Soylent Grid: it's Made of People!

Stephan Steinbach[*1], Vincent Rabaud[2] and Serge Belongie[2]
[1]Calit2 and [2]Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093, USA
`http://vision.ucsd.edu`

Abstract

The ground truth labeling of an image dataset is a task that often requires a large amount of human time and labor. We present an infrastructure for distributed human labeling that can exploit the modularity of common vision problems involving segmentation and recognition. We present the different elements of this infrastructure in detail, in particular the different vision Human Computational Tasks (HCTs) and Machine Computable Tasks (MCTs). We also discuss the impact of such a system on internet security vs. the current state of the art. Finally, we present our prototype implementation of such a system, named SOYLENT GRID, *on typical problems.*

1. Introduction

A researcher designing and implementing a computer vision system will often face an initial hurdle of acquiring a labeled ground truth dataset. Sometimes, a dataset is available off the shelf, but often the researcher must painstakingly collect and label the data by hand. Due to the large number of samples required to create adequate sets for training and testing, this can be a time consuming task.

For the sake of argument, let us consider the example depicted in Figure 1, where a researcher is labeling a dataset for a street sign recognition engine. In this case, the training data is made up of raw images from which street signs are first segmented, then labeled according to the isolated characters. This illustrates a common condition: while segmentation and text reading take a person a very short amount of time (a text labeling task in a recent project of our group took about 6 seconds per image and human user), that time (and hassle) grows prohibitively with the size of the dataset.

In order to reduce the labeling time, one available option is to employ an automated algorithm such as text detection or Optical Character Recognition (OCR) with thresholds set

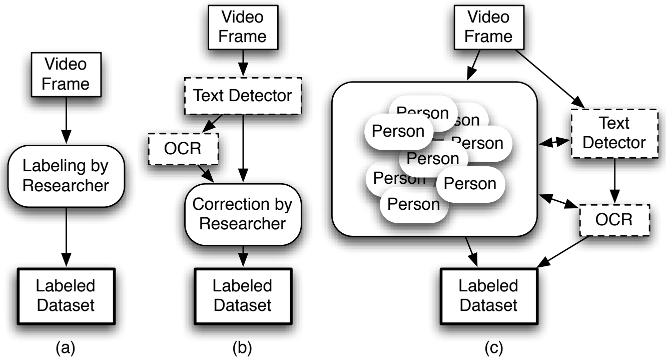


Figure 1. **Example of Distributed Interactive Computer Vision.** Legend: the boxes in the computational flow are rounded if they involve humans, and dashed if they require a computer algorithm. **(a)** When labeling images containing street signs, a researcher can crop the sign areas from the original pictures and then label them by hand. **(b)** On the other hand, by developing a text detector and an OCR engine, the researcher can greatly reduce the amount of data he actually needs to label. **(c)** We propose to go even further in the process and have the researcher only give directives to many human users distributed on a network. These users would perform the task for other reasons (*e.g.* , security) with labeling as a byproduct.

to have virtually 100% recall (at the expense of precision) to reproduce, or simulate, the rough sequence of steps in the labeling task. The researcher can then fill in the gaps by checking for errors (which is much faster than doing the task itself) and correcting mistakes (which costs as much as the original test).

However, the researcher could also take advantage of the atomic nature of the labeling subtasks. The problem would now be ideal for an application of a "Distributed Human Computation" (DHC, [15]) system. A DHC system distributes a task that is difficult for machines, but easy for a pool of human volunteers.

Furthermore, having a human being solve a visual recognition problem is a superset of the CAPTCHA paradigm [21], the now ubiquitous application of Turing tests. In a

[*]Present address: San Francisco, CA 94116, USA.

