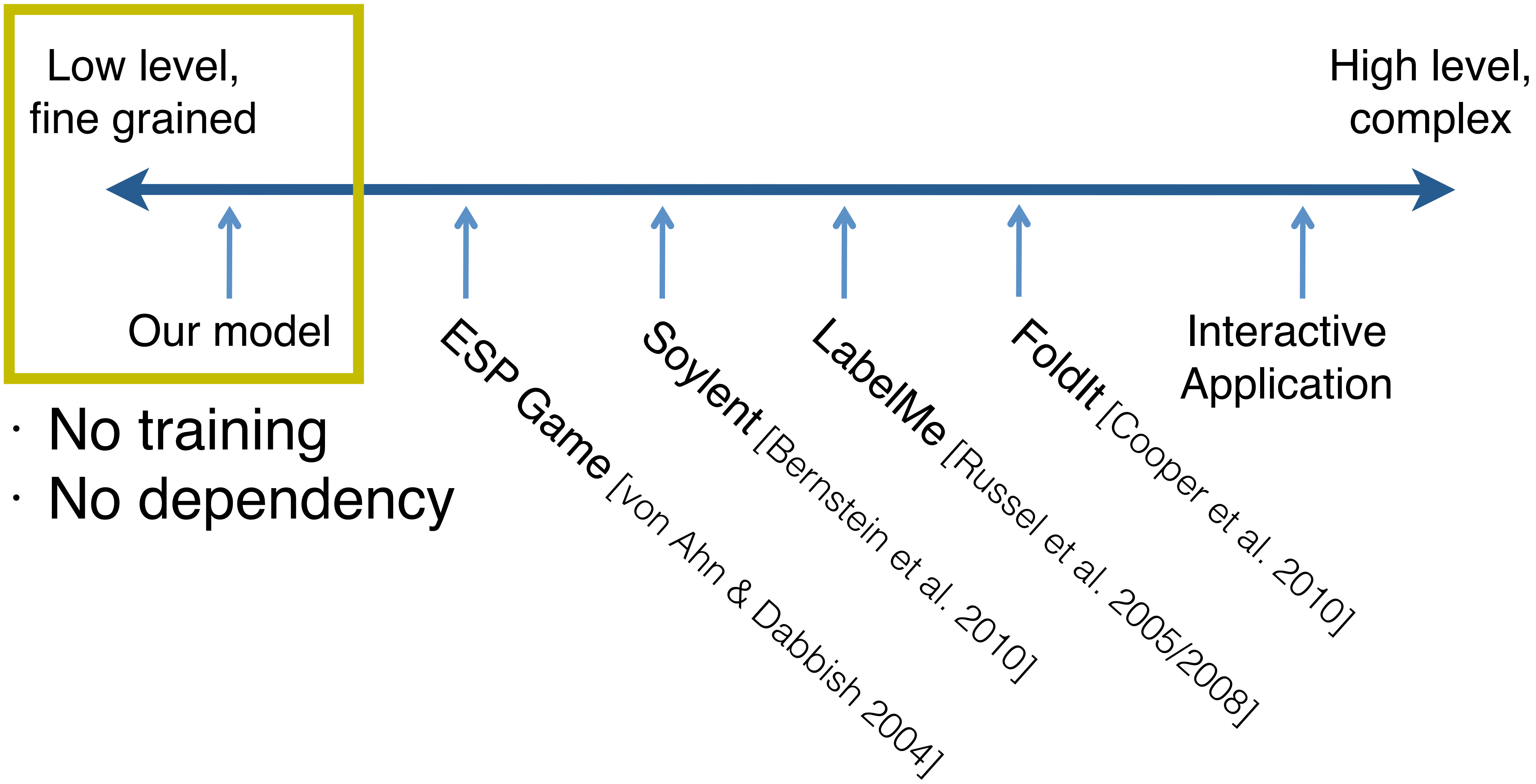


Human Computation **has been looked at before**. Here is an axis comparing the complexity of human cycles.
In contrast with many previous approaches where humans perform complex, high-level tasks, <click> **we go as far to the left as possible**, and advocate for extremely low-level "micro perceptual" queries. In our model, tasks are based on visual perceptual queries.

- <click> **No training or skill is needed**—any **sighted** human has good visual perception. Simple tasks help keep cost low, since we don't have to pay for training (up front or amortized).
- <click> There is **no dependency** (between tasks). Compared to typical distributed processing, HPs execute few operations per second and have high latency.
- <click> **Highly parallel**. In theory, with perfect parallelism, the algorithms I will show would take 3 minutes to complete.

So, our model can be summarized as **massive parallelism with extremely simple (training-free/instantaneous) visual queries**.
The human computation in our model is **online, not an offline process to generate training data.**

Type of Human Cycles



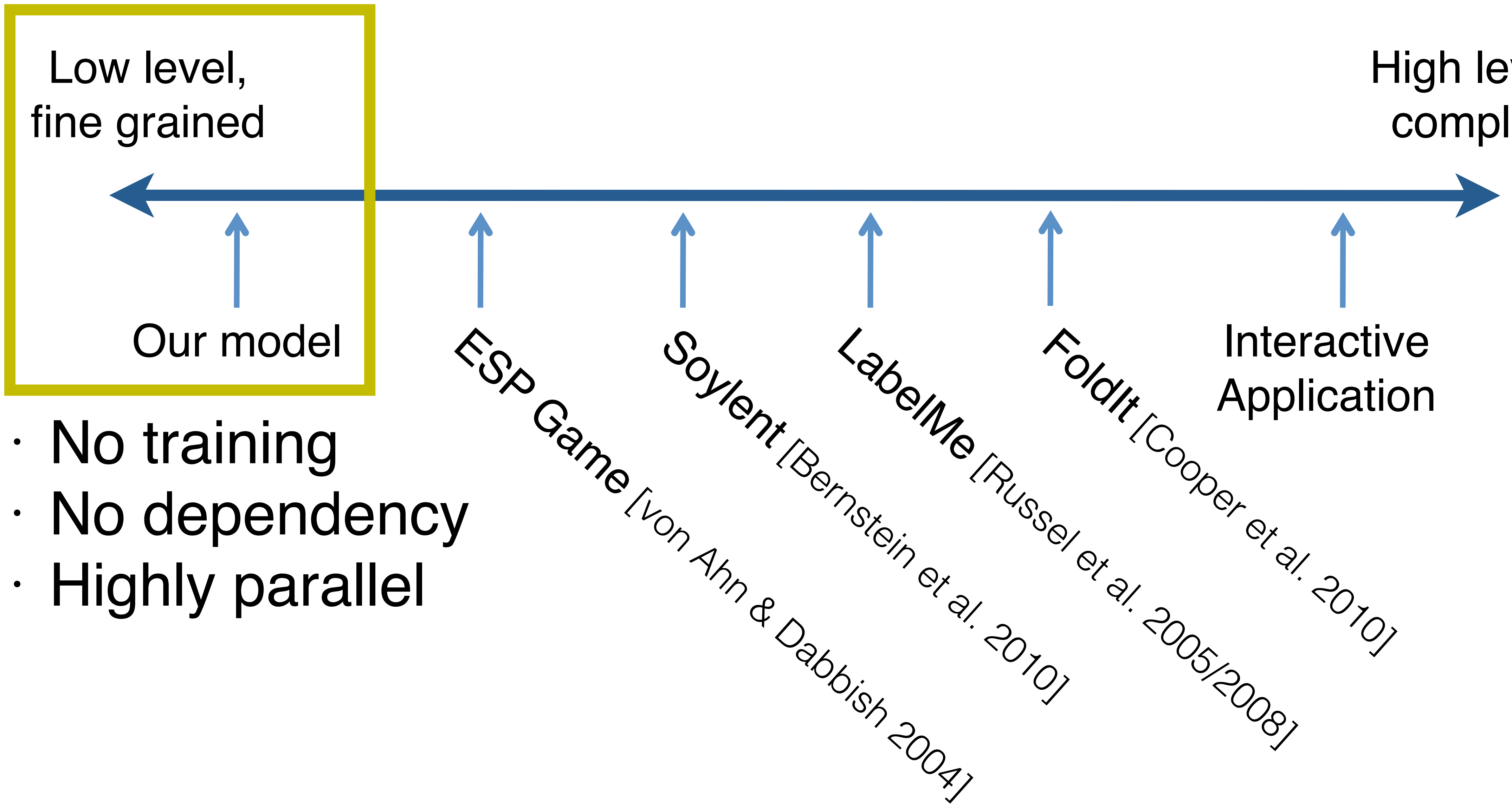
- No training
- No dependency

Human Computation **has been looked at before**. Here is an axis comparing the complexity of human cycles.
In contrast with many previous approaches where humans perform complex, high-level tasks, <click> **we go as far to the left as possible**, and advocate for extremely low-level "micro perceptual" queries. In our model, tasks are based on visual perceptual queries.

- <click> **No training or skill is needed**—any **sighted** human has good visual perception. Simple tasks help keep cost low, since we don't have to pay for training (up front or amortized).
- <click> There is **no dependency** (between tasks). Compared to typical distributed processing, HPs execute few operations per second and have high latency.
- <click> **Highly parallel**. In theory, with perfect parallelism, the algorithms I will show would take 3 minutes to complete.

So, our model can be summarized as **massive parallelism with extremely simple (training-free/instantaneous) visual queries**.
The human computation in our model is **online, not an offline process to generate training data.**

Type of Human Cycles



- No training
- No dependency
- Highly parallel

Human Computation **has been looked at before**. Here is an axis comparing the complexity of human cycles.
In contrast with many previous approaches where humans perform complex, high-level tasks, <click> **we go as far to the left as possible**, and advocate for extremely low-level “micro perceptual” queries. In our model, tasks are based on visual perceptual queries.

- <click> **No training or skill is needed**—any **sighted** human has good visual perception. Simple tasks help keep cost low, since we don't have to pay for training (up front or amortized).
- <click> There is **no dependency** (between tasks). Compared to typical distributed processing, HPs execute few operations per second and have high latency.
- <click> **Highly parallel**. In theory, with perfect parallelism, the algorithms I will show would take 3 minutes to complete.

So, our model can be summarized as **massive parallelism with extremely simple (training-free/instantaneous) visual queries**.
The human computation in our model is **online, not an offline process to generate training data.**

Guidelines for Designing Micro-Tasks

There are several requirements for a micro-task in our model of human computation.

- **Near-instantaneous**.
- **Well-defined** so we can program with it. An analogy 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. (We will see an example of this later.)

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

Guidelines for Designing Micro-Tasks

Task must be simple (instantaneous)

There are several requirements for a micro-task in our model of human computation.

- **Near-instantaneous**.
- **Well-defined** so we can program with it. An analogy 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. (We will see an example of this later.)

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

Guidelines for Designing Micro-Tasks

Task must be simple (instantaneous)

Task must be specific (well-defined)

There are several requirements for a micro-task in our model of human computation.

- **Near-instantaneous**.
- **Well-defined** so we can program with it. An analogy 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. (We will see an example of this later.)

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

Guidelines for Designing Micro-Tasks

Task must be simple (instantaneous)

Task must be specific (well-defined)

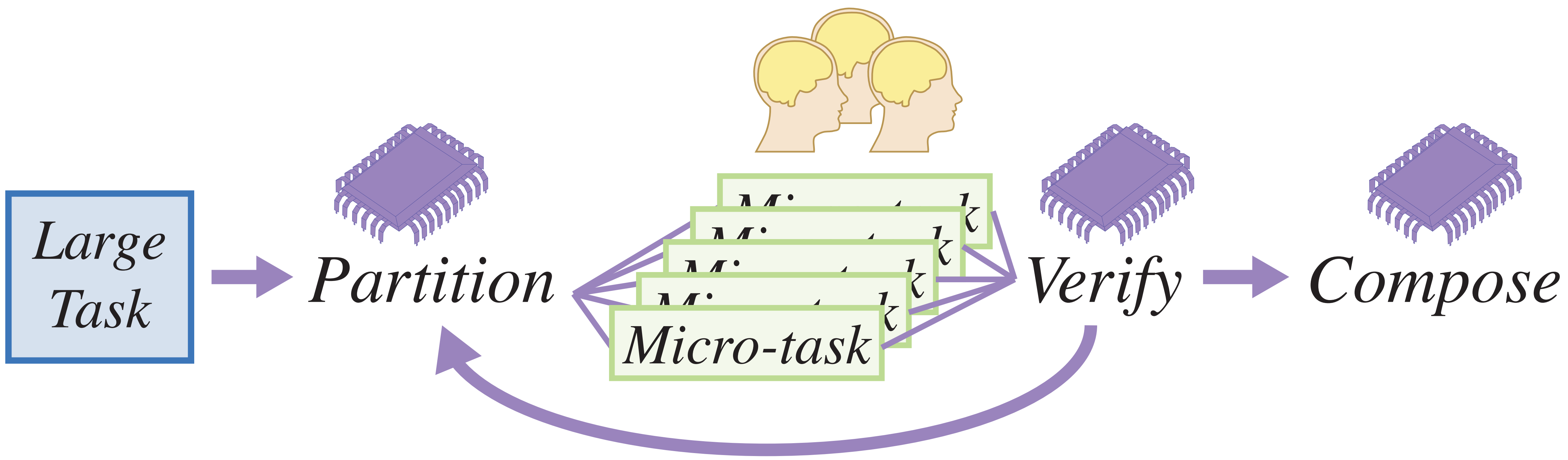
Task must be reliable (humans can do it)

There are several requirements for a micro-task in our model of human computation.

- **Near-instantaneous**.
- **Well-defined** so we can program with it. An analogy 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. (We will see an example of this later.)

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

Algorithm Design Pattern



This is the design pattern we use for our algorithms.

A large input task is **partitioned** by an electronic processor into a large number of parallel human micro-tasks.
The perceptual micro-tasks run on a **large pool** of distributed human processors.
An electronic processor collects and **verifies** the output of the human micro-tasks as they arrive, **potentially dispatching** new human micro-tasks if the micro-tasks fail verification or indicate that the large task should be partitioned further.
Finally, the electronic processor **composes** all of the human micro-task output and generates the algorithm's final output.

Issues

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)

- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!

humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:

<click> We send batches of six HC queries.

<click> We include 4 questions with known answers (sentinel operations or "gold data").

<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)

- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!

humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:

<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)

- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!

humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:

<click> We send batches of six HC queries.

<click> We include 4 questions with known answers (sentinel operations or "gold data").

<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Efficiency

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)
- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!
humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).
It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:
<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Efficiency Massive parallelism
Extremely simple visual queries

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)
- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!
humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).
It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:
<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation	money (via Amazon Mechanical Turk)
Efficiency	Massive parallelism Extremely simple visual queries
Quality Control	

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)

- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!

humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:

<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Efficiency Massive parallelism
Extremely simple visual queries

Quality Control Batches:

1	2	3	4	5	6

queries

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)

- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!

humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:

<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Efficiency Massive parallelism
Extremely simple visual queries

Quality Control Batches:

1	2	3	4	5	6	1	2	3	4

queries — queries with known answers

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)
- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!
humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).
It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:
<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Efficiency Massive parallelism
Extremely simple visual queries

Quality Control Batches:

1	2	3	4	5	6	1	2	3	4	1	2	3	4	5	6	1	2	3	4

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)
- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!
humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).
It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:
<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Issues

Motivation money (via Amazon Mechanical Turk)

Efficiency Massive parallelism
Extremely simple visual queries

Quality Control Batches:

4	3	2	6	4	1	2	6	2	4	1	3	3	5	1	3	4	1	5	2

Here is a summary of the issues that arise once humans are in the loop.

<click> Motivation / Incentives (money or fun)

- <click> We pay human processors with an online labor market, specifically Amazon's Mechanical Turk. AMT lets you advertise a job (description, payment amount, time estimate). It has a **large pool of workers** — tens or hundreds of thousands. It **has an API**, so you can program it.
- **If you can make a game** out of your human computation, it could become free. I like to 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.**

<click> Efficiency means using as little HC as possible. **HC is slow**, so this is typically the **bottleneck**. <click> We opt for massive parallelism with extremely simple visual queries in our examples.

<click> Quality control is important!

humans are: **noisy/inconsistent/non-deterministic**. depending on their motivations, they may **cheat**. humans also have **varying perceptual biases** % (perceptual biases as in depth scaling or bas-relief [Koenderink et al. 1992; Belheumer et al. 1997; Koenderink et al. 2001]).

It wouldn't be an algorithm if there were a researcher (or an expert) in the loop accepting and rejecting human computation. In our algorithms, to determine quality automatically:

<click> We send batches of six HC queries.
<click> We include 4 questions with known answers (sentinel operations or "gold data").
<click> We duplicate every query and <click> randomize the order. Duplication and randomization are commonly used in perceptual experiments. We also send the same batch to multiple HPs.