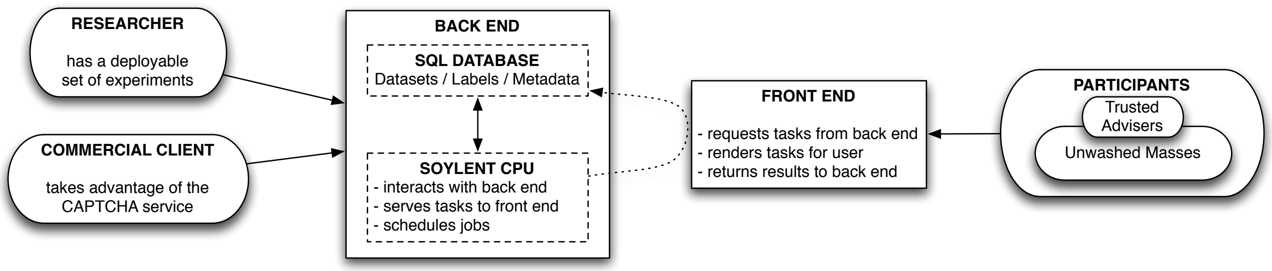

RESEARCHER
has a deployable set of experiments
COMMERCIAL CLIENT
takes advantage of the CAPTCHA service
BACK END
SQL DATABASE
Datasets / Labels / Metadata
SOYLENT CPU
- interacts with back end
- serves tasks to front end
- schedules jobs
FRONT END
- requests tasks from back end
- renders tasks for user
- returns results to back end
PARTICIPANTS
Trusted Advisers
Unwashed Masses

Figure 2. **Structure of a SOYLENT GRID.** The users benefiting from our grid can be of two kinds: researchers (needing some information analysis) and commercial clients (that simply use the Turing test generation service). These providers impose their constraints to the back end MySQL server by giving their datasets and describing the tasks to be performed by the end users. Next, when a participant requests a Turing test, the Java front end interacts with the server to get a Turing test and also tests the validity of the provided answers. Any information input by the participant (like the answer itself or the time taken to answer) is also sent back to the server for statistical purposes.

CAPTCHA, a user has to transcribe a piece of text that has been deformed so that it is hard to decipher for a machine, often used as part of an approval process for access to a web resource.

The labeling problem can be seen as a more complex kind of CAPTCHA, where the true answer (or label) is unknown. Therefore, what we propose is a visual recognition test based distributed infrastructure for labeling. At the beginning of such a grid computation, a researcher submits a job. On the other end are the visual tasks that a user solves for other purposes, like passing a security check, and thereby provides a certain form of labeling as a byproduct. Furthermore, what represents a difficult machine learning problem changes over time. If the newly proposed CAPTCHA is to take advantage of the forefront of machine intelligence/computer vision research, our system, nicknamed SOYLENT GRID, must accept new tasks that match the current state of the art.

We will first review some previous work in Section 2. We will then detail the different elements of a SOYLENT GRID in Section 3, as well as how to use it and characterize its behavior. We will finally present in Section 4 some real-life examples on which SOYLENT GRID is being tested.

2. Prior Work

The spirit of our design reaches back to Turing's original *imitation game* [20]. Rather than focusing on the common interpretation that there are things that only humans or only machines can do, we examine the ever growing hazy middle area where both humans and machines can contribute.

Recent work on breaking popular CAPTCHAs [18, 12, 5] suggests that the current avenue of research into CAPTCHAs may be drawing to a close: in the quest to stave off automated cracking algorithms, CAPTCHAs are becoming too difficult for humans to pass. This trend, as well as the emergence of new kinds of Turing tests like Microsoft's Asirra [14] and CAPTCHA "mashups" such as HotCaptcha [4], suggests a return to more concrete problems of machine learning for the design of CAPTCHA problems as suggested by [16].

Online Machine Intelligence, including Interactive Computer Vision, is also a growing field. Algorithms such as SEVILLE [9] involve human help in a boosting framework when necessary to improve the performance of a pattern recognition system. The use of humans in the loop for various tasks has a very strong literature in internet security (CAPTCHA, SpamNet), but also has several antecedents in machine learning ground truth labeling.