Three Example Algorithms

Given an image, create

I will show three micro perceptual human computation algorithms:

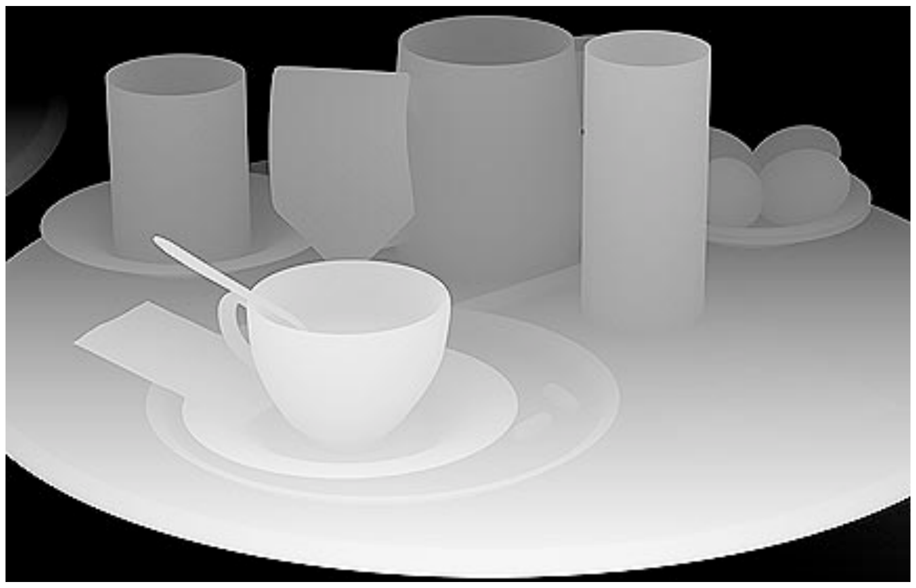
- recovering depth layers from a photograph (useful for object insertion/removal, de-hazing, depth of field, retargeting)
- normal map (useful for relighting or surface reconstruction)
- bilateral symmetry map (useful for edit propagation, retargeting)

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

Three Example Algorithms

Given an image, create

- depth layers



I will show three micro perceptual human computation algorithms:

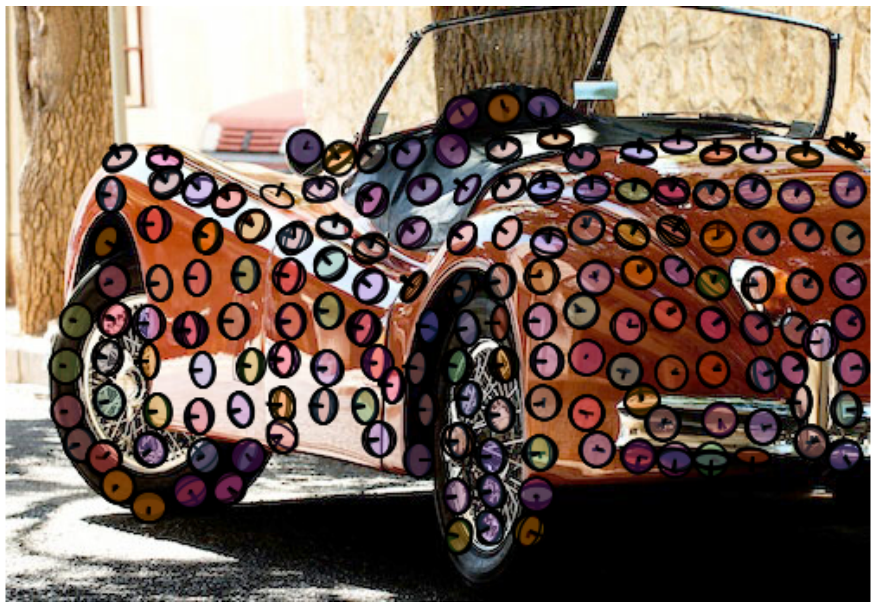
- recovering depth layers from a photograph (useful for object insertion/removal, de-hazing, depth of field, retargeting)
- normal map (useful for relighting or surface reconstruction)
- bilateral symmetry map (useful for edit propagation, retargeting)

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

Three Example Algorithms

Given an image, create

- depth layers
- a normal map



[Pedro Ribeiro Simões]

I will show three micro perceptual human computation algorithms:

- recovering depth layers from a photograph (useful for object insertion/removal, de-hazing, depth of field, retargeting)
- normal map (useful for relighting or surface reconstruction)
- bilateral symmetry map (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 for machine learning algorithms.

Three Example Algorithms

Given an image, create

- depth layers
- a normal map
- a bilateral symmetry map

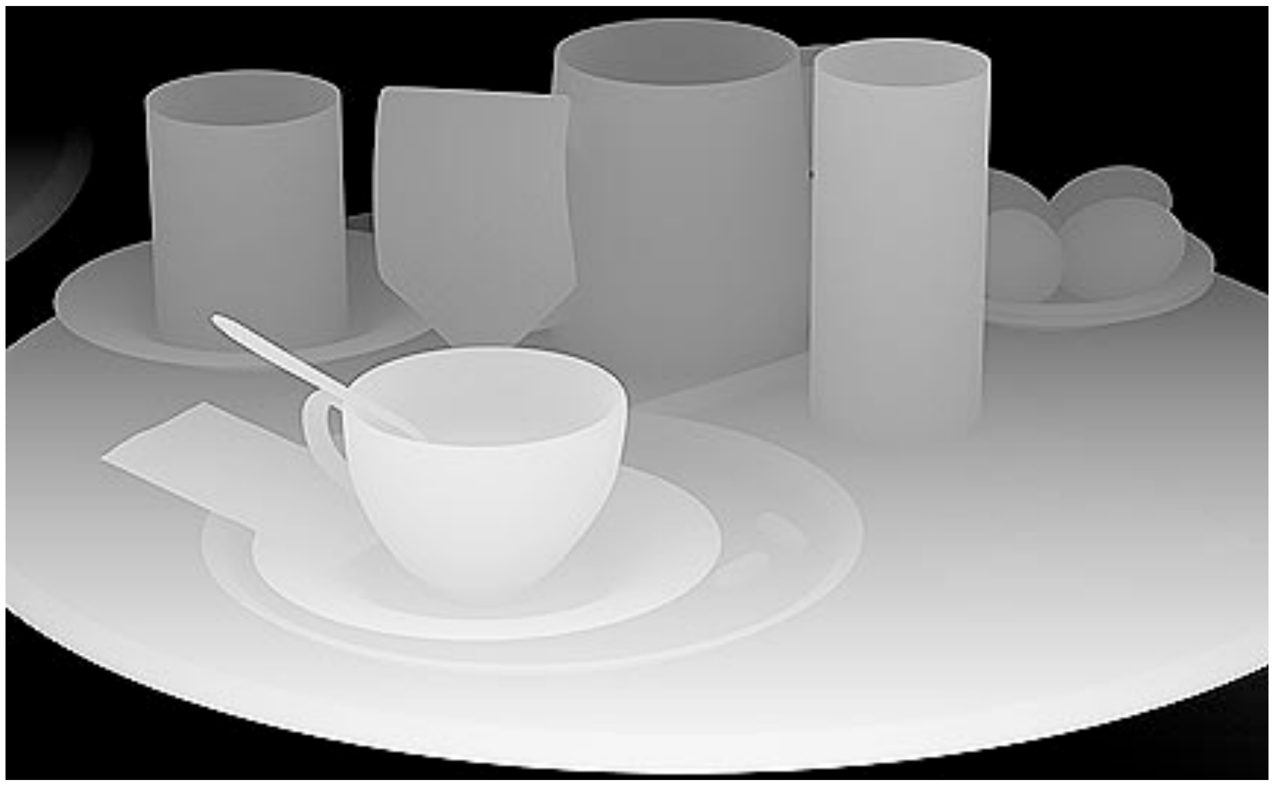
[flickr user dalbera]

I will show three micro perceptual human computation algorithms:

- recovering depth layers from a photograph (useful for object insertion/removal, de-hazing, depth of field, retargeting)
- normal map (useful for relighting or surface reconstruction)
- bilateral symmetry map (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 for machine learning algorithms.

Algorithm 1: Depth Layers



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

Here is an example, synthetic, depth map. A depth map stores, for each pixel, its distance from the camera. The discontinuities in the depth map allow us to infer the depth layering of objects in the image.

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.

Automatic methods

e.g. [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009]

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... <click> <click>
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Automatic methods

e.g. [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009]

Depth increases in the up direction

Make3D [Saxena et al. 2009]

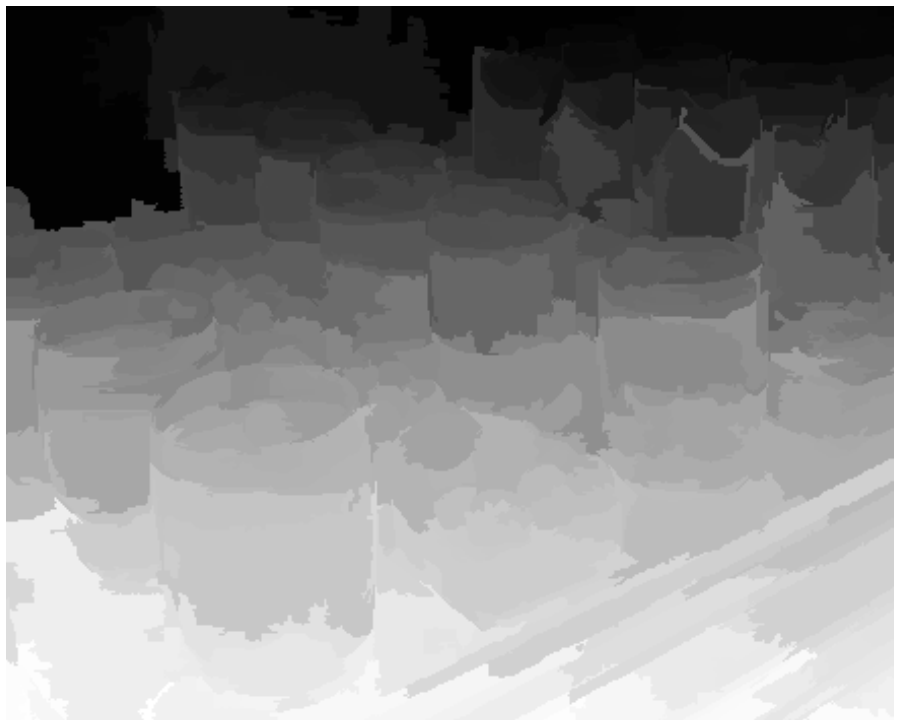
We aim to be more robust than automatic techniques. % [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009].
For example, Make3D [Saxena et al. 2009] seems to assume that... <click> <click>
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Automatic methods

e.g. [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009]

Depth increases in the up direction

Color similarity implies depth similarity



Make3D [Saxena et al. 2009]

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... <click> <click>
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Automatic methods

e.g. [Hoiem et al. 2005; Assa and Wolf 2007; Saxena et al. 2009]

Depth increases in the up direction

Color similarity implies depth similarity

Not always correct

Make3D [Saxena et al. 2009]

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... <click> <click>
This is not always correct.
<click>
And some images are very, very challenging, such as artwork.

Automatic methods

Depth increases in the up direction

Color similarity implies depth similarity

Not always correct

Some images are very challenging (art)

[Hiroshige]

And some images are very, very challenging, such as artwork.

% There are also 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 a pixel?"

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 a 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 a pixel?"

- Too fine, can be ambiguous

Ask "What is the depth of an object?"

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 a pixel?"

- Too fine, can be ambiguous

Ask "What is the depth of an object?"

- Segmentation is too difficult

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 a pixel?"

- Too fine, can be ambiguous

Ask "What is the depth of an object?"

- Segmentation is too difficult

Ask "What is the depth of a patch in the image?"

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 a pixel?”

- Too fine, can be ambiguous

Ask “What is the depth of an object?”

- Segmentation is too difficult

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

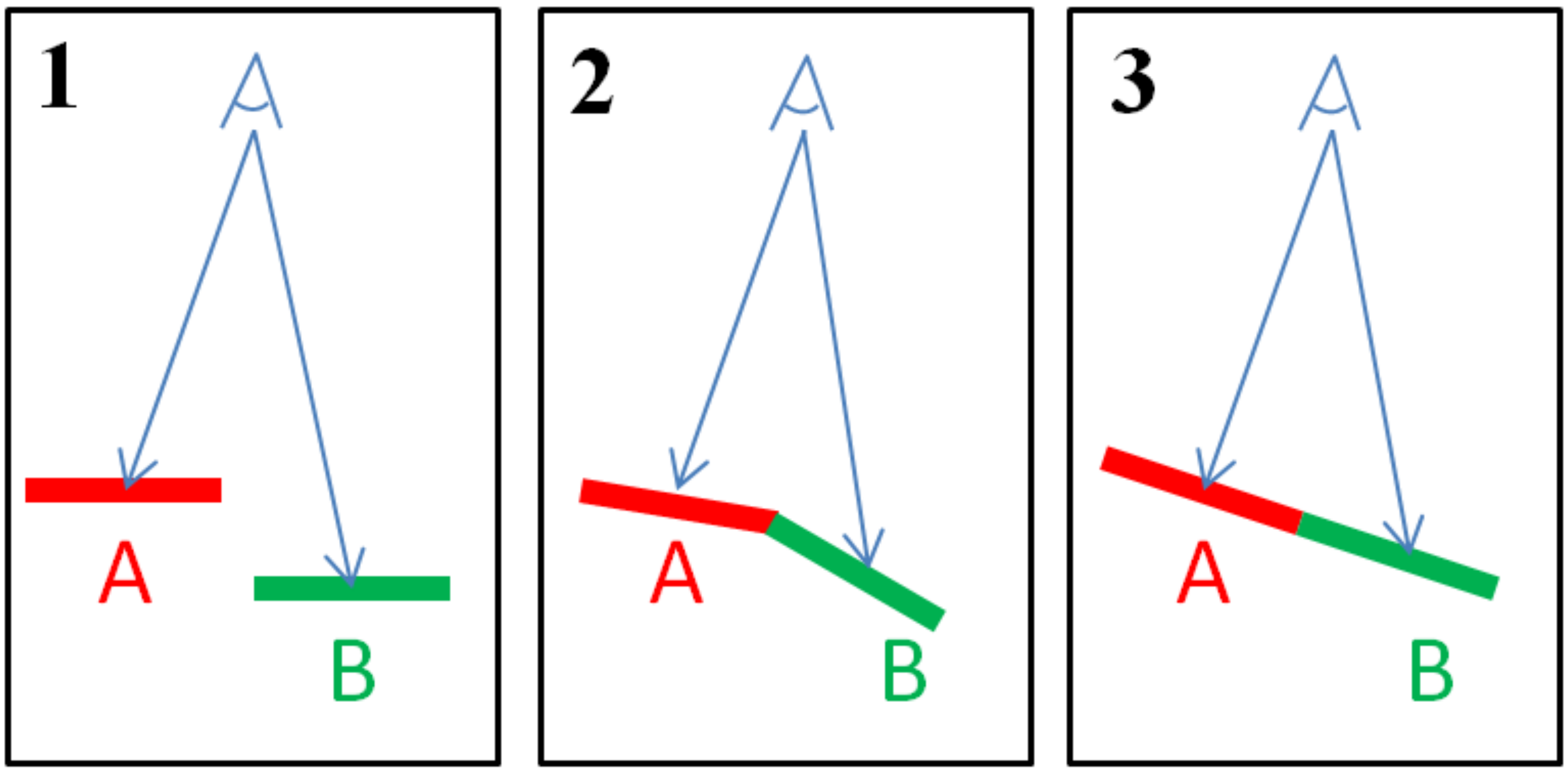
This is reliable ([Koenderink 2001]).
<click>
However, it's still ambiguous:
Case 1 depth jump between A and B
Case 2 non-smooth depth change between A and B
Case 3 smooth depth change between A and B

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:



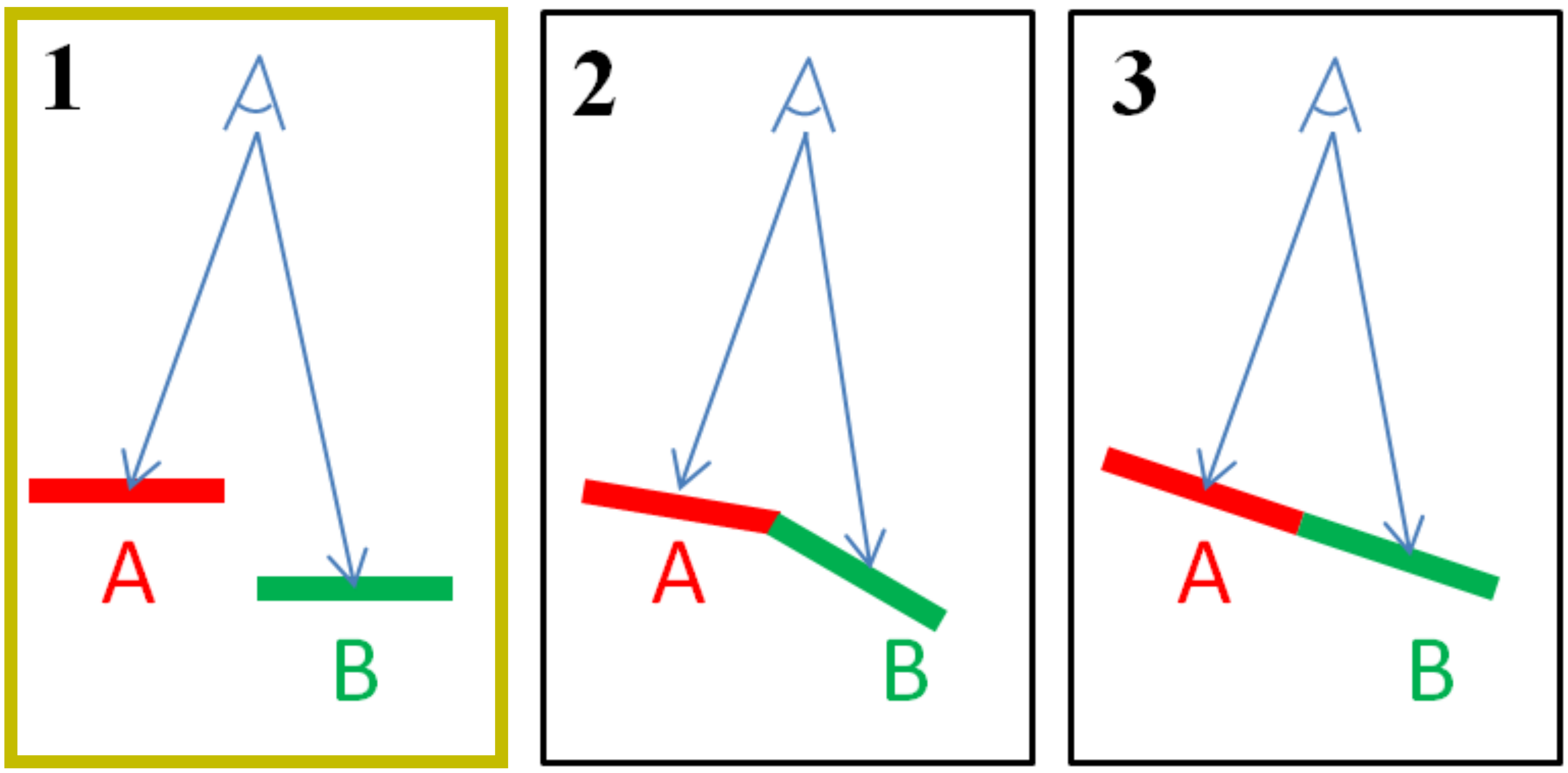
This is reliable ([Koenderink 2001]).
<click>
However, it's still ambiguous:
Case 1 depth jump between A and B
Case 2 non-smooth depth change between A and B
Case 3 smooth depth change between A and B