To get humans interested in participating, various types of delivery mechanisms are used, including CAPTCHAs themselves, ground truth labeling systems [2] hidden under a game [22, 23], proofreading [6], or even micropayments [1]. Also, as in the case of our motivating problem domain, humans can participate as *remote sighted guides* to assist blind individuals [11].

Moreover, Distributed Human Computing itself is still relatively new, with papers [15] focusing on how to gather meaningful information from massively distributed systems, with purely-human based processing distributed according to a cost (usually factoring in the "cost" of human interest). Nonetheless, the infrastructures for parallel computing have been in place for a few decades and the Internet itself provides a rich substrate to solve computationally intensive problems like signal processing [8] or prime number cracking [3].

3. Soylent Grid

As depicted in Figure 2, the system is expressed as a relationship between three groups of human users (Participants, Commercial Clients, Researchers) and several foundation elements (Front end, Back end, Database). Jobs are processed by the grid, on datasets, and can be represented

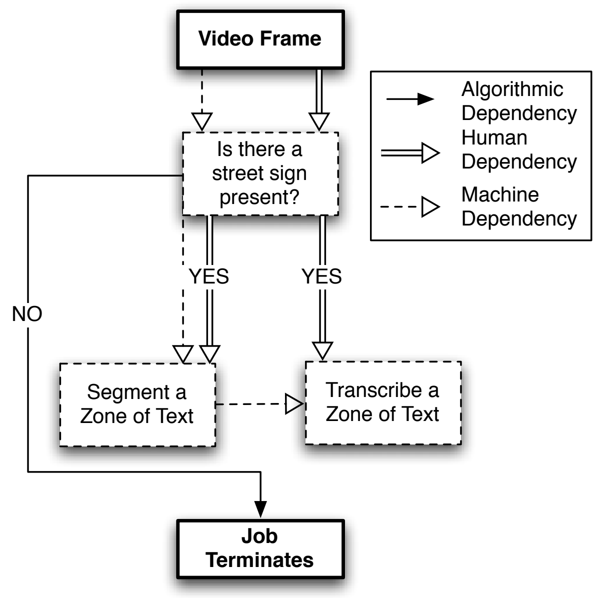


Figure 3. **Example of a job DAG**. This figure details the DAG of a Street Sign Labeling job in the SOYLENT GRID. Each dotted-bordered block is actually a task and may be proposed to a machine or a human participant, assuming its specific human/machine dependencies are met. When all the tasks are complete, the job is considered done (per frame). For this job, the machine processing option requires that OCR follow segmentation, while a human participant could complete the last two tasks out of order.

as Directed Acyclic Graphs of Tasks (*cf*. Figure 3).

3.1. Users

There are three main groups we define as having an interaction with the SOYLENT GRID:

1. **Researchers**: they define jobs on the back end, submit data and connect on-line input streams.

2. **Clients**: they would like to take advantage of the CAPTCHA or spam filtering capabilities of the grid.

3. **Participants**: they complete the tasks and can either be "Trusted Advisers" or "Unwashed Masses".

A Trusted Adviser (TA) is assumed to provide unequivocally correct answers. If an unambiguous task is run on a network of only Trusted Advisers, the SOYLENT GRID effectively reduces to the same system as proposed in Distributed Human Computation[15]. There is no confidence checking with Trusted Advisers, but outliers are still rejected as in standard statistical models.

On the other hand, Unwashed Masses (UM) users contribute by completing the CAPTCHA tasks presented by the SOYLENT GRID. By chaining several tasks together, some having already been solved and others not, the system creates CAPTCHAs that can be embedded by clients in web pages. The tasks that have known answers are treated as Turing tests, and failing them automatically rejects the answers given for the unsolved tasks. We do not assume any sort of cookies or other identity tracking techniques. For our purpose, each instance of a CAPTCHA request for the grid represents one unique contribution from some UM participant. If the Turing tests are passed, the labels given for the unsolved tasks are accepted into the results database.