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:



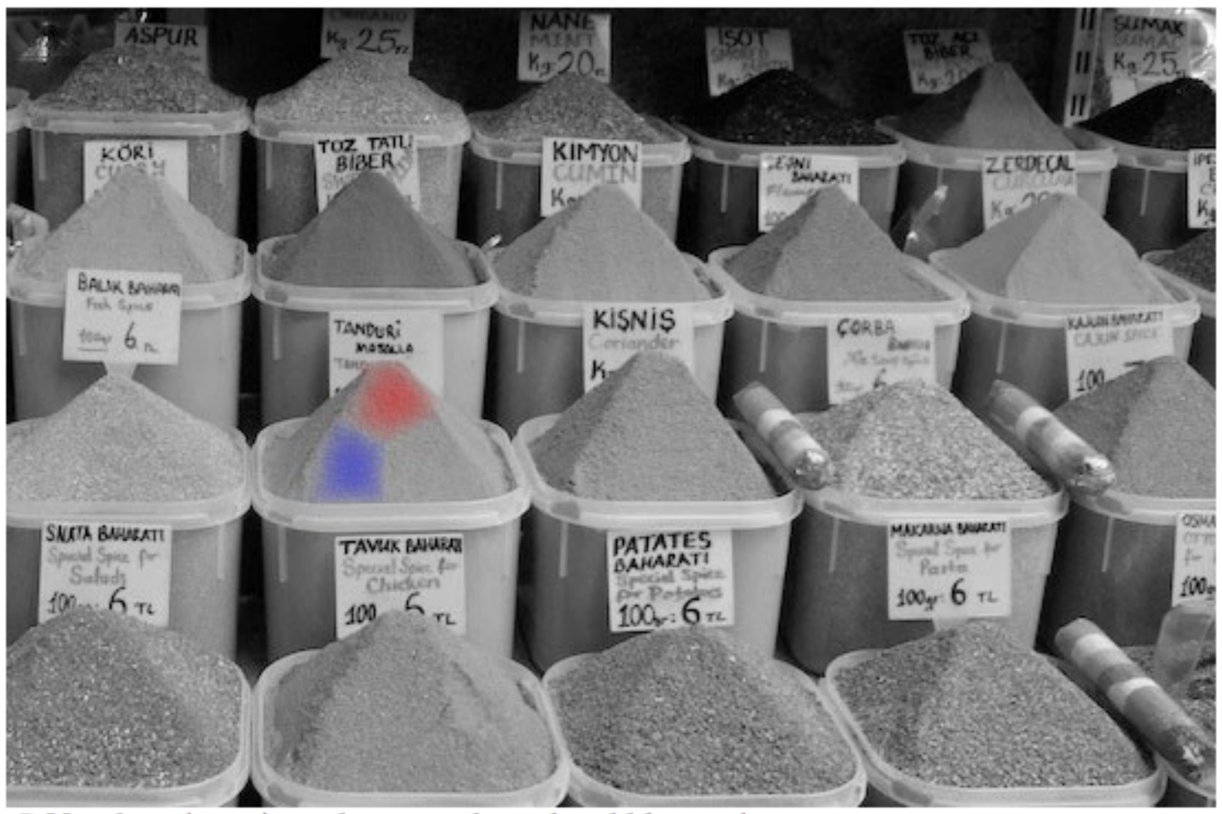
This is reliable ([Koenderink 2001]).
<click>
However, it’s still ambiguous:
Case 1 depth jump between A and B
Case 2 non-smooth depth change between A and B
Case 3 smooth depth change between A and B

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

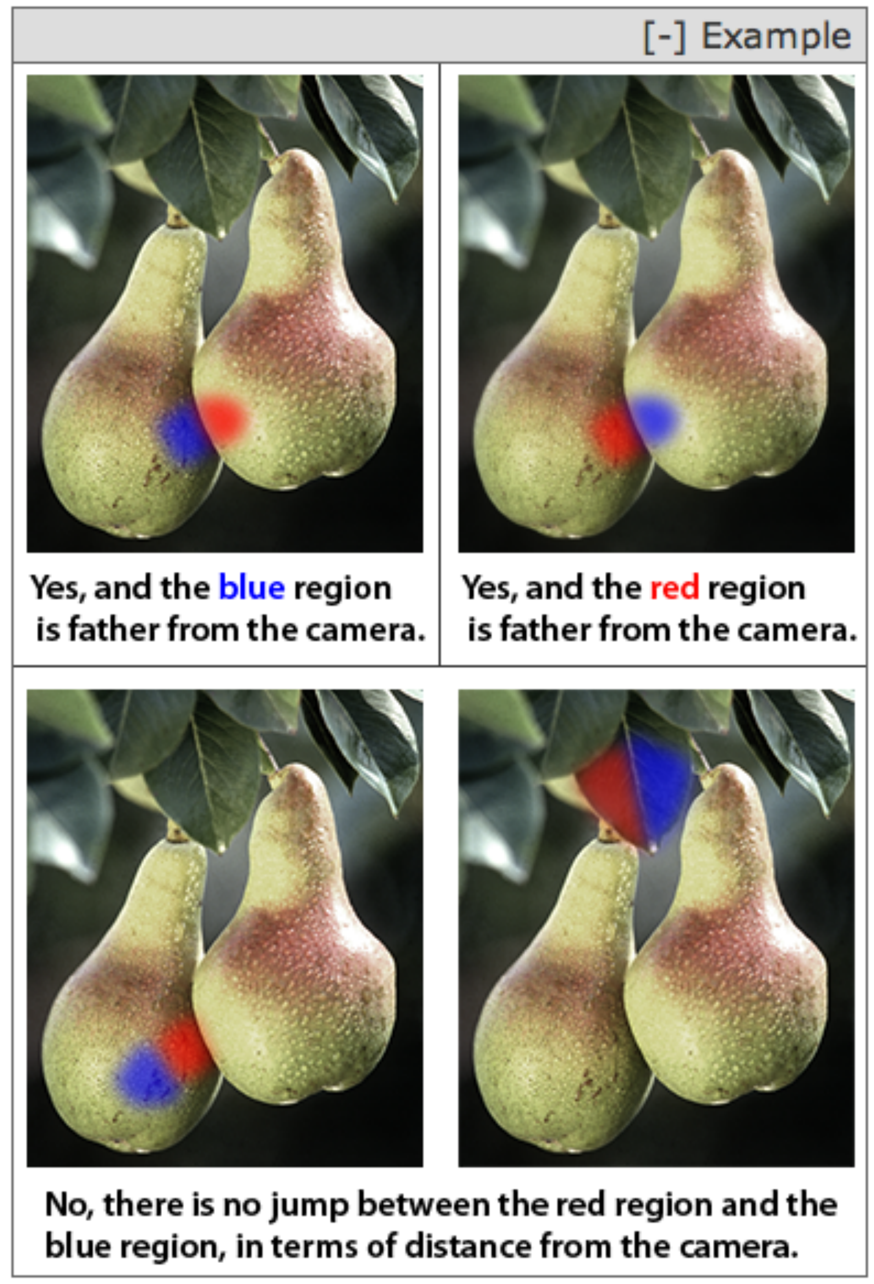
Our Micro-Task

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.

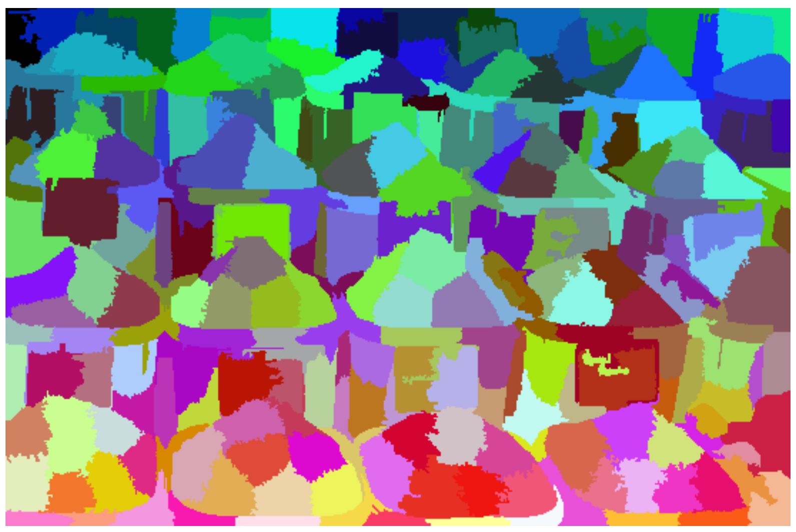


- ○ **No**, there is no jump between the red and blue regions.
- ○ Yes, and the **blue** region is farther from the camera.
- ○ Yes, and the **red** region is farther from the camera.

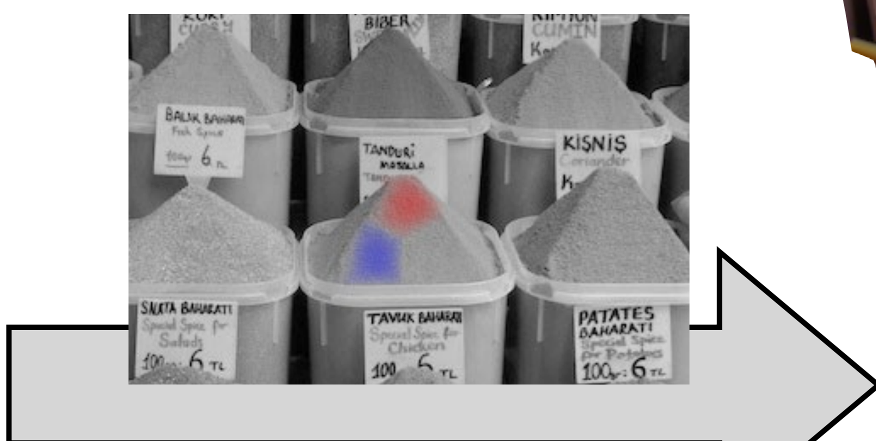


Here is the micro-task that human processors actually see.
Note the static example in the corner. That’s it, there is no other training.

Algorithm



Patch
Segmentation



Micro-tasks (human)

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

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

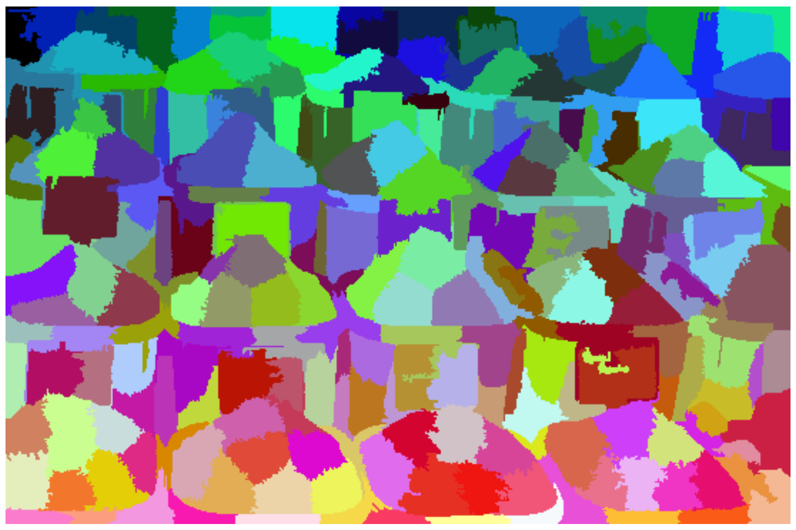
Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

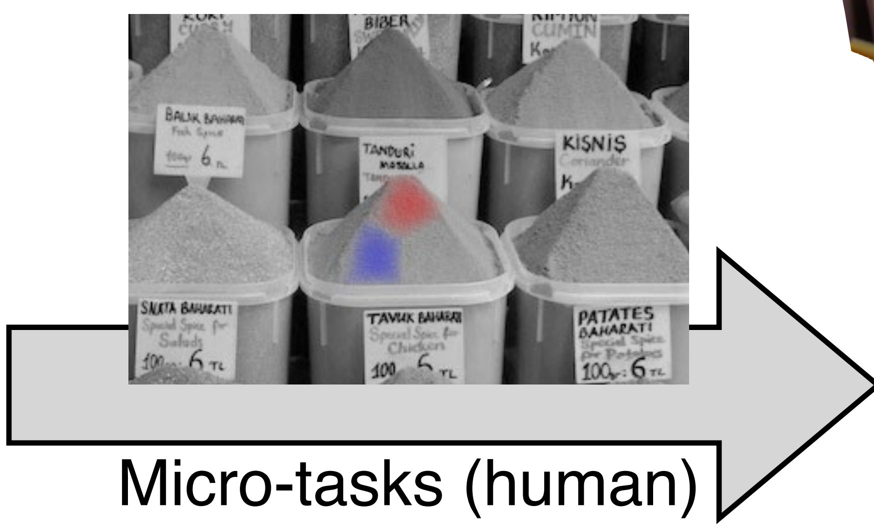
<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm



Patch
Segmentation



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

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm

Quality Control Setup

Micro-tasks (human)

Patch Segmentation

Laplace equation

$$\Delta f = 0$$

with constraints

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

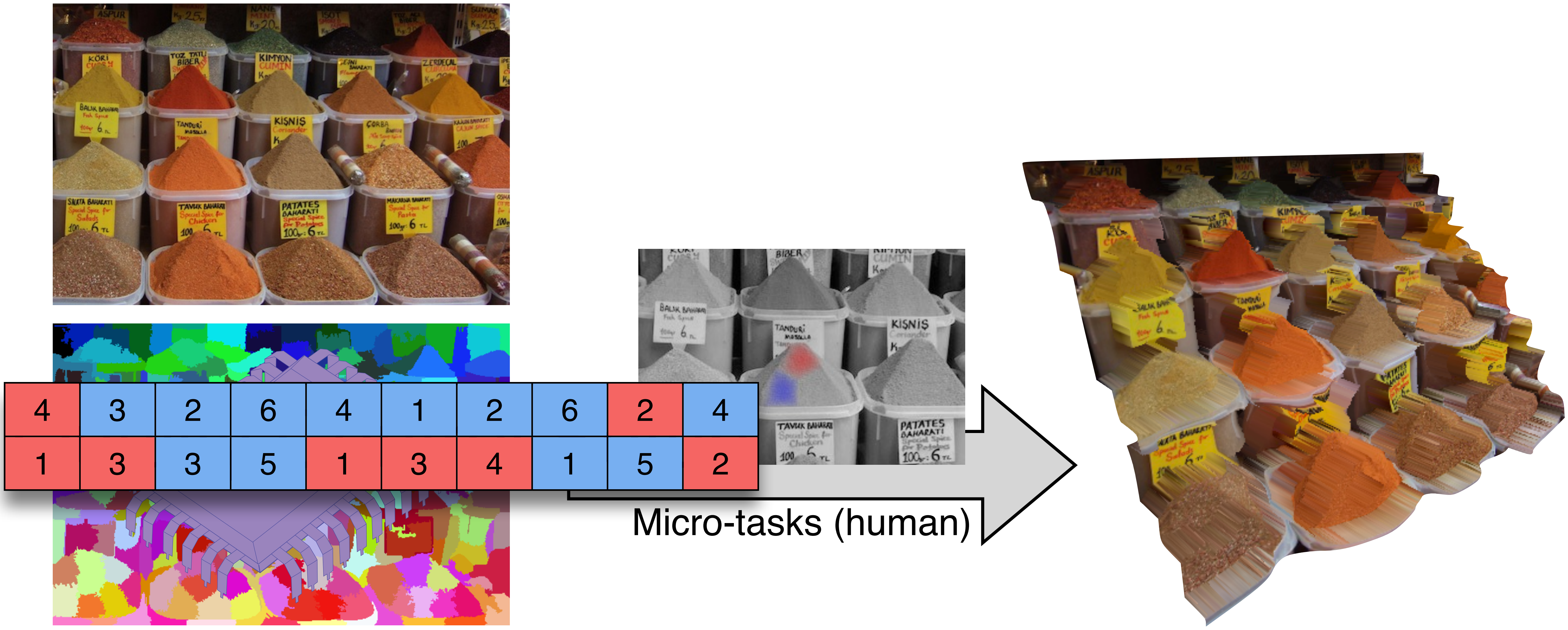
Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks, and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm



Patch Segmentation

Laplace equation

$$\Delta f = 0$$

with constraints

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

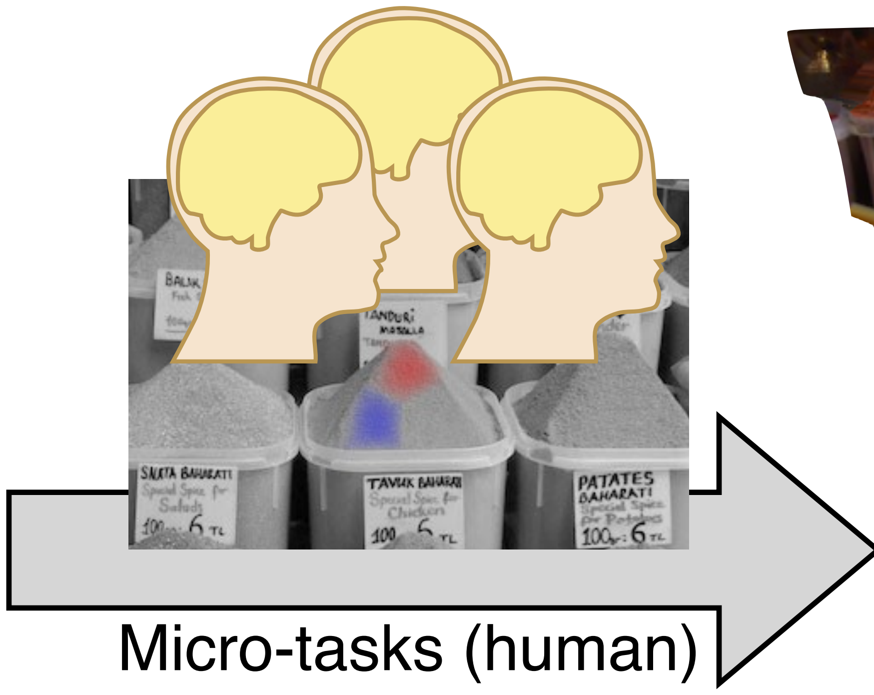
The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm

Patch
Segmentation



Laplace equation

$$\Delta f = 0$$

with constraints

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

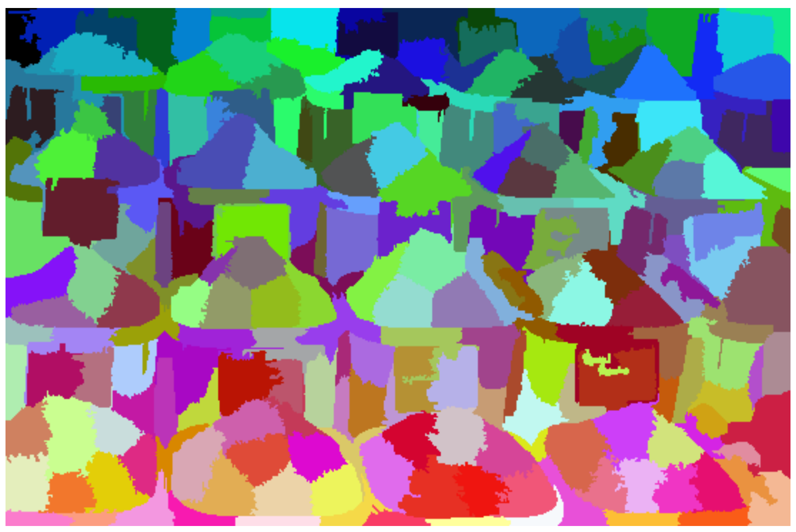
Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

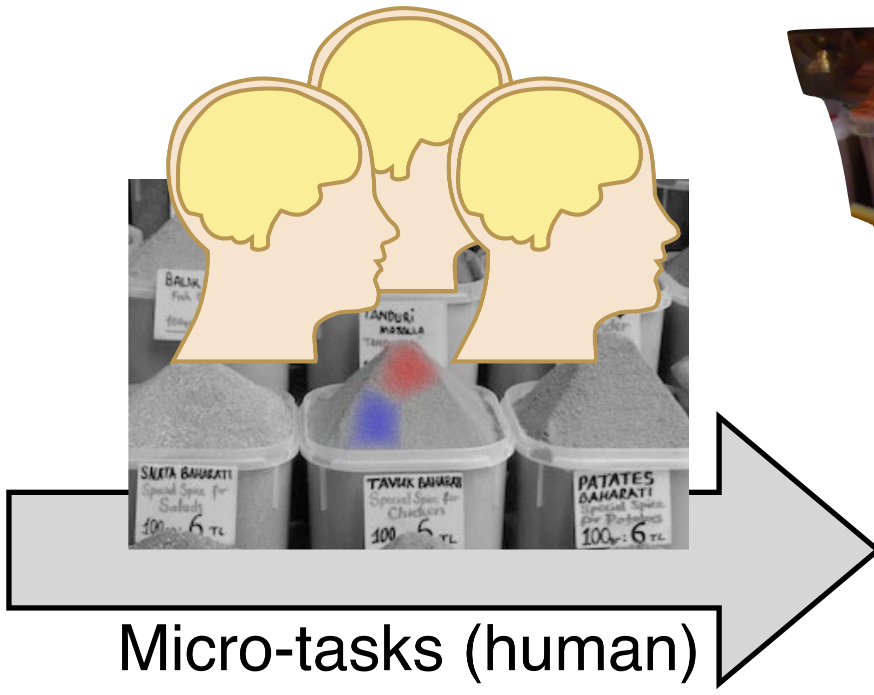
<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm



Patch
Segmentation



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

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

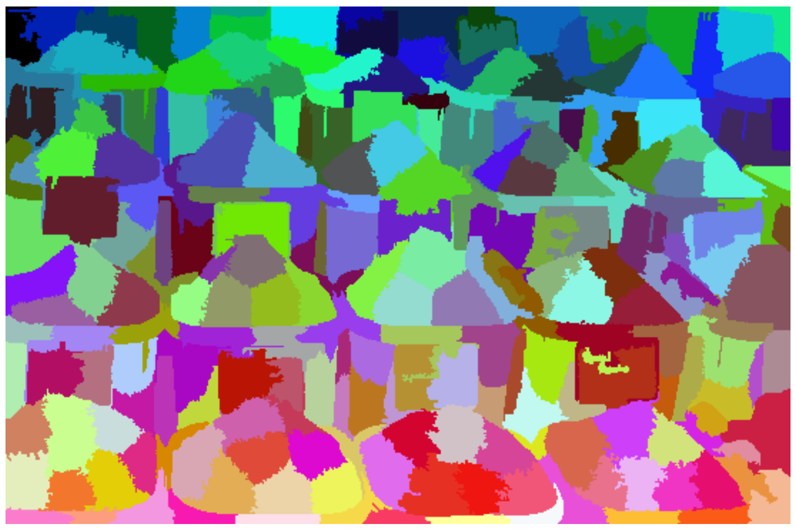
Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

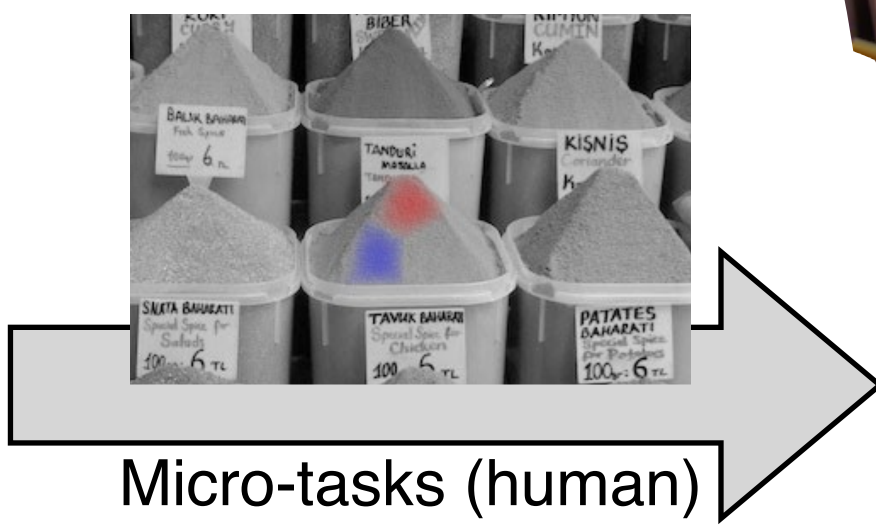
<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

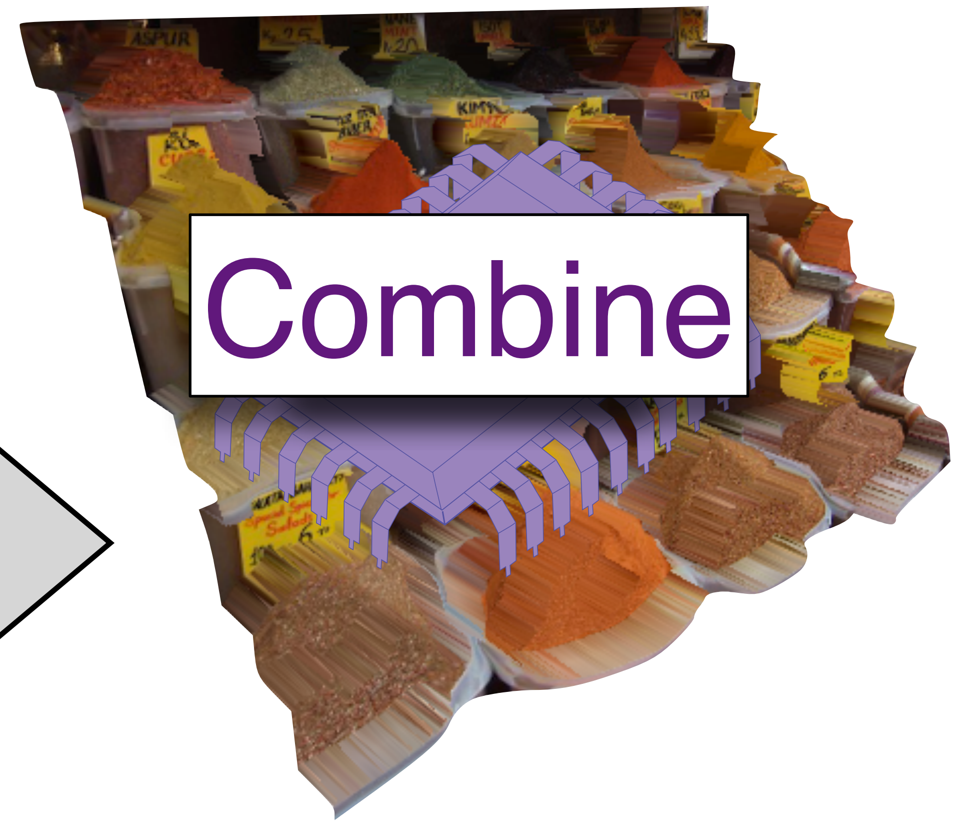
<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm



Patch
Segmentation





Laplace equation

$$\Delta f = 0$$

with constraints

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

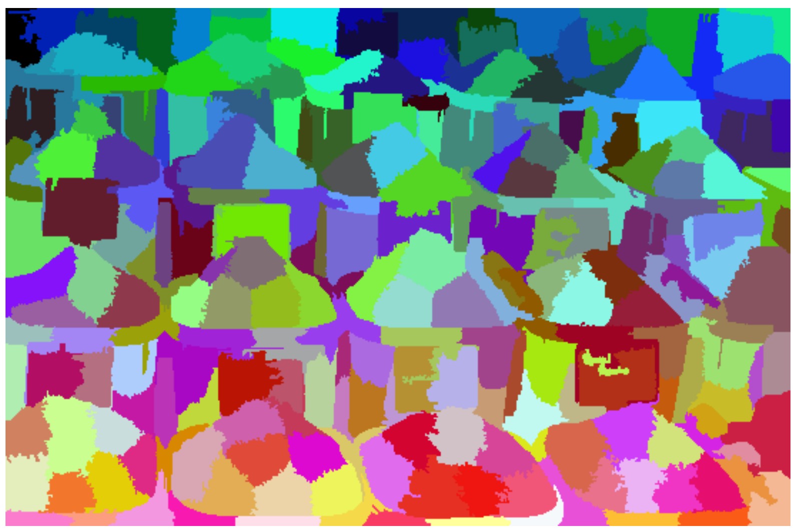
Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

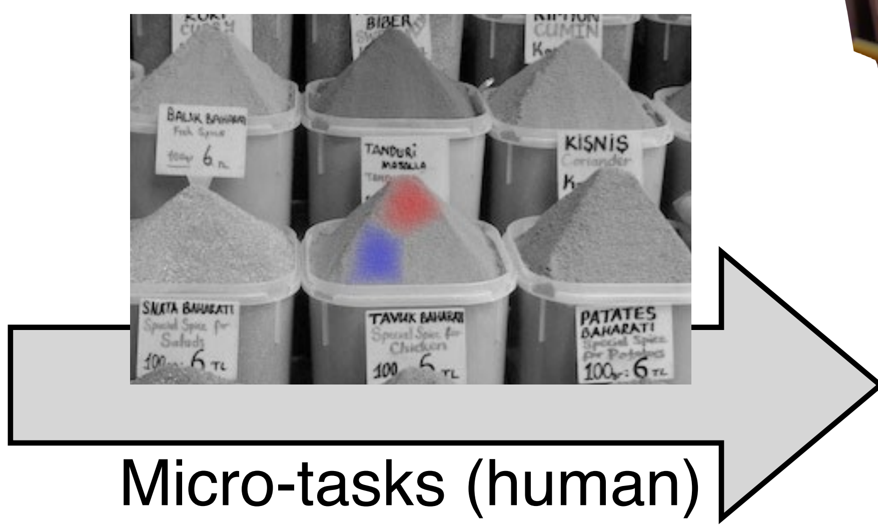
<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm



Patch
Segmentation



Laplace equation

$$\Delta f = 0$$

with constraints

This is an overview of our HC algorithm, and I will walk through the steps.

We first partition, or segment, the input image into superpixel patches.<click>

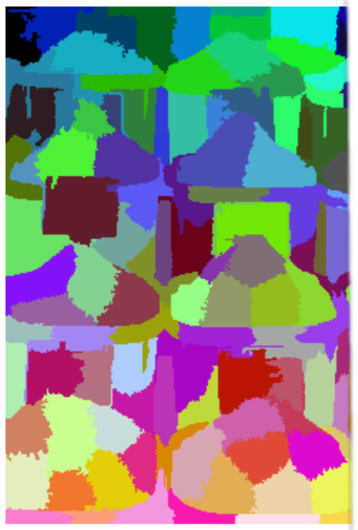
Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.