Because of the UM contribution of our system, we impose a gentle rule of thumb: no task presented to an UM participant may take longer than a certain amount of time. In practice, we choose 10 seconds. What that means is not that any single task takes no longer than 10 seconds, but rather that the combination of tasks together takes no longer than 10 seconds. Tests that empirically are deemed to take too long can be pulled or divided into simpler problems, for example, by cropping the image or applying some filter prior to asking the test question.

In the case where the grid is only composed of Unwashed Masses participants, and where there is only one job, composed of two CAPTCHA tasks (one known, one unknown), SOYLENT GRID reduces to reCAPTCHA [7].

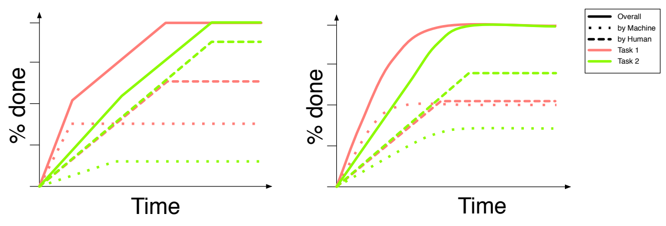
This division of participants suggests a spectrum: we never sit in the case were there is only UM or only TA contribution, but rather employ some hybrid of both in order to process jobs effectively. TA contribution is expensive: these people must be employed or volunteer in some way; UM contribution is cheap, but their responses must be filtered and tested, and therefore UM contribution is less efficient.

3.2. Tasks

A *task* is a single operation that is applied to an image as part of a *job*. To clarify, the street sign labeler described in Figure 3 would be a *job*, while the "Is there a street sign in this image?" step would be a *task*.

A task is completed by sampling the response of computational and participant responses to the task. The exact number of samples that are collected, and the balance between computational and participant responses before a task may be considered "complete" by the grid is governed by two metrics, *MCT-confidence* and *task ambiguity*.

Machine Computable Task confidence, or *MCT-confidence* for short, is a value between 0 and 1 that summarizes the success rate of a machine-based approach to a given task. This is used to grade or weight the response of a computational contribution. For example, if for the sign reading task there are no computational solutions that provide any correct positive result, the MCT-confidence is 0 and the task can only be completed by human input. For a MCT-confidence greater than 0, some mix of computational contribution and human contribution will be used, based on available computational resources. Confidence depends



(a) (b)

Figure 4. **Labeling and Active Learning.** These simplified plots describe the case where two tasks are submitted to the grid, the second one depending on the completion of the first one. While the amount of data labeled by humans grows linearly over time, the data labeled by machines can grow differently according to the algorithms on the SOYLENT CPU. **(a)** In a strict labeling task, the quality of the labeling by computers will stall after some time because of the limitations of the algorithms. **(b)** In an *active learning* paradigm, the computers take advantage of the labeling already done by humans to improve the efficiency of the underlying algorithms and therefore complete the task sooner.

on the algorithms used and their interaction with the users (*cf*. Figure 4).

The *task ambiguity* refers to the spread of the possible label distribution. The larger the ambiguity, the more trials it will take to capture the accurate distribution of correct labels for this task. For example, when asking to identify the digit “5” in a blurry picture, a bimodal distribution peaked on “5” and “S” would reflect the ambiguity of the task.

3.3. Categories of Tasks

We have identified several classes of computer vision related tasks; we will detail each of them and demonstrate some security statements. For this purpose we will consider that the images used in the tasks are 800×600, which approximately translates to $\mathcal{N} = 5 \times 10^5$ pixels. Our goal is to show how the soylent tasks can compete with CAPTCHAs [21], for which a random guess of the answer has a probability $\frac{1}{36^5} \simeq 10^{-8}$ (assuming a typical five letter/digit guess) while an engineered (computational) solution [5] allegedly succeeds in 5% of the images in the worst case (up to 100% for certain CAPTCHAs). Some illustrations are provided in Figure 5.

“Name that Thing”

The first task is related to *object recognition*. When presented with a picture, the user needs to identify the components in the scene and then answer a question about the content: “what is this object?”, “what is on the table?”, “what is the color of the apple?” *etc*. A toy example could be to identify a letter displayed on the screen while a more practical example could be an OCR task or even simply a CAPTCHA.