Algorithm



Pa

Segm

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)) ≥ 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
```

equation

$= 0$

straints

This is an overview of our HC algorithm, and I will walk through the steps.

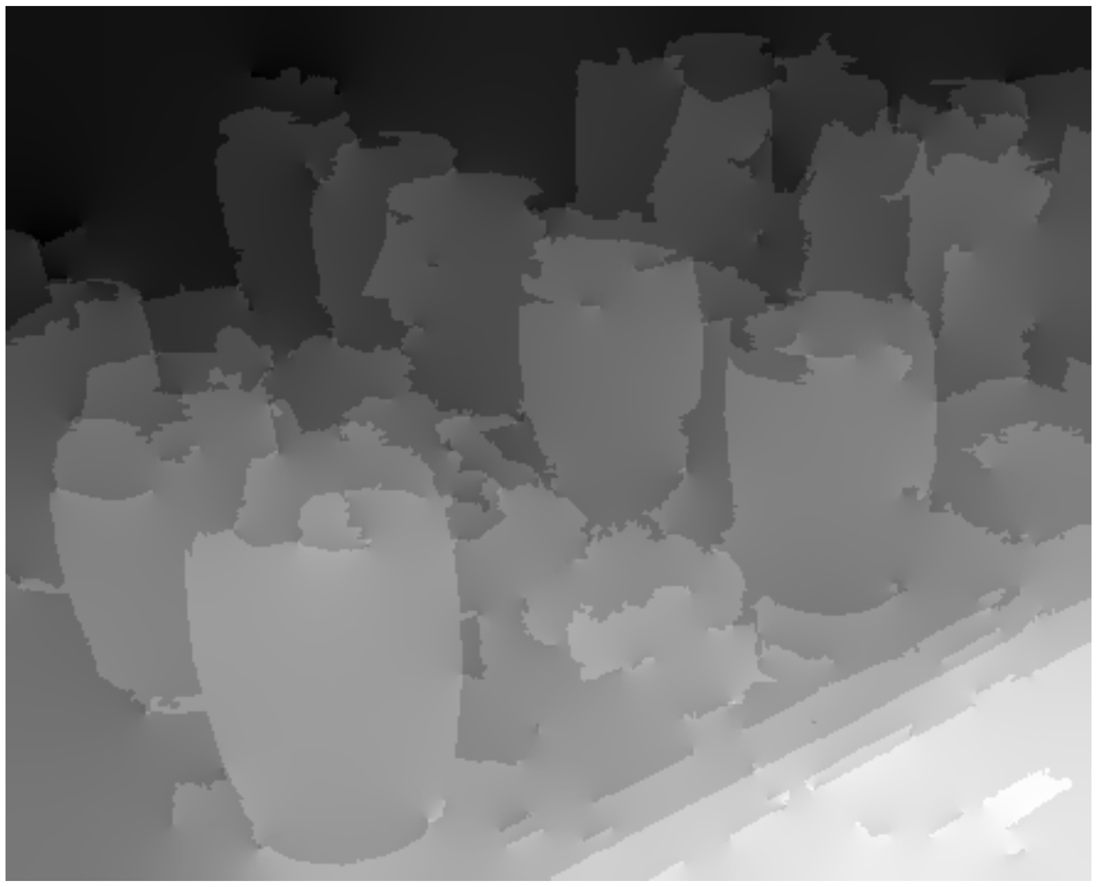
We first partition, or segment, the input image into superpixel patches.<click>

Next we perform our <click> quality control setup, which are our <click> batches of micro-tasks,
and we <click> dispatch each batch to three human processors.

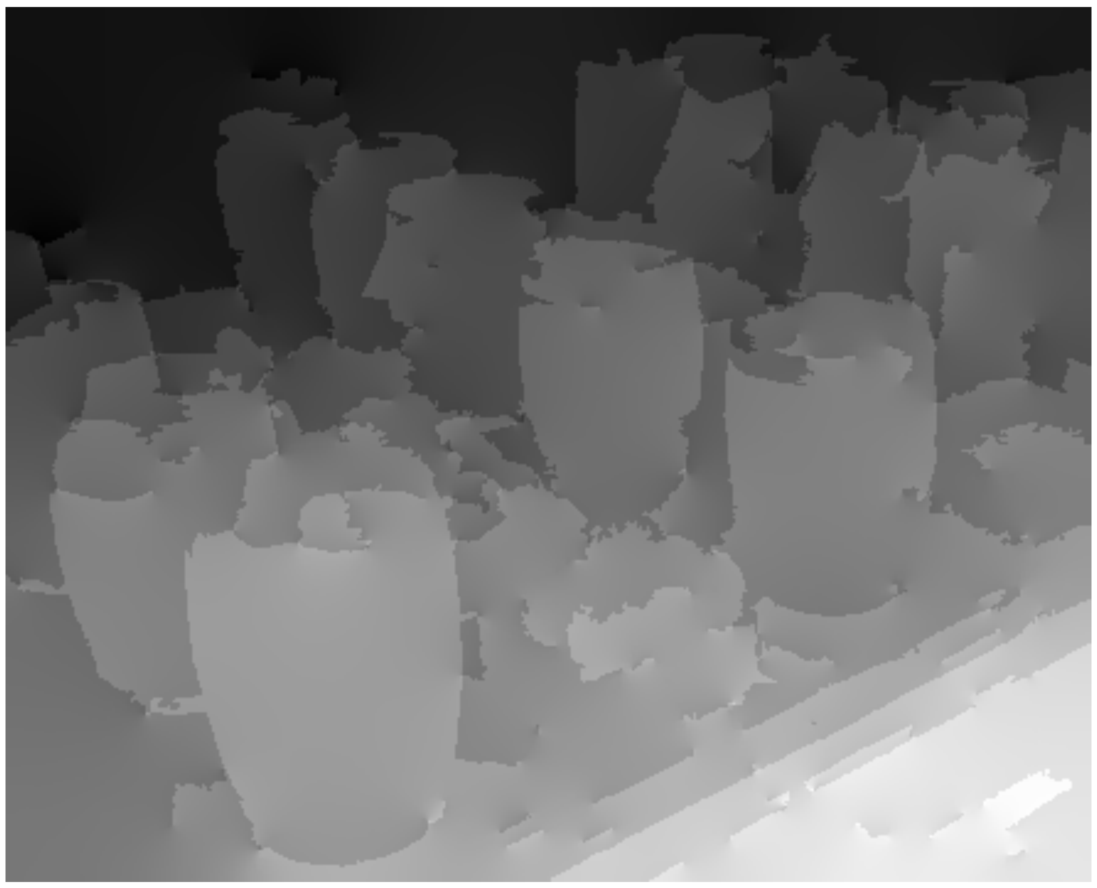
The human processors output a discrete, relative depth ordering (farther, same, or closer) between adjacent regions in the image.
% We take the discrete, relative depth ordering from the (human) micro-tasks and use them as offsets of -1, 0, or 1 between adjacent regions in the image.

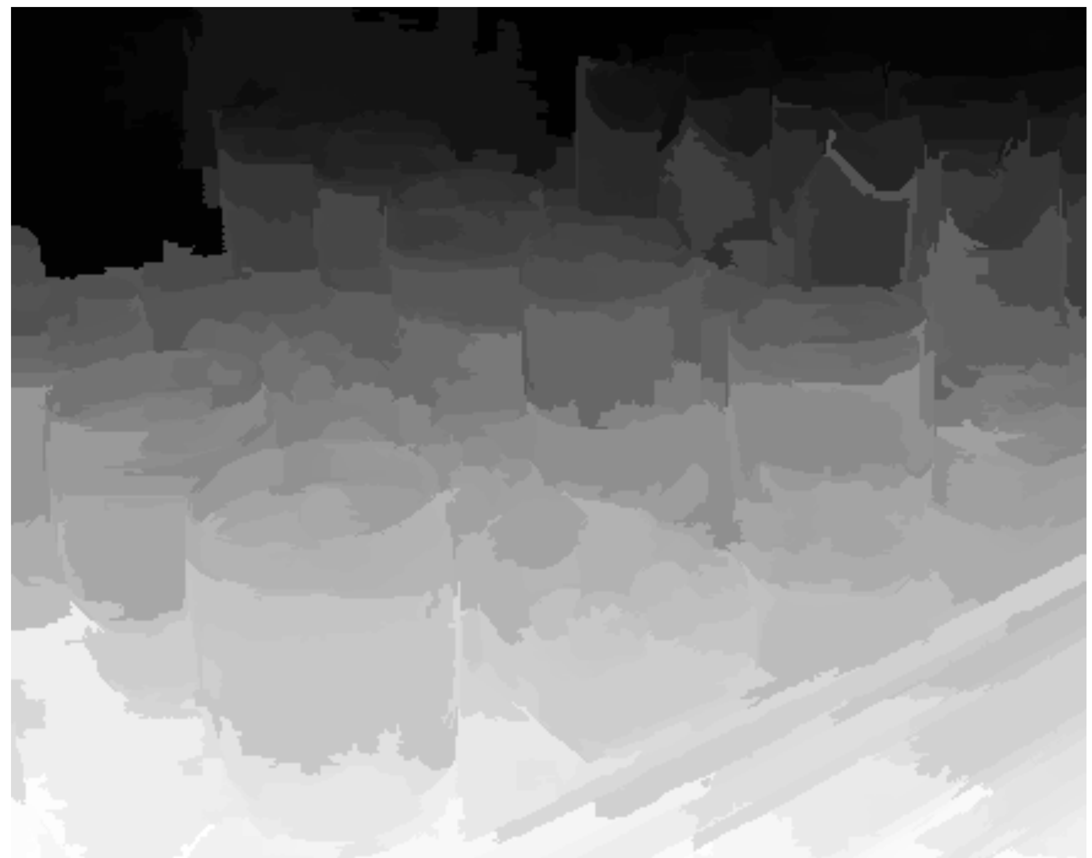
<click>We verify whether the HC passes the quality control tests; if not, we re-dispatch it.
<click>Finally, to construct a depth map, the machine solves a Laplace equation with derivative constraints (of -1, 0, or 1) across region boundaries.

<click>Since we are computer scientists, and we love to design algorithms, this is what it looks like in pseudocode.



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. This simulates having a shallow focus in a photograph.

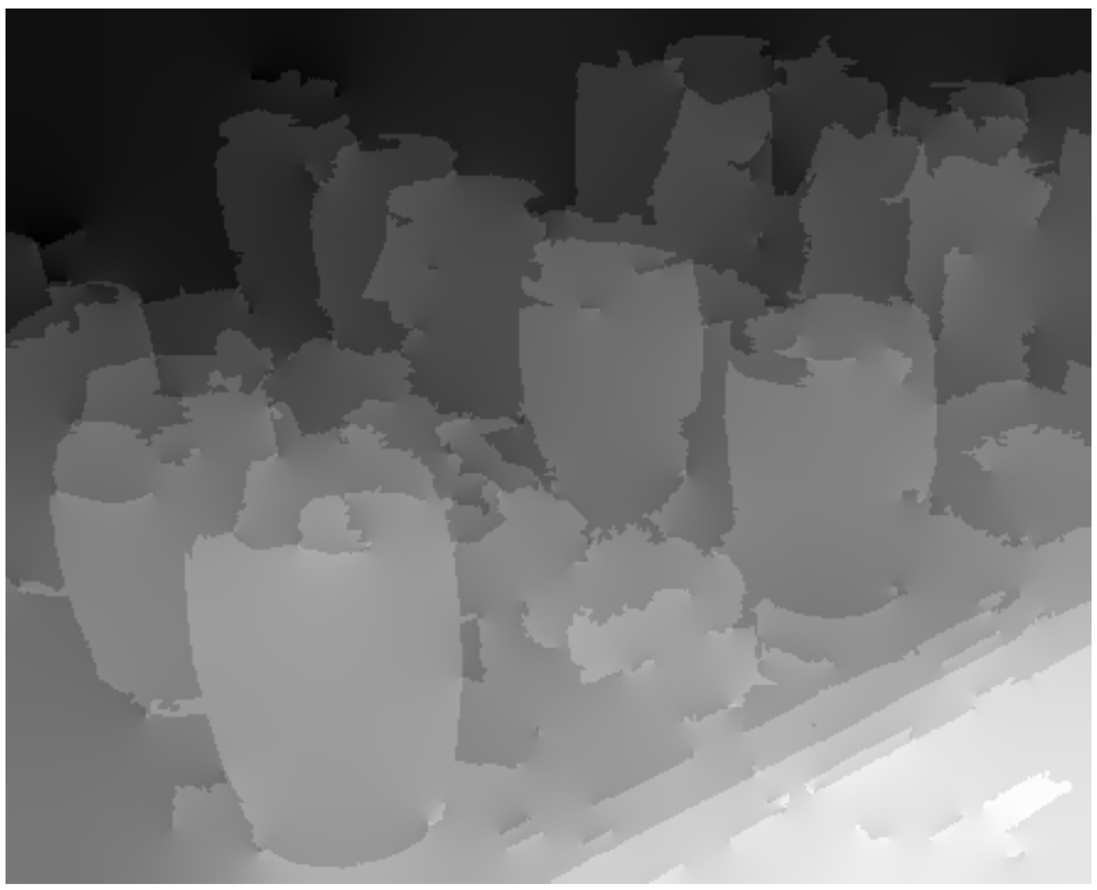




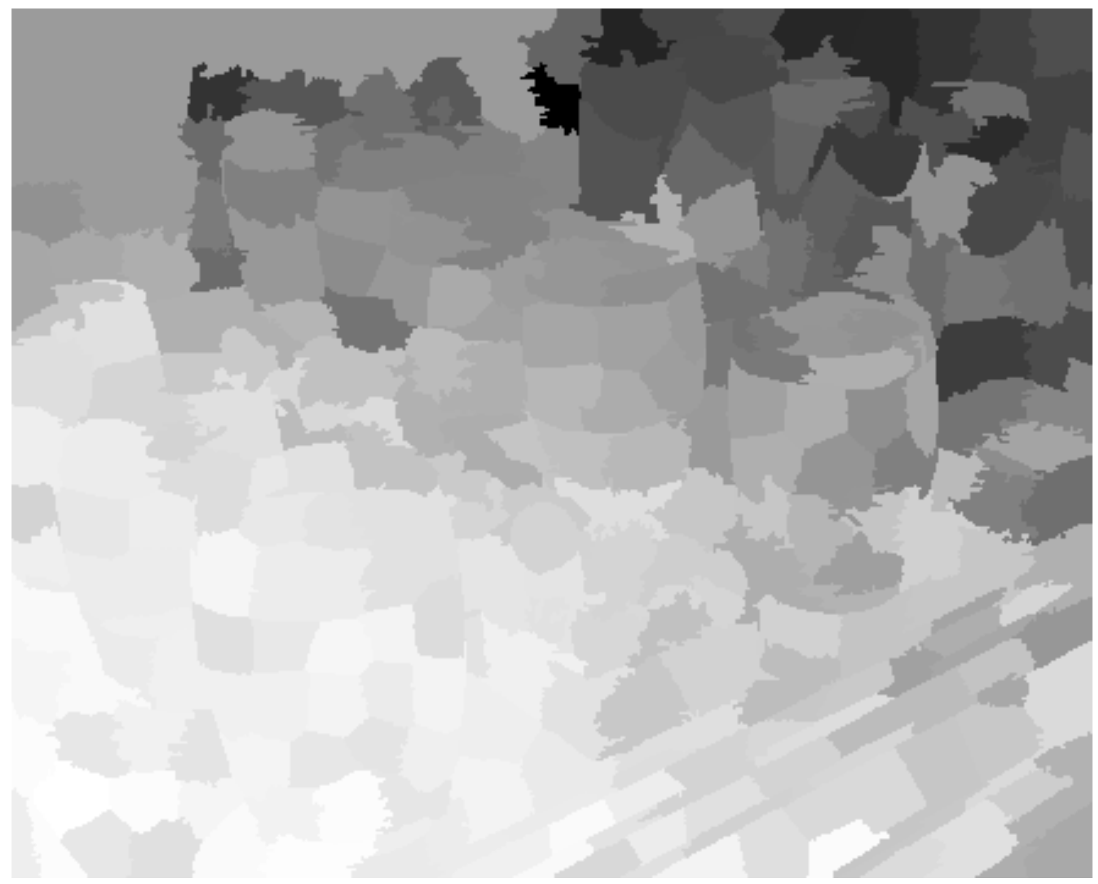
Automatic (Make3D)

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. This simulates having a shallow focus in a photograph.

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. This simulates having a shallow focus in a photograph.

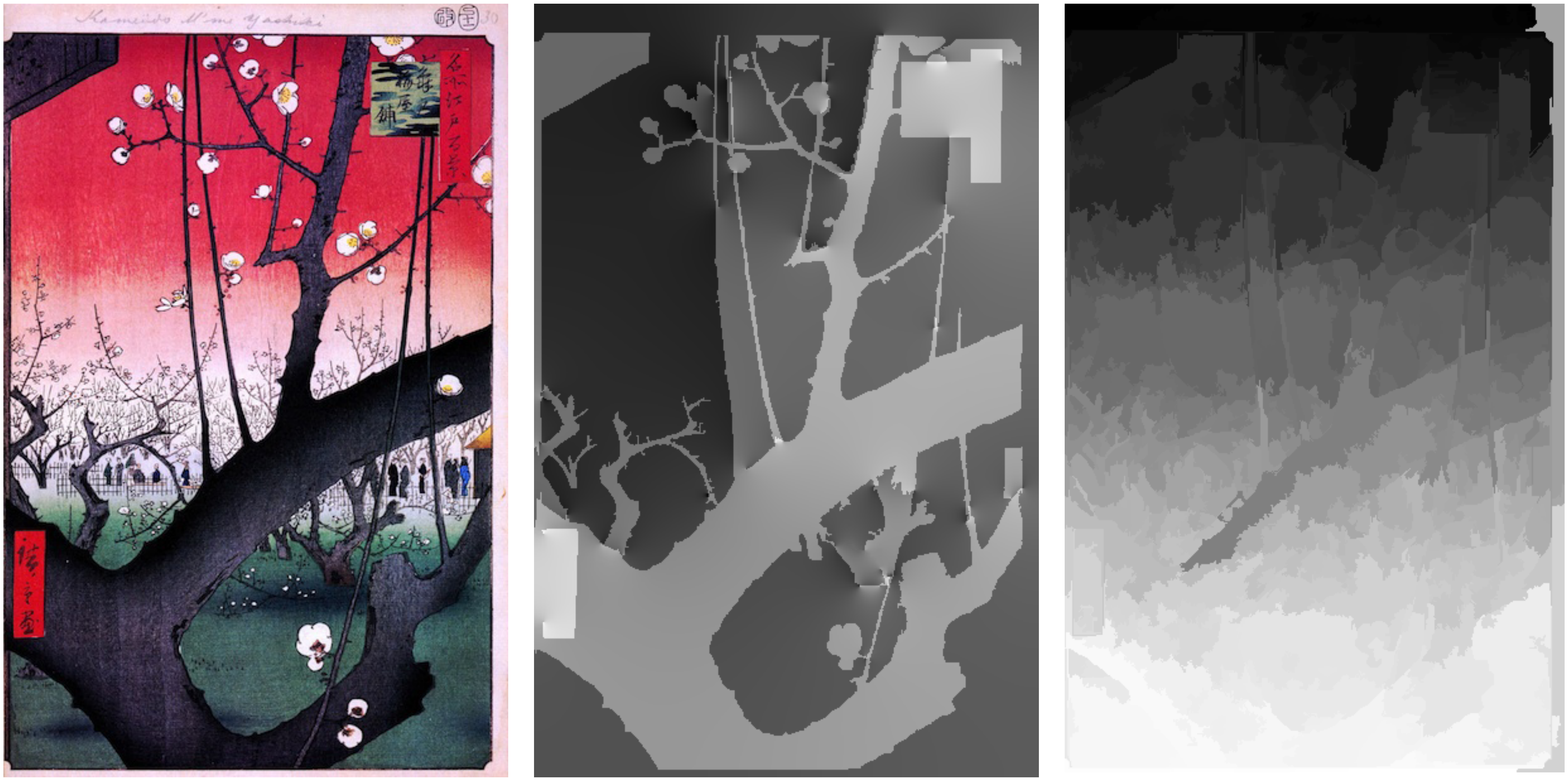


discrete depth



absolute depth

This is the comparison to an “absolute depth” version of the micro-task I promised. Here, humans specified the “absolute” depth of each patch, rather than which of two neighboring patches is closer.
While the result 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.



Automatic (Make3D)

Here are reconstructed depth layers for Hiroshige's ``Kameido Umeyashiki'' woodblock print using our human computation algorithm vs. automatic results produced by Make3D [Saxena et al. 2008].
These artistic inputs create a challenge for automatic algorithms, since they were not meant to handle them.

And here we use the depth map to apply a depth-based blurring effect.

Here are reconstructed depth layers for Hiroshige's ``Kameido Umeyashiki'' woodblock print using our human computation algorithm vs. automatic results produced by Make3D [Saxena et al. 2008].
These artistic inputs create a challenge for automatic algorithms, since they were not meant to handle them.

And here we use the depth map to apply a depth-based blurring effect.

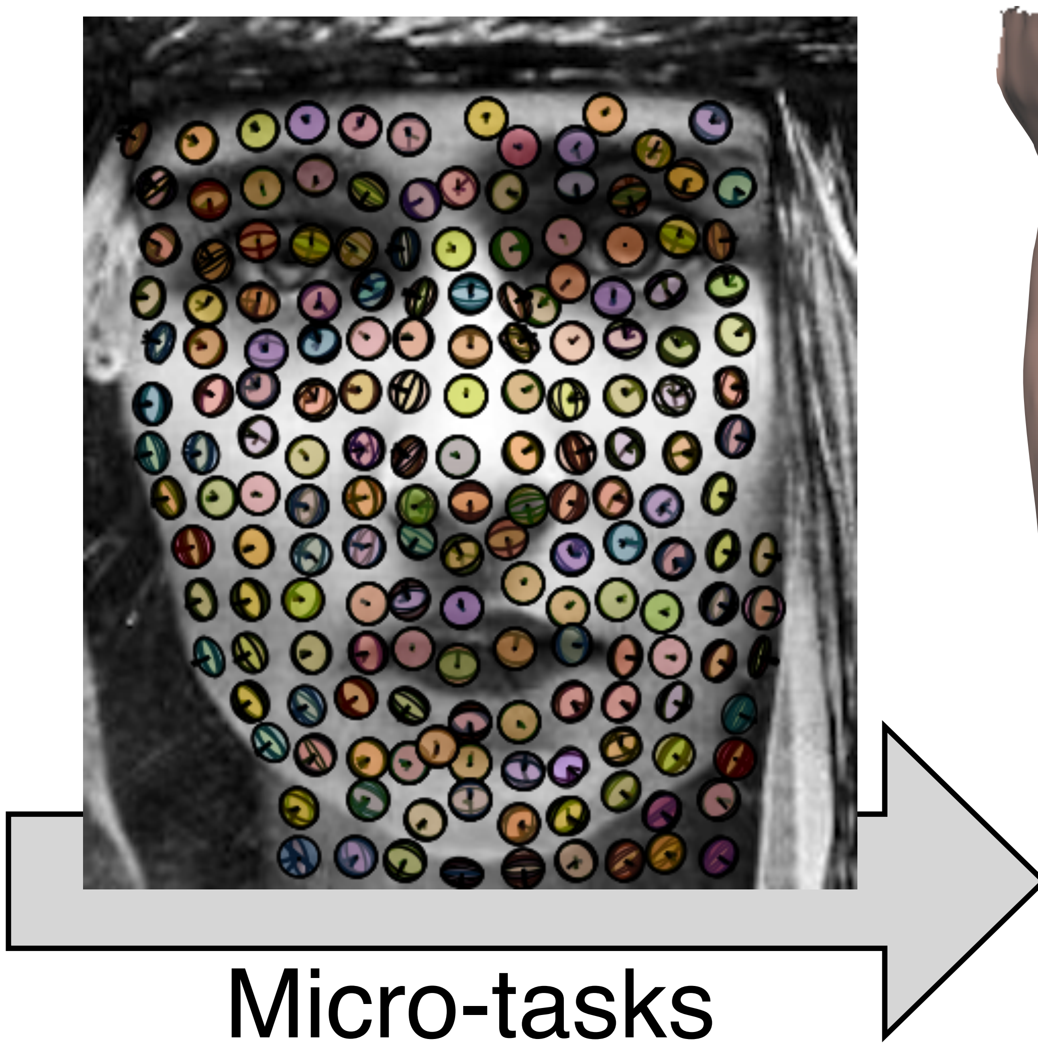
Algorithm 2: Normal Map

In our second algorithm, we create a normal map for a given image. This can be used for relighting or for surface reconstruction.
% And in fact, this is a human-based shape-from-shading algorithm.

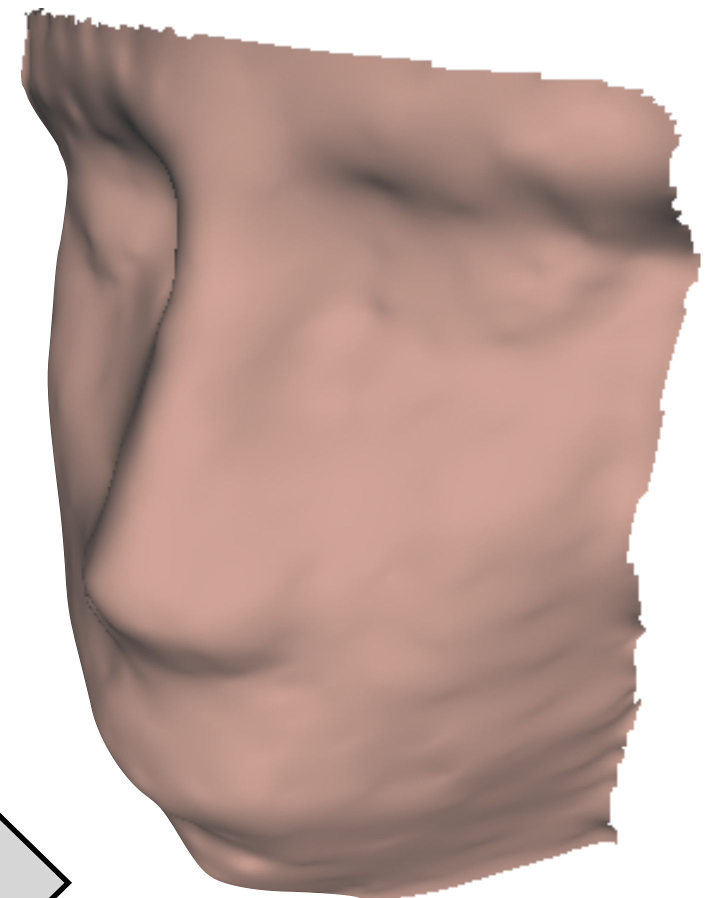
We use a **"gauge figure" micro-task** that comes from the perception literature [Koenderink et al. 1992]; it was also used by [Cole et al. 2009] for gathering normals using the Mechanical Turk.
I should mention that **[Cole et al. 2009] was the inspiration** for this research.
HPs **orient** the gauge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

Algorithm

Select normal locations



Micro-tasks (human)



Bi-Laplace equation

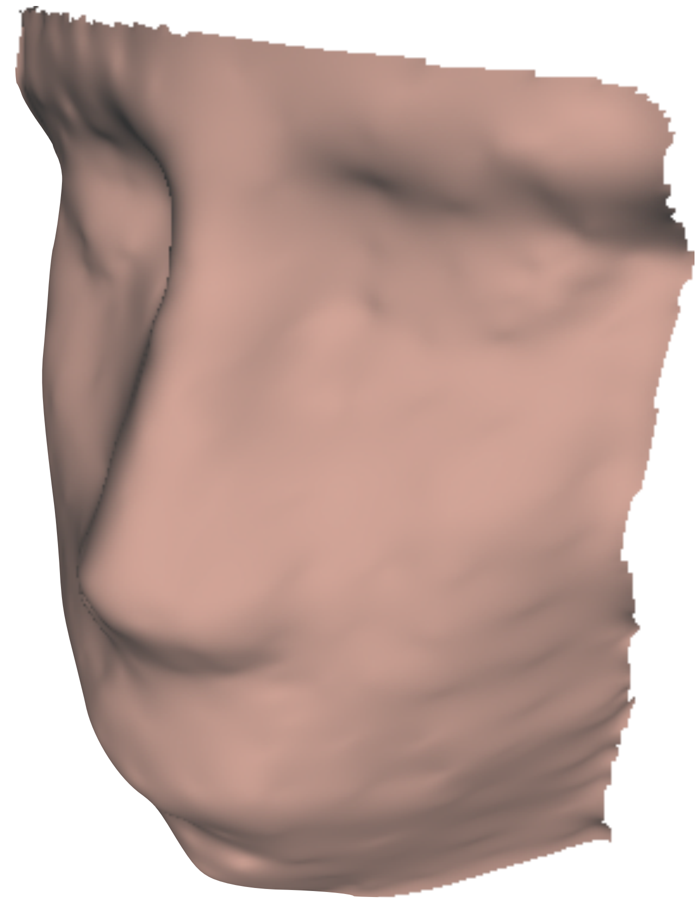
$$\Delta^2 f = 0$$

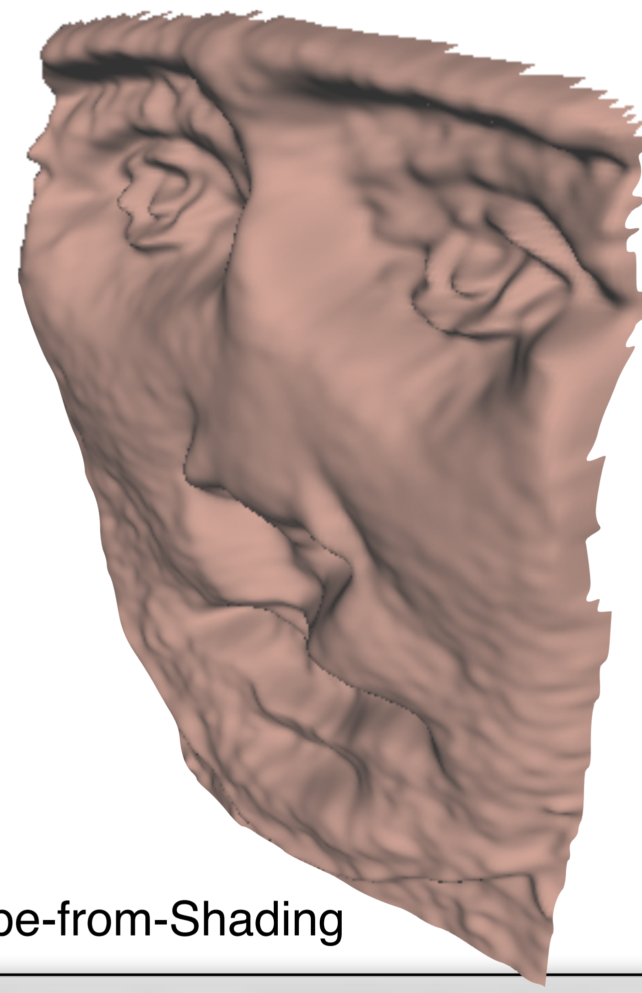
with constraints

Our normal map algorithm has a **similar quality control setup** to our depth layers algorithm. There are two differences. First, we need **thresholds** for comparing normals. Second, **human depth perception differs by a z-scaling factor, so we must account for humans differing internal biases**.
Since our micro-task batches include queries with known answers, we can **align or rectify the depth scaling** of each HP's normals with the scaling of the **known answers**.