In most languages, an average speaker uses a 1000-word daily vocabulary. When asked for a word describing the scene, a random guess would succeed with a high probability of 10^{-3}. An easy way to decrease this success rate, is to propose an identification task i times consecutively. A random guess would then succeed with a probability of only 10^{-3i}, which competes with CAPTCHAs after three trials.

“Generalized Where’s Waldo”

The next set of tasks concerns *object detection* and uses a simpler interface where only clicking is needed. The human participant is presented with a picture of a scene and different types of questions are asked using the ability of the user to locate objects: “click on the car,” “click on the round object,” “click on two similar objects,” *etc*.

In order to make this test more secure, and less prone to successful random guess, the user is actually asked to click several times on an object or to draw inside it. Once enough pixels have been selected, the task is passed. This method bears some resemblance to [23].

Consider as an example problems of the following form: “which pictures depict the same object?” This example is used in [14]. For such a test, let us assume 16 images are displayed in a square grid, with k of them depicting the same kind of object. Given the $\sum_{k=2}^{k=8} C_{16}^k$ possible k-tuples (we keep k between 2 and 8 to make it easier for the user), we have a random probability of success given by:

$$p_{\text{similar}} = \frac{1}{\sum_{k=2}^{8} C_{16}^k} \simeq 3 \times 10^{-5} \qquad (1)$$

It is also important to notice that an unlabeled picture can be included among the 16, hence providing a soft labeling of it (but doubling the probability above, as the right answer with or without this extra picture has to be considered as valid).

For a question of the type “How many . . . ?”, we decide to ask the question in a way that does not require the user to count and use the keyboard: “click once and only once on each instance of the object.” For the sake of argument, let us assume there are usually between 1 and 10 instances of the object and that the average area of an object is the generous value of $\mathcal{A} = 10^4$ pixels. Now, when presented with a scene where there are k instances of this object, there are $\mathcal{A}^k$ valid combinations of clicks, while there are $\sum_{i=1}^{10} C_{\mathcal{N}}^i$ possible combinations of clicks. Consequently, a random guess has the following average probability of success:

$$p_{\text{count}} = \frac{1}{10} \sum_{k=1}^{10} \frac{\mathcal{A}^k}{\sum_{i=1}^{10} C_{\mathcal{N}}^i} \simeq 4 \times 10^{-12} \qquad (2)$$

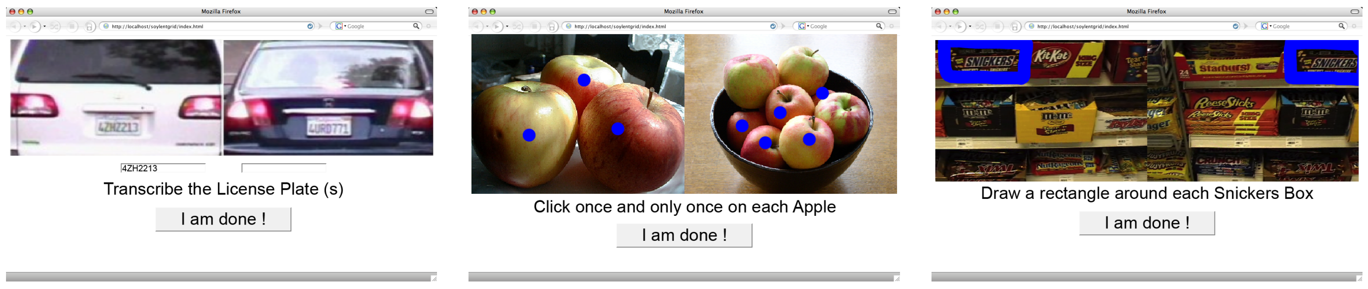

Mozilla Firefox
4ZH2213
Transcribe the License Plate (s)
I am done !
Click once and only once on each Apple
I am done !
Draw a rectangle around each Snickers Box
I am done !
(a)
(b)
(c)

Figure 5. **Examples of SOYLENT TASKS.** The illustrated tasks each contain two instances displayed in the browser: one has already been answered and is used as a Turing test, the other one has to be answered for labeling. **(a)** The first type of task is a simple OCR. **(b)** The second task is a counting task: in order to simplify the task and actually increase tremendously its security, the user is asked to click once and only once on each instance of an object. A byproduct of this task is a distribution of where an object is clicked on and therefore of what is most important to a human labeler. **(c)** The third task is related to segmentation: by using a thick pencil (allowing a certain margin of error), the user has to segment objects.

"Trace This"

Tasks of the form "Trace This" deal with *image segmentation*. As in the previous tasks, the user is presented with an image of a scene and is then asked to draw the (possibly partial) contour of an object as in [19]. Applicable metrics for segmentation agreement are presented in [17].

Of course the answer is accepted with a a certain tolerance. Estimating the robustness to a random attack is unfortunately not as trivial in this kind of task.

"Hot or Not"

We use the "Hot or Not" designation to describe the category of task involving human preferences represented by partial orderings or numerical scales. Several websites, like `http://hotornot.com` or `http://thefunniest.info`, have already harvested a rich database of such ratings. A SOYLENT TASK could then simply ask the user to choose the k pictures that are the most pleasing to the eye, for example, leveraging samples from the extreme ends of the label distributions. Such an approach is in fact used by [4].

3.4. Jobs

The Street Sign Recognition pipeline was discussed previously, where a frame of video is processed through three stages. Translated to SOYLENT GRID, the logic is shown in Figure 3.

Internally, this is implemented as a Directed Acyclic Graph attached to each item of data submitted to the job needing some labeling/processing. The researcher submitting the job defines this DAG by specifying the different tasks that need to be processed (segmentation, recognition, ...) as well as their order relationship ("first segment the street sign") and logic ("if there is no street sign in the image, break")

When an item of data is randomly selected for processing, the SOYLENT CPU selects a node in its DAG as the next test to be completed. The node is selected with respect to the distribution of answers it has already received: if the distribution is very peaked on one answer, then this task is probably solved or too easy: it can be marked as "labeled" and pushed to the "done" stack. On the other hand, if the distribution has a large standard deviation or multiple maxima, the researcher is warned for a possible ambiguity in this item.

Note that dependencies in the job/DAG can be computational-path specific: if the street sign labeler is processed entirely by Participant contribution, then the Segmentation and Object Recognition phase can happen at the same time; for the computational solution, the Object must be segmented before the Optical Character Recognition can occur.

3.5. Throughput and Processing Time

The grid, as a computational system, can be modelled in very general terms using standard mathematical tools.

Consider a job with only one task that is used to label n items. The task is only performed by m Trusted Advisers and it takes at most t_{task} to complete. The job will trivially take at most $n\frac{t_{\text{task}}}{m}$ to achieve.

On the other hand, if the job is processed by only Unwashed Masses, several tasks need to be asked per job: k to label images (usually, $k = 1$ for simplify) and the rest as a Turing test. If the type of task has a MCT-confidence of C, and we want a chance of false-positive (right answer but by a machine) of less than 10^{-5} (typical value required for a CAPTCHA), the task has to be asked at least $\lceil \log_C(10^{-5}) \rceil$ times. As seen previously, this does not necessarily mean that the task has to be asked several times in a row, it can be asked several times at once by combination. In both cases, it will take a time of:

$\left(k+\left\lceil\log_C\left(10^{-5}\right)\right\rceil\right)t_{\text{task}}$ and these combinations of tasks have to be asked $\frac{n}{k}$ to have all the data labeled once.