We implemented this algorithm in an adaptive manner, so large variations in normals leads to more 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.
<click>
Here is a comparison of our human SfS to the best output from among the Shape-from-Shading approaches in a recent survey.
% "Tsai and Shah" from [Durou et al. 2008] survey.

<click>
Here we apply the normal map to add two new lights to this old photograph of a face.





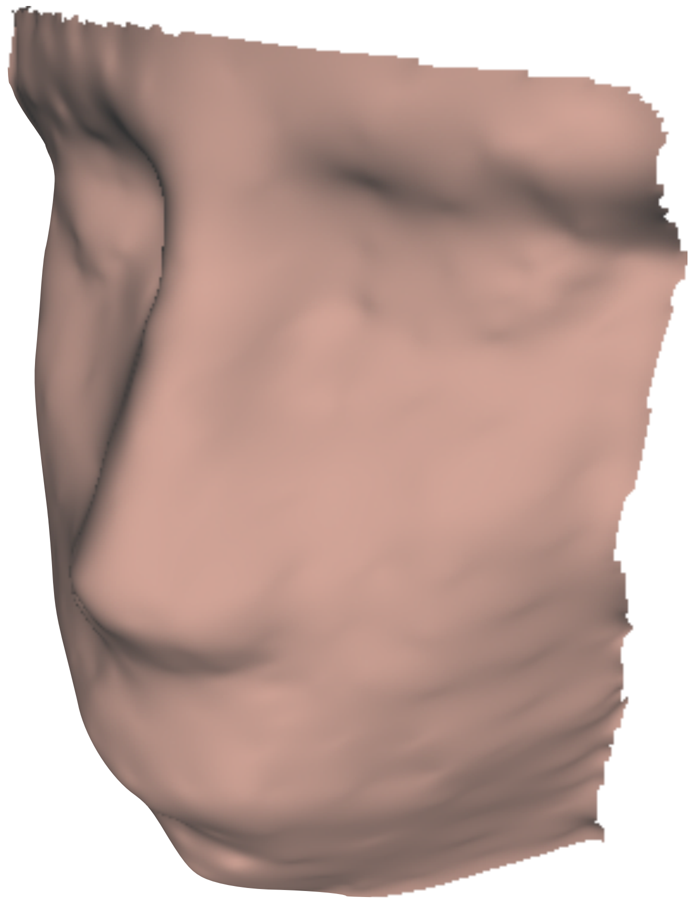
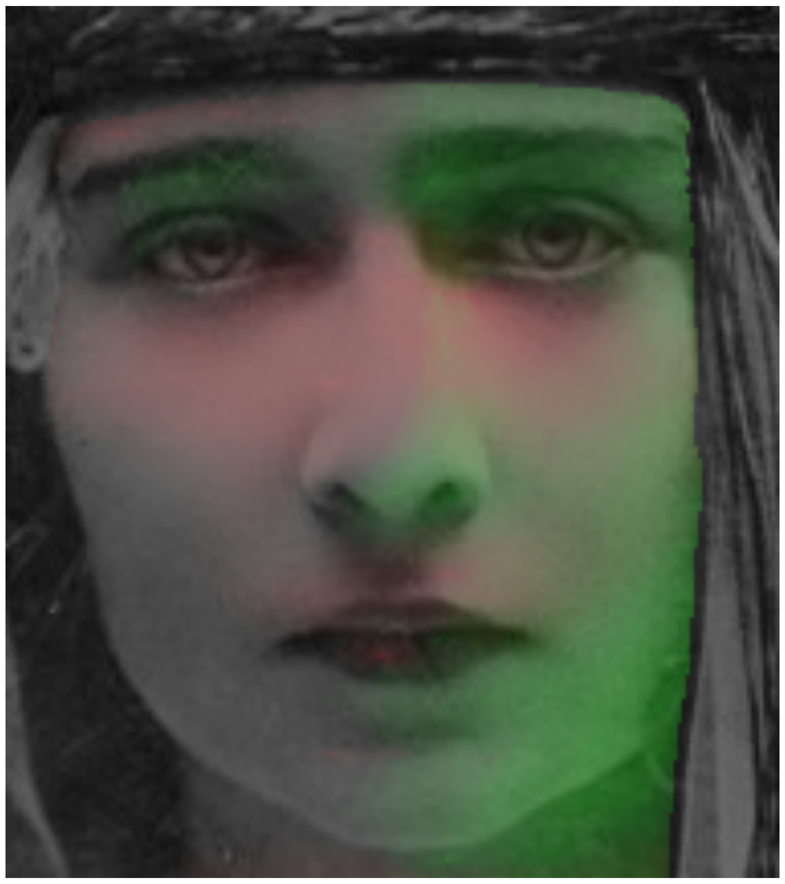
Shape-from-Shading

Our normal map algorithm has a **similar quality control setup** to our depth layers algorithm. There are two differences.
First, we need **thresholds** for comparing normals. Second, **human depth perception differs by a z-scaling factor, so we must account for humans differing internal biases**.
Since our micro-task batches include queries with known answers, we can **align or rectify the depth scaling** of each HP's normals with the scaling of the **known answers**.
We implemented this algorithm in an adaptive manner, so large variations in normals leads to more 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.
<click>
Here is a comparison of our human SfS to the best output from among the Shape-from-Shading approaches in a recent survey.
% "Tsai and Shah" from [Durou et al. 2008] survey.

<click>
Here we apply the normal map to add two new lights to this old photograph of a face.



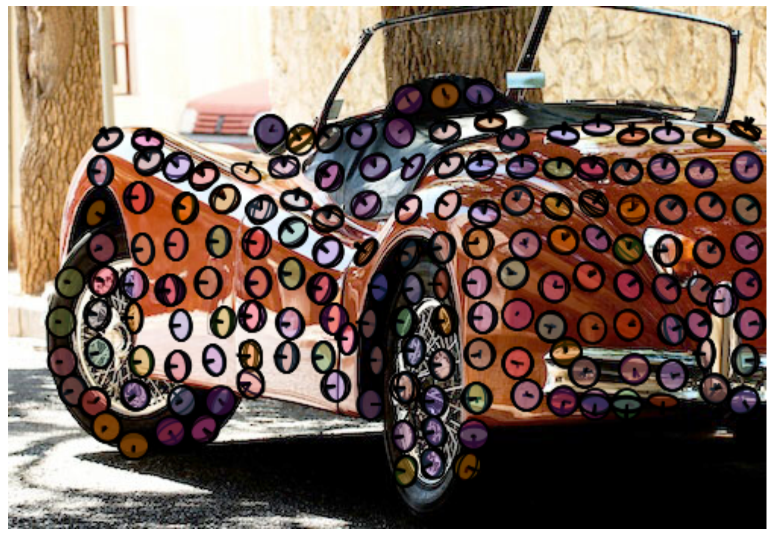



Our normal map algorithm has a **similar quality control setup** to our depth layers algorithm. There are two differences. First, we need **thresholds** for comparing normals. Second, **human depth perception differs by a z-scaling factor, so we must account for humans differing internal biases**.
Since our micro-task batches include queries with known answers, we can **align or rectify the depth scaling** of each HP's normals with the scaling of the **known answers**.
We implemented this algorithm in an adaptive manner, so large variations in normals leads to more queries.

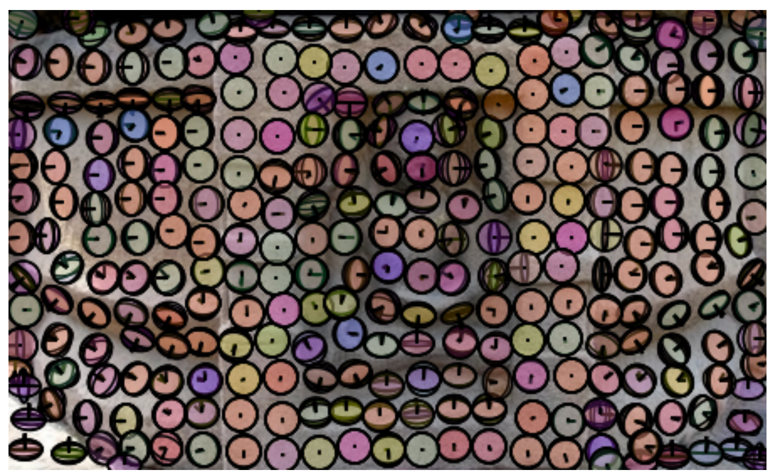
For **composition**, we solve a bilaplace equation (bilaplacian = 0) with the user-given normals as least-squares constraints. This removes noise and ensures the consistency of the normals.
<click>
Here is a comparison of our human SfS to the best output from among the Shape-from-Shading approaches in a recent survey.
% "Tsai and Shah" from [Durou et al. 2008] survey.

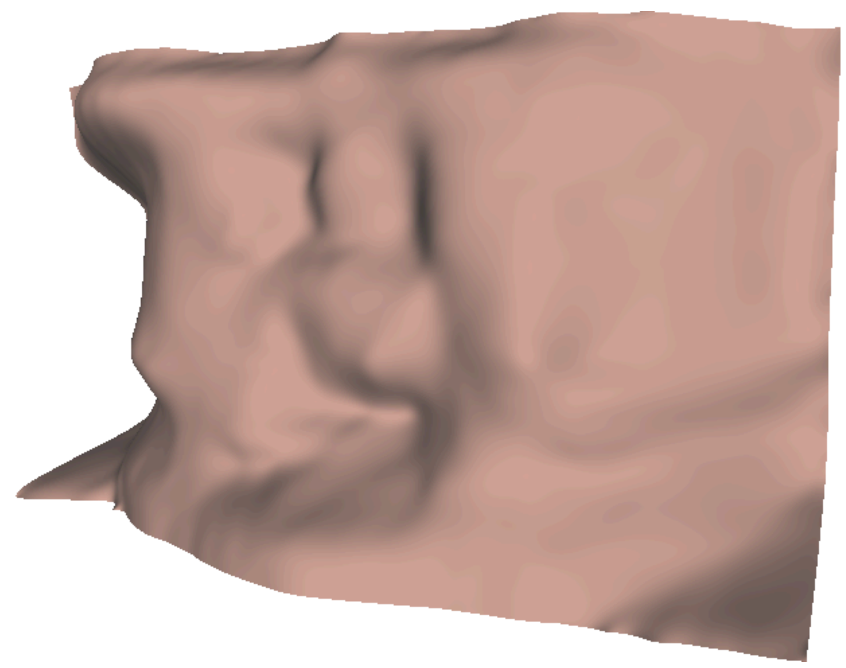
<click>
Here we apply the normal map to add two new lights to this old photograph of a face.

[Pedro Ribeiro Simões]



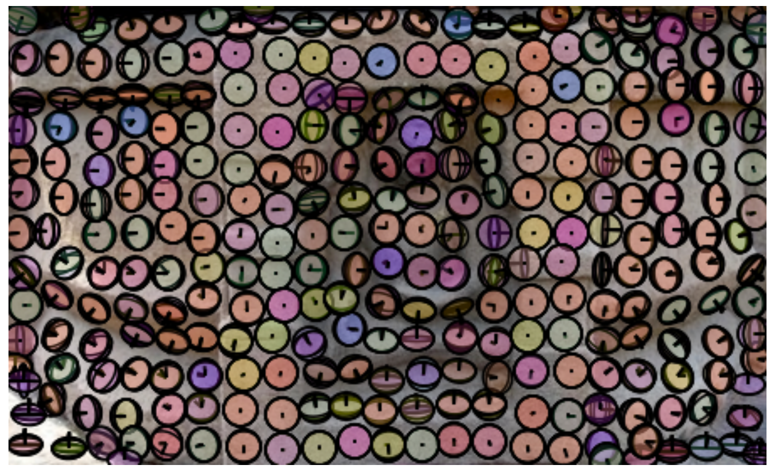
[Warren Apel]

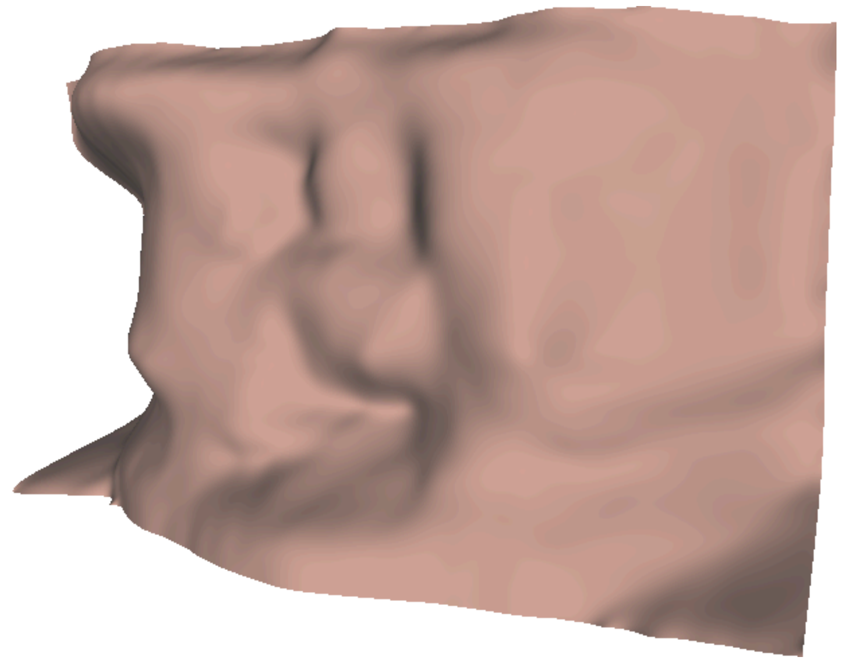


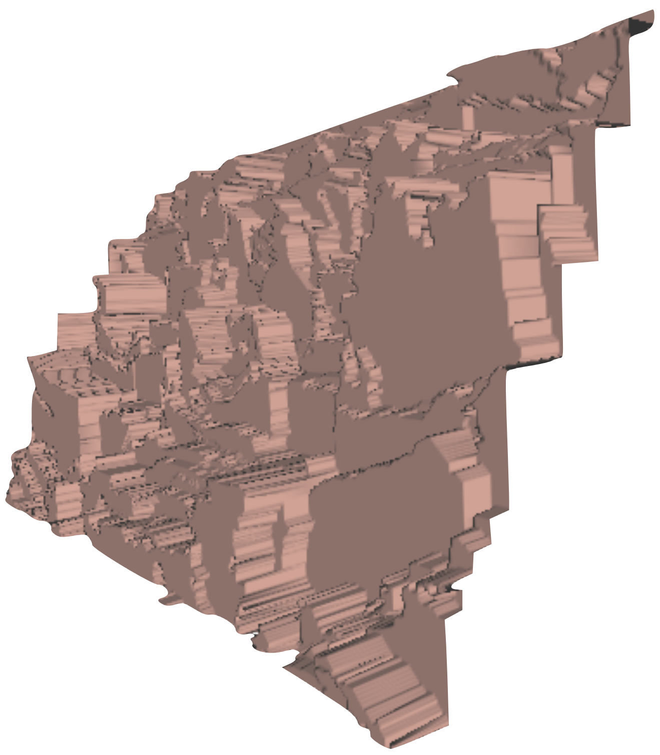


Here are two more examples of 3D reconstructions.
<click>
And here is a comparison to purely automatic approaches: Make3D (left) and two shape-from-shading approaches: (middle) Tsai and Shah and (right) Falcone and Sagona.

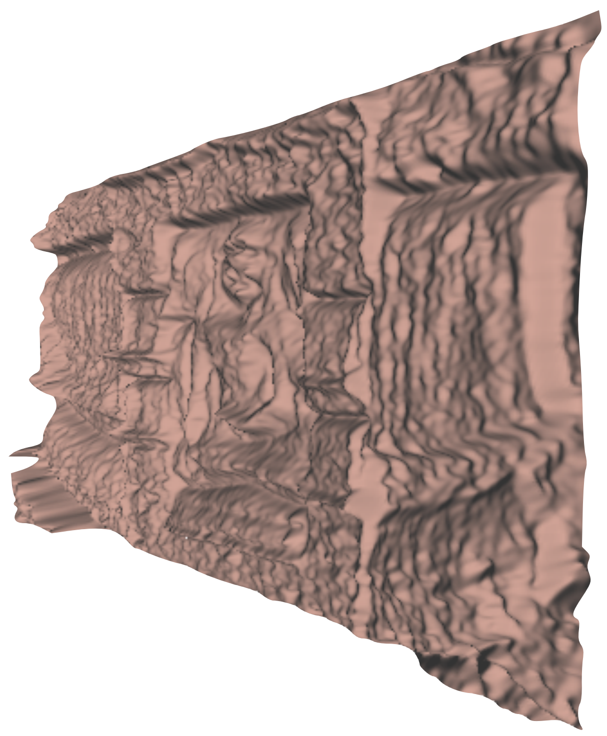
[Warren Apel]

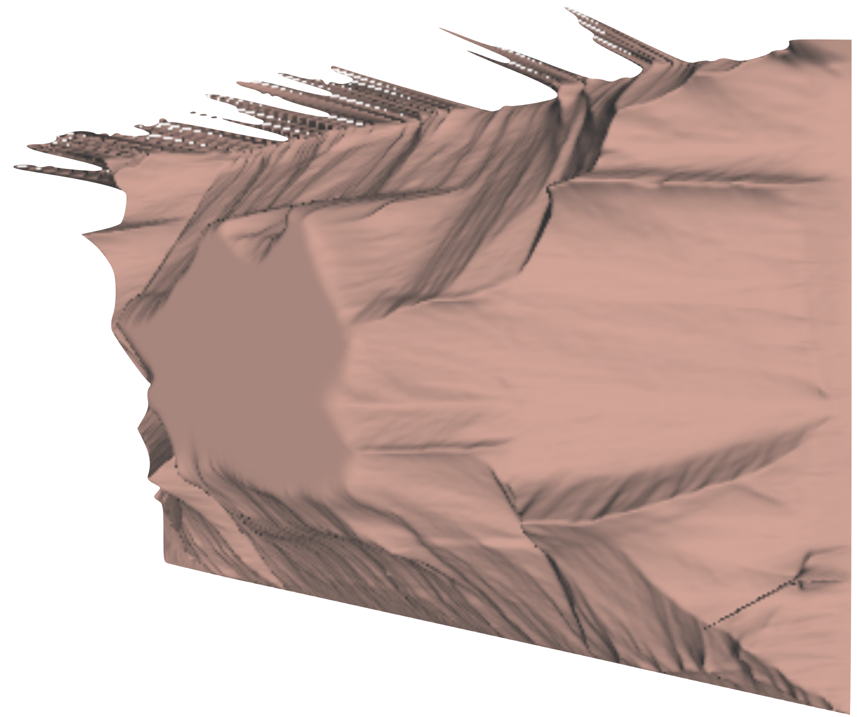






Make3D

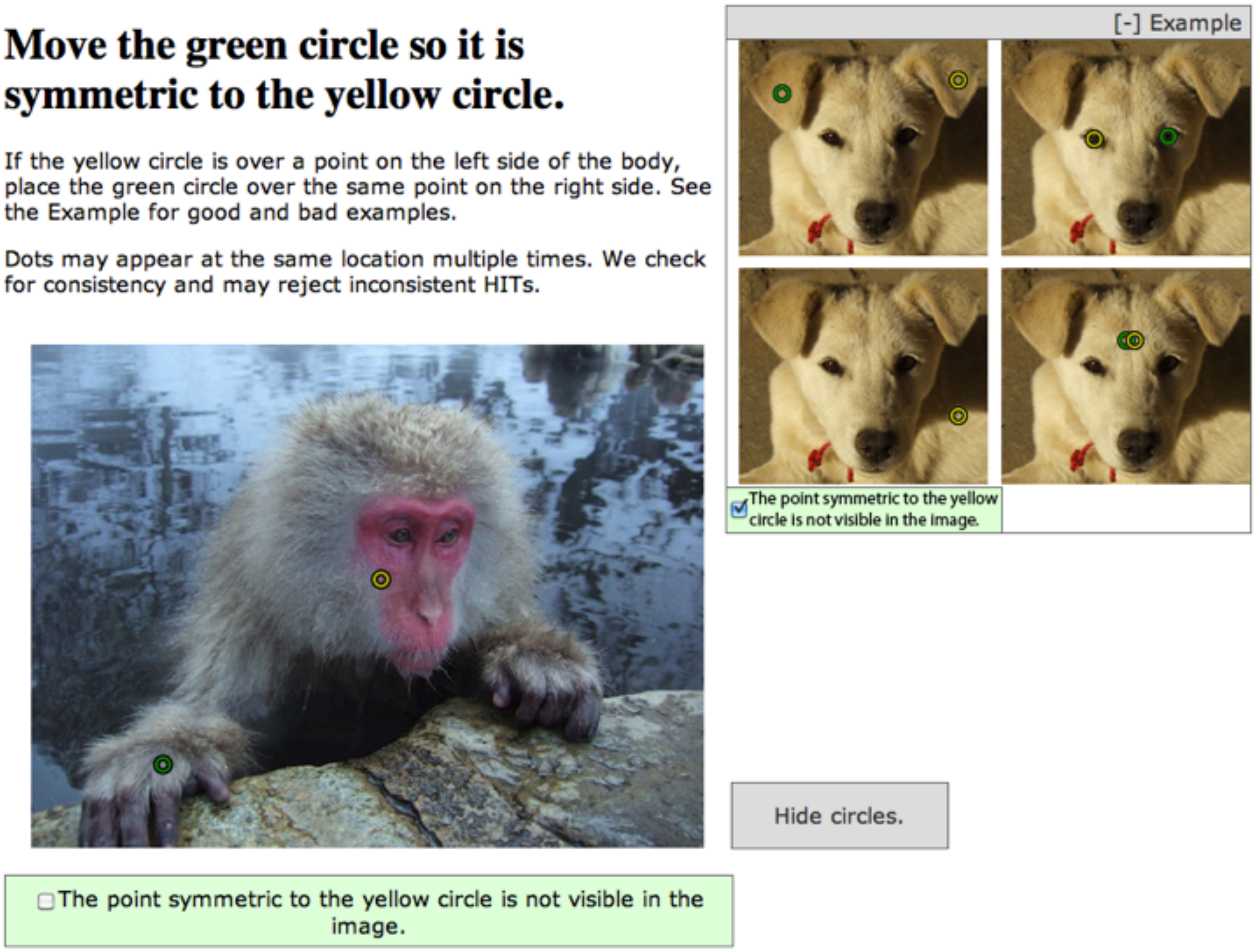




Shape-from-Shading

And here is a comparison to purely automatic approaches: Make3D (left) and two shape-from-shading approaches: (middle) Tsai and Shah and (right) Falcone and Sagona.

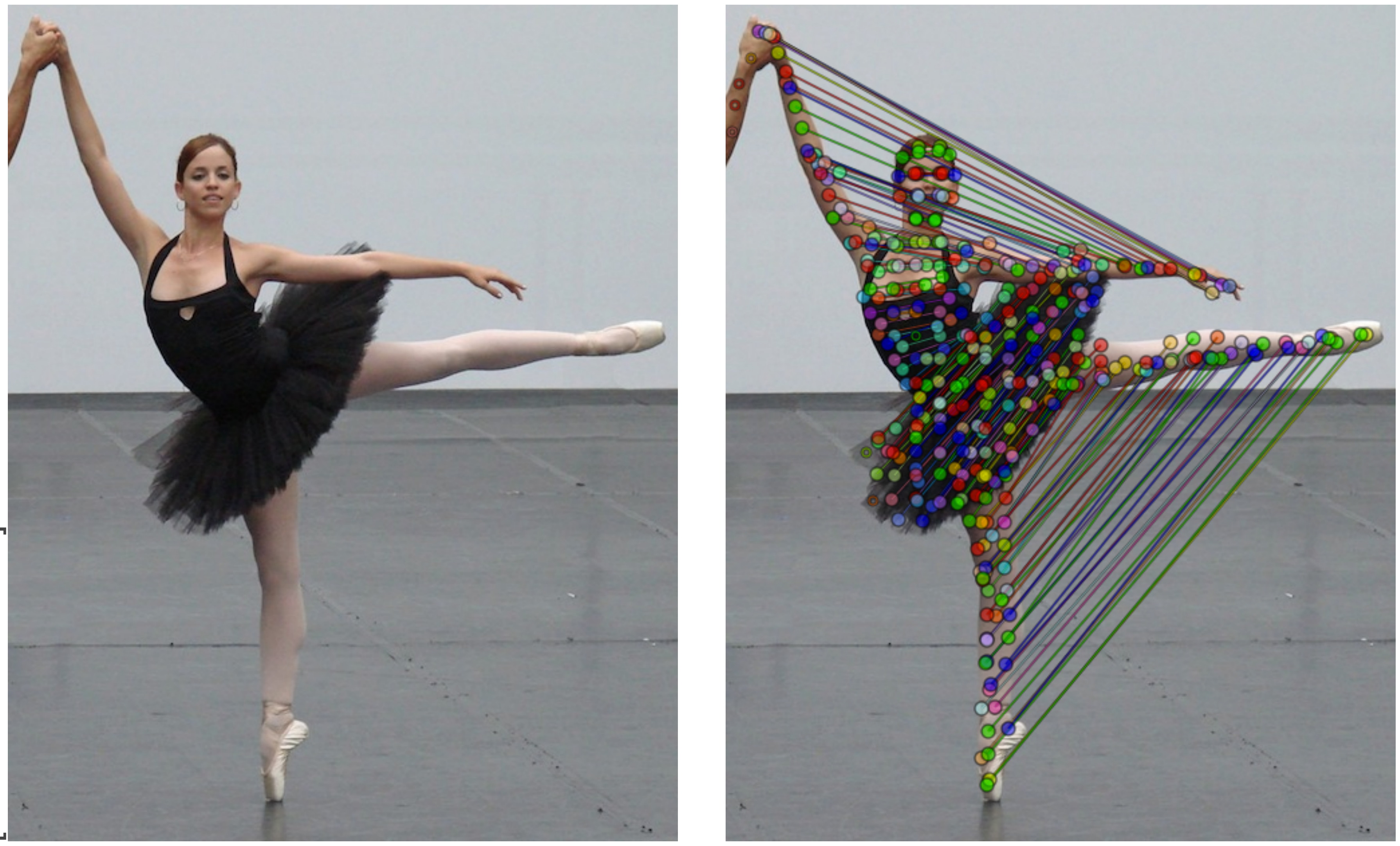
Algorithm 3: Bilateral Symmetry Map



In our third algorithm, we create a bilateral symmetry map for an object in an image. This can be used for propagating edits in photoshop from one side of the object to the other, or for minimizing perceptual distortions when retargeting an image.

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.

Our algorithm is able to compute a high-quality bilateral symmetry map even when the objects are highly distorted in image-space, like the dancer.

% For the car, our algorithm correctly finds the reflective symmetry and is not confused by the one visible door.

Statistics