To obtain a distribution of answers, the tasks also need to be labeled several times, s times. Therefore, the total time required to label the n items by m' UMs s times is:

$$s\frac{n}{k}\frac{\left(k+\left\lceil\log_C\left(10^{-5}\right)\right\rceil\right)t_{\text{task}}}{m'} \tag{3}$$

By looking at the above equation and comparing it with the TA case, it appears that using UMs is profitable as long as the overal time competes with the TA case. This is equivalent to have $\frac{s}{k}\left(k+\left\lceil\log_C\left(10^{-5}\right)\right\rceil\right)\frac{m}{m'}$ is close or inferior to 1, *i.e.* if (a) there are enough UMs (m' big) (b) the MCT-confidence of the task is low enough (C small) (c) the task is not too ambiguous for humans (s small). This tradeoff simply states that the better the machines are at solving a task (and therefore at fooling the grid), the more human contribution is needed. It also defines a boundary for how good the level of security is for a Turing test.

The throughput of UM contribution is also reduced by the possibility of a lurking adversary or human error. We must, however, rely on statistical outlier rejection in order to handle those cases. By providing a margin of MCT-confidence against machine intelligence attack of 10^{-5}, we have provided a fairly sizeable probabilistic buffer against an outright malicious attack.

Furthermore, there are two kinds of attacks. In the one case, where an attacker wishes to access what is hiding behind the grid, the previous probability holds. If, however, an attacker wishes to damage the SOYLENT GRID itself, there is in fact more protection: in order to do this, the attacker must correctly lie on each task that is not a test, yet tell the truth on each test. Assuming there are two tasks per session (as in reCAPTCHA), the attacker has a 50% chance of guessing the position of the labeling task: damaging the grid is in this case twice as hard as passing through it.

Returning to the issue of throughput, however, our labels per contribution figure needs to be combined with the size of our grid, in terms of "hits per unit time", as we are not dealing with dedicated users. If u is the number of users per second, the throughput of the grid (in tasks/second) is therefore:

$$\frac{k}{k+\left\lceil\log_C\left(10^{-5}\right)\right\rceil}u \tag{4}$$

Asking simpler tasks (hence increasing C), or putting the system under attack and lowering the "acceptance" thresholds (both cases result in higher amounts of double-checking), would effectively lower the fraction and result in a less efficient, slower grid.

4. Proposed Applications

Some of the soylent tasks have been implemented and are currently part of two larger scope projects meant to help blind people perceive their environment.

4.1. Soylent Texty

In the Soylent Texty, the soylent tasks are used to label images to train an outdoor sign reader. The processing outline involves a sign detector (based on color histograms) followed by a text detection algorithm (from [24, 13]) and an OCR (HP's Tesseract). The first two steps require some training and therefore some labeling. We simply added a tight threshold of success to these algorithms and every image not passing it was simply added to the pool of images needed to be solved.

This tight threshold of success represents a low MCT-confidence, and using the completely non-specialized off the shelf OCR Tesseract represents a medium level of confidence ($C = 0.3$ for each letter). It takes a human approximately $t_{\text{task}} = 0.8$ seconds per letter (for an overall task chosen to be of $t_{\text{total}} = 10$ seconds), so we set the number of unlabeled letters to ask (k) as the biggest integer verifying:

$$\left(k+\left\lceil\log_C\left(10^{-5}\right)\right\rceil\right)t_{\text{task}} \leq t_{\text{total}}$$

i.e.

$$k = \left\lfloor\frac{t_{\text{total}}}{t_{\text{task}}} - \left\lceil\log_C(10^{-5})\right\rceil\right\rfloor = 2$$

Our version of SOYLENT GRID was implemented to protect a wiki from spammer edits by proposing a SOYLENT TASK.

4.2. GroZi

The aim of the GroZi project (`http://grozi.calit2.net`) is to develop a grocery shopping aid for the visually impaired. The application of SOYLENT GRID to this project is thus far a prototype. The goal is to develop a wearable computer with a camera that can lead a blind user to a desired product in a grocery store by analyzing the video stream.

The degree to which human beings could participate in the system (as remote sighted guides) ranges from none at all to virtually unlimited. If no human user is involved in the loop, only computer vision algorithms solve the identification problem. But in principle, if there were an unlimited number of humans in the loop, all the video frames could be submitted to a SOYLENT GRID, be solved immediately and sent back to the device to guide the user.