example	micro-tasks used	total $ cost	successful micro-task duration: avg	successful micro-task duration: median	algorithm delay until % complete: 50%	algorithm delay until % complete: 100%
normal map	1620–4340	$5.04–10.76	8.8 s	8.1 s	1.1–5.0 hrs	2.8–15.1 hrs
depth layers	2669–7620	$6.41–17.15	6.2 s	5.5 s	0.95–1.6 hrs	3.7–8.0 hrs
symmetry map	1020–1740	$3.24–3.92	9.0 s	8.5 s	0.4–1.6 hrs	0.7–4.9 hrs

- We use **a lot** of micro-tasks, which is why we emphasize **massive parallelism**.
- The **cost** is low enough to be **practical**, $3-$11 or $17. The depth layers algorithm is the most expensive, because there is a micro-task per neighboring patches, not per patch.

<click>

- Micro-tasks take **little time**! 5–9 seconds, and we believe that it is mostly interface overhead, like mouse movement.
- This means that our algorithms could complete in **~3 minutes** if we had enough HPs running in parallel.

<click>

- The main problem is the total time, which is long. The delay is entirely due to waiting for HPs to choose to perform the tasks.
    - This is a “**market problem**” in the Amazon Mechanical Turk.
    - % 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.
    - Would be helped if this were a **popular algorithm** (if we were generating a constant stream of micro-tasks).
    - It’s **anyone’s guess** how the market of HC will change in the future; there are new online labor markets that don’t have these properties.

Statistics

example	micro-tasks used	total \$ cost	successful micro-task duration avg	successful micro-task duration median	algorithm delay until % complete 50%	algorithm delay until % complete 100%
normal map	1620–4340	\$5.04–10.76	8.8 s	8.1 s	1.1–5.0 hrs	2.8–15.1 hrs
depth layers	2669–7620	\$6.41–17.15	6.2 s	5.5 s	0.95–1.6 hrs	3.7–8.0 hrs
symmetry map	1020–1740	\$3.24–3.92	9.0 s	8.5 s	0.4–1.6 hrs	0.7–4.9 hrs

- We use **a lot** of micro-tasks, which is why we emphasize **massive parallelism**.
- The **cost** is low enough to be **practical**, \$3-\$11 or \$17. The depth layers algorithm is the most expensive, because there is a micro-task per neighboring patches, not per patch.

<click>

- Micro-tasks take **little time**! 5–9 seconds, and we believe that it is mostly interface overhead, like mouse movement.
- This means that our algorithms could complete in **~3 minutes** if we had enough HPs running in parallel.

<click>

- The main problem is the total time, which is long. The delay is entirely due to waiting for HPs to choose to perform the tasks.
    - This is a “**market problem**” in the Amazon Mechanical Turk.
    - % 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.
    - Would be helped if this were a **popular algorithm** (if we were generating a constant stream of micro-tasks).
    - It’s **anyone’s guess** how the market of HC will change in the future; there are new online labor markets that don’t have these properties.

Statistics

example	micro-tasks used	total $ cost	successful micro-task duration: avg	successful micro-task duration: median	algorithm delay until % complete: 50%	algorithm delay until % complete: 100%
normal map	1620–4340	$5.04–10.76	8.8 s	8.1 s	1.1–5.0 hrs	2.8–15.1 hrs
depth layers	2669–7620	$6.41–17.15	6.2 s	5.5 s	0.95–1.6 hrs	3.7–8.0 hrs
symmetry map	1020–1740	$3.24–3.92	9.0 s	8.5 s	0.4–1.6 hrs	0.7–4.9 hrs

- We use **a lot** of micro-tasks, which is why we emphasize **massive parallelism**.
- The **cost** is low enough to be **practical**, $3-$11 or $17. The depth layers algorithm is the most expensive, because there is a micro-task per neighboring patches, not per patch.
<click>
- Micro-tasks take **little time**! 5–9 seconds, and we believe that it is mostly interface overhead, like mouse movement.
- This means that our algorithms could complete in **~3 minutes** if we had enough HPs running in parallel.
<click>
- The main problem is the total time, which is long. The delay is entirely due to waiting for HPs to choose to perform the tasks.
    - This is a “**market problem**” in the Amazon Mechanical Turk.
    - % 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.
    - Would be helped if this were a **popular algorithm** (if we were generating a constant stream of micro-tasks).
    - It’s **anyone’s guess** how the market of HC will change in the future; there are new online labor markets that don’t have these properties.

Statistics

example	micro-tasks used	total $ cost	successful micro-task duration avg	successful micro-task duration median	algorithm delay until % complete 50%	algorithm delay until % complete 100%
normal map	1620–4340	$5.04–10.76	8.8 s	8.1 s	1.1–5.0 hrs	2.8–15.1 hrs
depth layers	2669–7620	$6.41–17.15	6.2 s	5.5 s	0.95–1.6 hrs	3.7–8.0 hrs
symmetry map	1020–1740	$3.24–3.92	9.0 s	8.5 s	0.4–1.6 hrs	0.7–4.9 hrs

- We use **a lot** of micro-tasks, which is why we emphasize **massive parallelism**.
- The **cost** is low enough to be **practical**, $3-$11 or $17. The depth layers algorithm is the most expensive, because there is a micro-task per neighboring patches, not per patch.

<click>

- Micro-tasks take **little time**! 5–9 seconds, and we believe that it is mostly interface overhead, like mouse movement.
- This means that our algorithms could complete in **~3 minutes** if we had enough HPs running in parallel.

<click>

- The main problem is the total time, which is long. The delay is entirely due to waiting for HPs to choose to perform the tasks.
    - This is a "**market problem**" in the Amazon Mechanical Turk.
    - % 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.
    - Would be helped if this were a **popular algorithm** (if we were generating a constant stream of micro-tasks).
    - It's **anyone's guess** how the market of HC will change in the future; there are new online labor markets that don't have these properties.

Summary

HC algorithms can work where automatic algorithms still cannot.

Identify the essential difficulty, and rephrase the algorithm in terms of micro human perception.

Problem	Micro-task	Combining Algorithm
depth layers	identify depth jumps	laplace equation
normal map	orient thumbtacks	bi-laplace equation
symmetry map	position point pair	none

If this were a Photoshop plug-in, would you use it?

Automatic algorithms **cannot**: this is **especially true for graphics** and visual problems.

Our key point is to: “Identify the essential...”

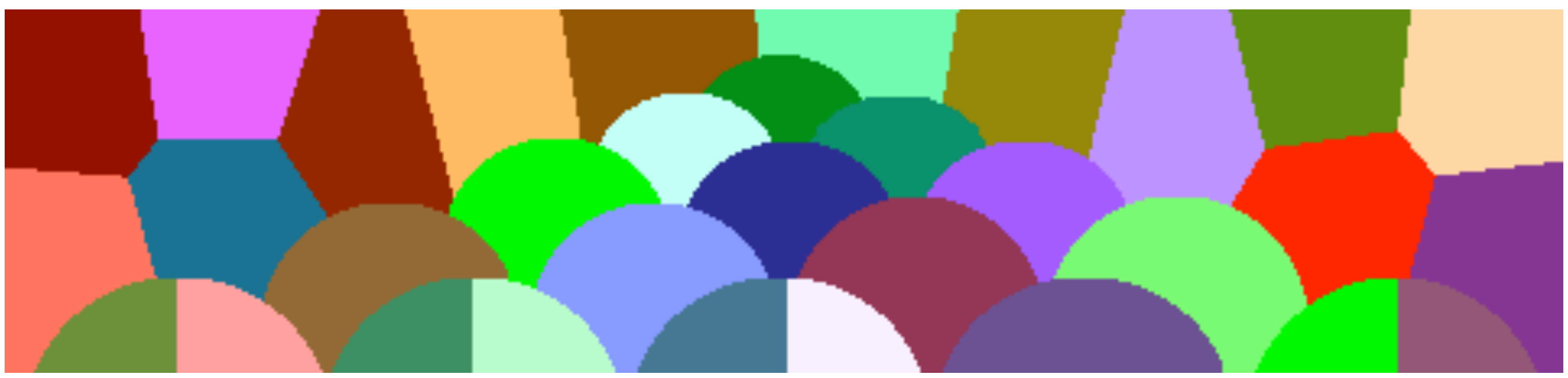
If this were a Photoshop plug-in...
Efficiency: timing, cost, quality control, optimizing micro-task designs (optimizing perceptual experiment design)

Thank you. Questions?

End

Accuracy

[flickr user fictures]



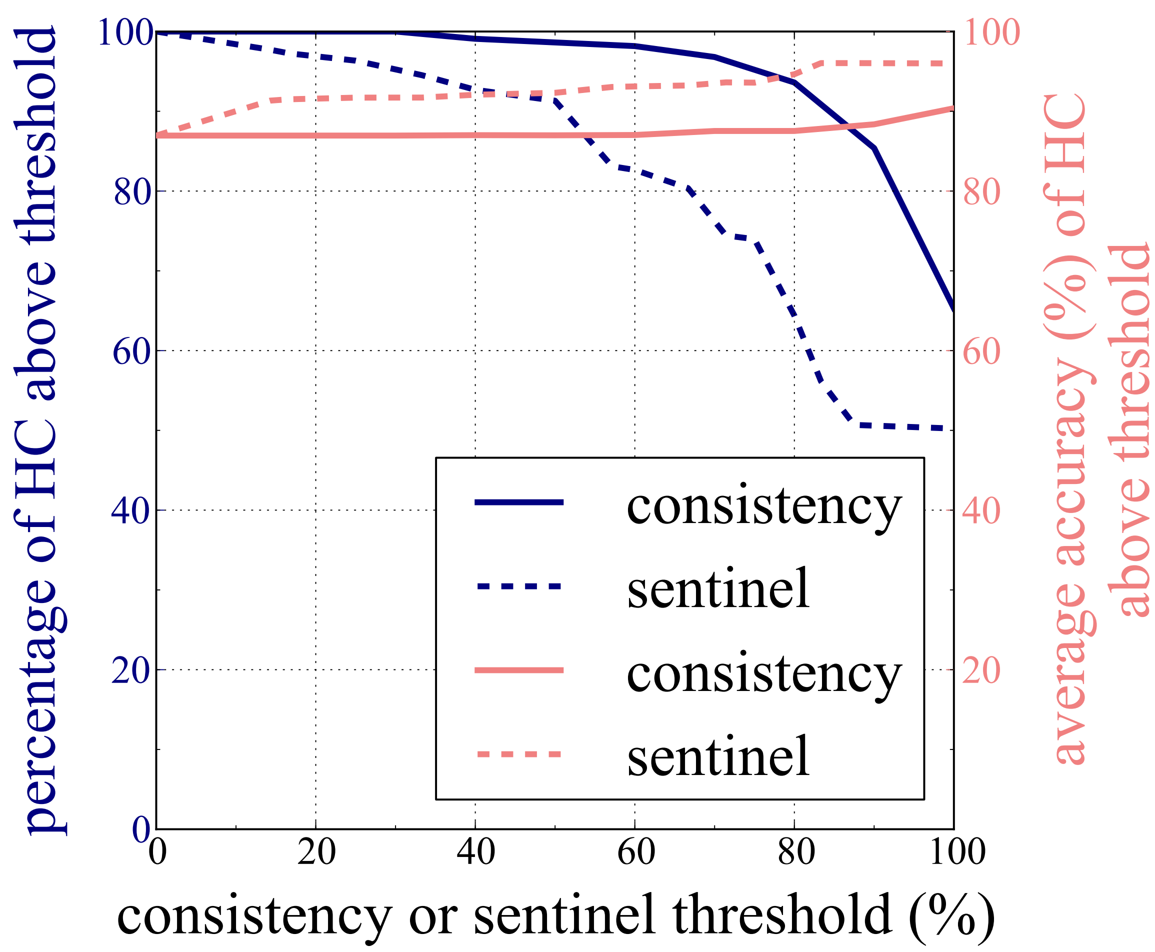
To test accuracy as a function of quality control parameters, we used this billiards image.
We ran many variations; the data you will see in the following graphs was generated from 4 calibrated runs.

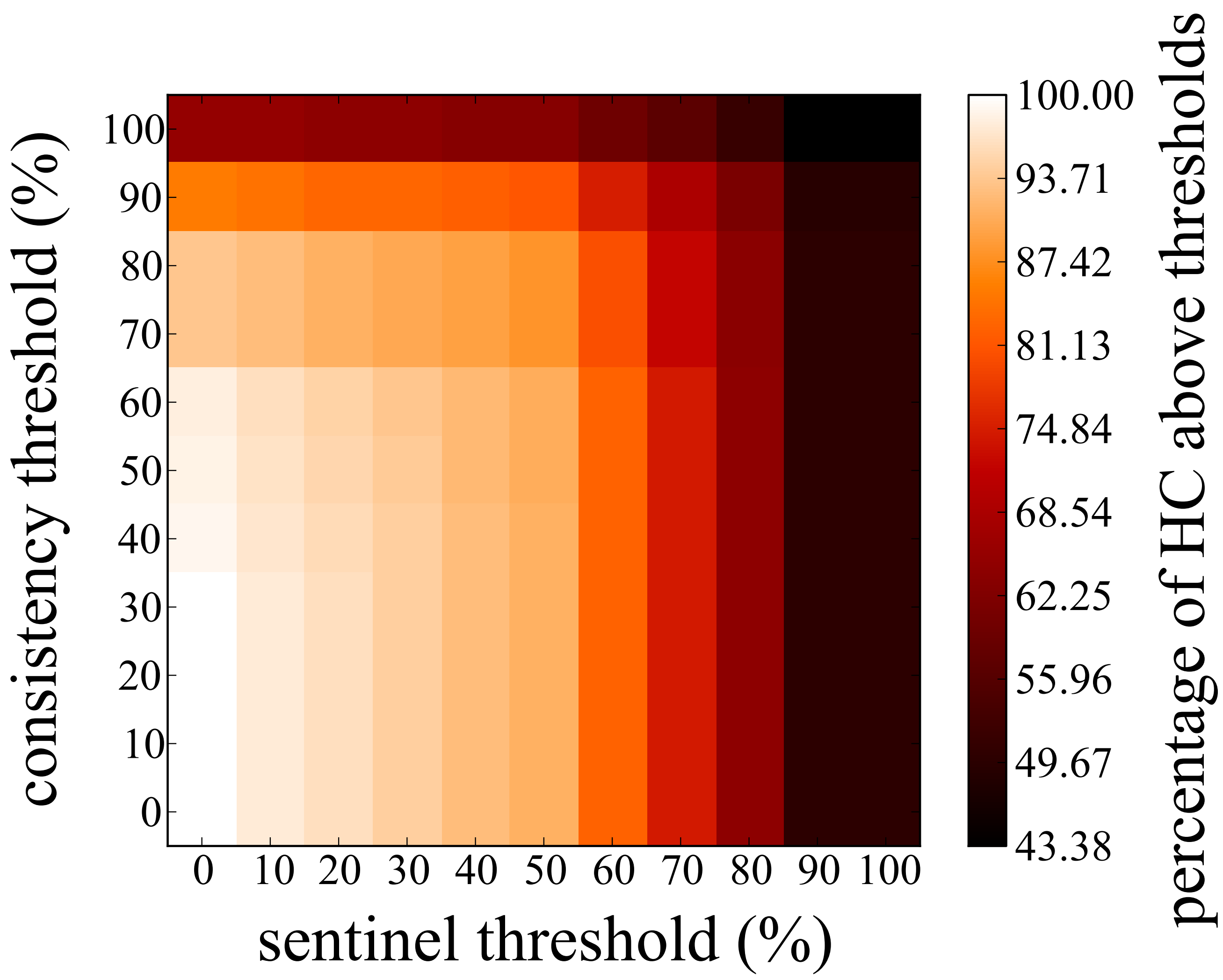
We varied the percent of queries in each batch that had to be internally consistent before throwing the entire batch out.
We varied the percent of sentinel queries in each batch that had to be accurate before throwing the entire batch out.

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.

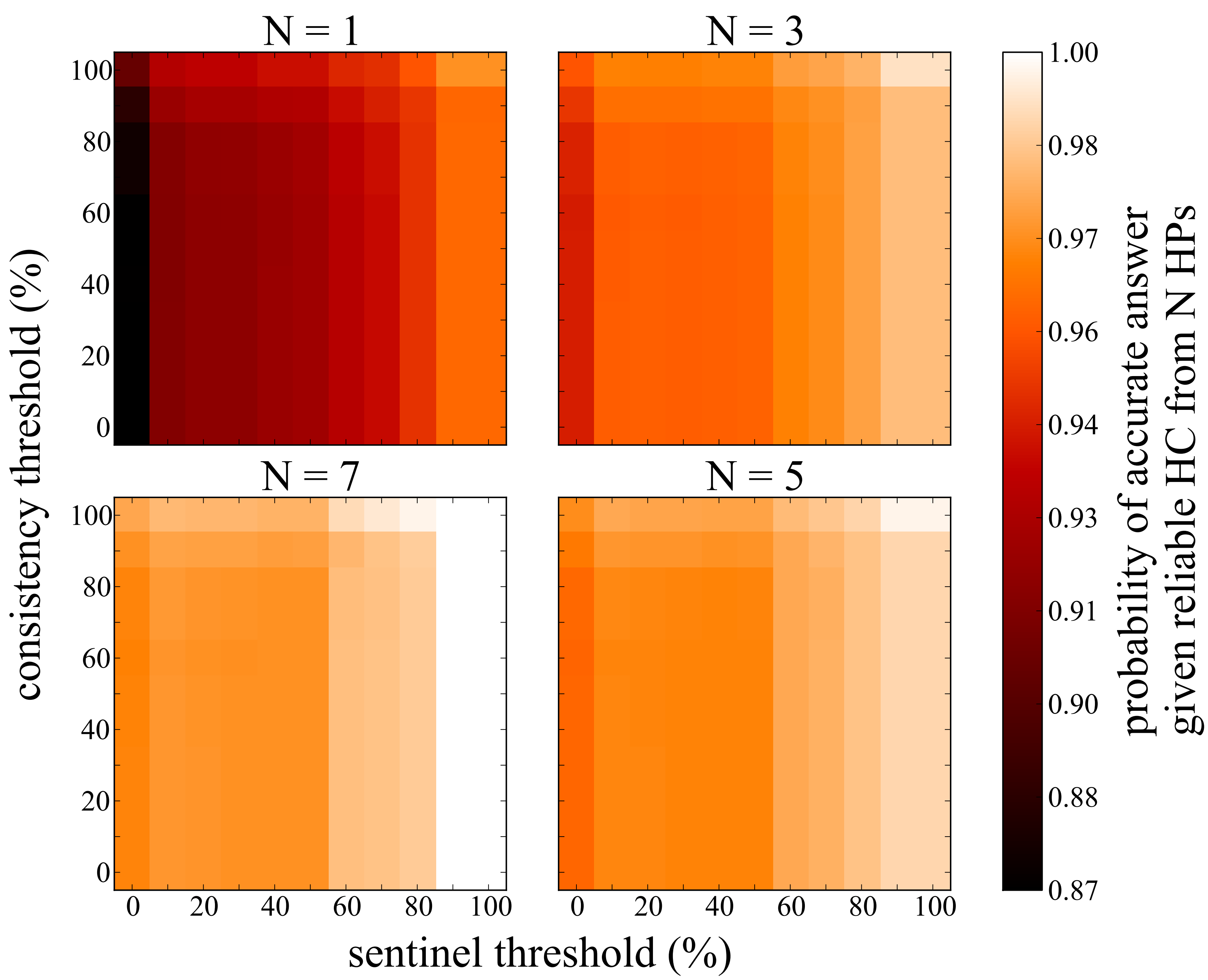
- The vast majority (94%) of HC batches are 80% consistent (or more), though only 65% are 100% consistent.
- Batches are more stratified in agreeing with sentinel operations. Only 74% were correct for 75% or more of the sentinel ops; only 50% were correct for 100%.

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

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.

Cost and Reliability

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

example	successful micro-task duration		algorithm delay until % complete	
	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").

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:

Related Work (2/6)

Online:

- [von Ahn 2008]
- [Little et al. 2010a,b] and [Bernstein 2010]
- [Bigham et al. 2010] and [Bernstein 2011]
- [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)).

[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.