We decided to adopt an approach similar to Soylent Texty to train an algorithm to perform recognition on certain grocery products. The challenge in building such a database is that it is very specific: the database needs to contain images of grocery products in a real environment. To our knowledge, there is no such a database, hence the demand for SOYLENT GRID.

5. Conclusion

In this paper we established the infrastructure of a novel human- and machine-based computing grid for labeling large amounts of image data while offering robust Turing tests as a byproduct. We have also demonstrated the flexibility of the grid with respect to the users and the tasks, as well as its robustness in producing results and in facing possible hacker attacks.

In future work we will explore deeper connections to several of the cited relevant works, *e.g.* by packaging the tasks as a game, as in ESP GAME. Finally, the main focus of the development is now centered on the incorporation of active learning in the processing loop so as to diminish the number of items that require human processing.

Acknowledgment

This work was funded in part by NSF Career Grant #0448615, the Alfred P. Sloan Research Fellowship and the UCSD Division of the California Institute for Telecommunications and Information Technology, Calit2. The authors would also like to thank S. Agarwal, B. Ochoa and P. Dollár for helpful discussions. The paper title is inspired by the film *Soylent Green* (Metro-Goldwyn-Mayer, 1973).

References

[1] Amazon Mechanical Turk. `http://www.mturk.com`.

[2] Google Image Labeler. `http://images.google.com/imagelabeler/`.

[3] Great Internet Mersenne Prime Search. `http://www.mersenne.org`.

[4] HotCaptcha. `http://hotcaptcha.com`.

[5] OCR Research Team, Automated Protection From Automated Systems. `http://ocr-research.org.ua`.

[6] Project Gutenberg Distribute Proofreader. `http://www.pgdp.net`.

[7] reCAPTCHA. `http://recaptcha.net`.

[8] SETI@Home, Search for Extra-Terrestrial Intelligence. `http://setiathome.berkeley.edu`.

[9] Y. Abramson and Y. Freund. Active learning for visual object recognition. UCSD Technical Report, 2005.

[10] H. S. Baird and D. P. Lopresti, editors. *Human Interactive Proofs, Second International Workshop*, volume 3517 of *Lecture Notes in Computer Science*. Springer, 2005.

[11] J. Brabyn, Membec, W. Crandall, and W. Gerrey. Remote reading systems for the blind: A potential application of virtual presence. In *Engineering in Medicine and Biology Society, 1992. Vol.14. Proceedings of the Annual International Conference of the IEEE*, volume 4, pages 1538–1539, 1992.

[12] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski. Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs). In Baird and Lopresti [10], pages 1–26.

[13] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. *IEEE Computer Vision and Pattern Recognition*, 02:366–373, 2004.

[14] J. Douceur, J. Elson, and J. Howell. Asirra. `http://research.microsoft.com/asirra`.

[15] C. Gentry, Z. Ramzan, and S. Stubblebine. Secure distributed human computation. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 155–164, New York, NY, USA, 2005. ACM Press.

[16] D. P. Lopresti. Leveraging the CAPTCHA problem. In Baird and Lopresti [10], pages 97–110.

[17] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[18] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In *IEEE Computer Vision and Pattern Recognition*, pages I–134– I–141, 2003.

[19] B. Russell, A. Torralba, and W. T. Freeman. LabelMe, the image annotation tool. MIT lab memo AIM-2005-025, Sep 2005. `http://labelme.csail.mit.edu`.

[20] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, October 1950.

[21] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt*, pages 294–311, 2003.

[22] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the ACM CHI*, 2004. `http://espgame.org`.

[23] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64, New York, NY, USA, 2006. ACM Press. `http://www.peekaboom.org`.

[24] A. Yuille, D. Snow, and M. Nitzberg. Signfinder: using color to detect, localize and identify informational signs. In *International Conference on Computer Vision*, pages 628–633, 1998.